École de Printemps

Cryptographie et sécurité informatique

Sorèze, Tarn, 24-28 avril 2006

PRÉSENTATIONS & ARTICLES

Table des matières

La cryptographie par la théorie des graphes

Jean-Jacques Quisquater

Professeur à l'Université de Louvain
Belgique

Graph Theory and Cryptography

Jean-Jacques Quisquater
jjq@dice.ucl.ac.be
April 24, 2006
http://uclcrypto.org

Ecole de printemps -

Warning: Audience may be addicted by Powerpoint.
Use with moderation.

Menu

- IC, graph, crypto, CRYPTOgraphIC
- Cryptology
- Encryption
- Identification
- Integrity: hash functions
  - Large graphs and cryptology
  - Cayley graphs
  - Expanders (SL(2,*p*))
  - Hash function (Zémor-Tillich)
  - (Generating S_m, A_m)
- Research
- Conclusions

Cryptology

- Confidentiality, integrity, identification (design and attacks: cryptography and cryptanalysis)

Encryption by graphs

- 2 oriented graphs
- Nodes are messages and ciphertexts
- Labelled edges chosen by secret key
- Cayley graphs
- **Yevgeniy Dodis** and Adam Smith, "Entropic Security and the Encryption of High-Entropy Messages", *Theory of Cryptography Conference (TCC)*, February 2005.

Identification: isomorphism

- Alice (prover) and Bob (verifier)
- Two isomorphic public graphs G1 and G2 (same graph with 2 representations)
- Isomorphism P is private (secret)
- Isomorphism between graphs is a difficult problem
- Proof of knowledge of isomorphism without revealing it
- Use of a new random isomorphic graph G3: there are two permutations P13, P23: Bob chooses one and Alive gives it.

Integrity: Hash functions

- Integrity: send a message *M* with an added information for detection of change
- Detection or correction codes if noise
- Cryptographic hash function *h*(), value protected by signature, if malicious context
- Hash function: given *h*(*M*), difficult to find another message with same value *h*(*M*): other definitions
- collisions

Uses of hash functions

- Integrity (downloads)
- Signatures (long messages)
- Key exchange (export)
- Encryption
- Identification
- Lot of uses in complex systems (airplane, …)

Short story of hash functions

- Nothing at the beginning: problem! No research
- Theory not very strong (Merkle-Damgaard): iterative design (no precomputation, no parallel computation)
- Standards without hash (ISO-9796)
- MD2,4,5 (Rivest): hash of 128 bits, 1990 RFC1319-1321: used in many internet protocols
- SHA-0, SHA-1: design based on MD-5, by NSA, standard NIST; 1993-1994: 160 bits
- SHA-256, …: recent standards
- RIPEMD (NESSIE):
- http://paginas.terra.com.br/informatica/paulobarreto/hflounge.html

Attacks

- *N* outputs
- Exhaustive search (*N*/2 computations)
- Birthday paradox: collisions (square root of *N* computations)
- Time-memory computations
- http://keylengh.com
- Specific attacks (cryptanalysis)
- Notion of broken

- MD4, MD5, SHA-0, … « broken » (August 2004) by Chinese scientists
- See program codes in
  http://www.stachliu.com.nyud.net:8090/collisions.html
- SHA-1 (random collision in 2^{63} computations)
- State-of-the-art at
  http://www.csrc.nist.gov/pki/HashWorkshop/program.htm

October 31-November 1st, 2005

- Confidence in others?

Replacing the hash function

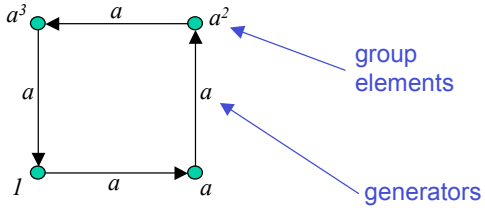
- Not easy
- No IETF protocol (ipsec, ssl, tls, …) is enough flexible: changes will be needed
- Changes of format: a short-time possible solution

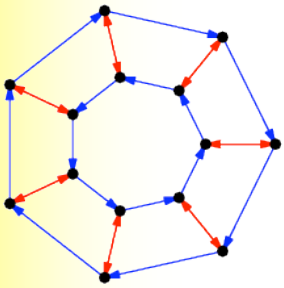
Large graphs and Cryptology

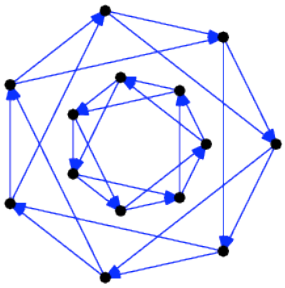
- *Protocols*: graph isomorphism (theoretical, pedagogical, not efficient)
- *Primitives* for integrity: Zémor-Tillich fonction (Eurocrypt'91, Crypto'94)
- *Attacks*: use of expanders
- *Proofs* of security: use of expanders
- Physical *distribution* of secrets: (delta-*D*)-graphs (Q., 1998), Cayley graphs

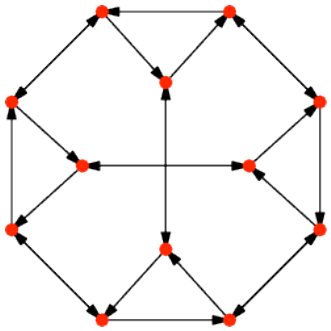
Cayley graphs (1)

- Generator set *A*
- Generated group *G*
- Graph:
  - group elements as vertices
  - edges defined by generators

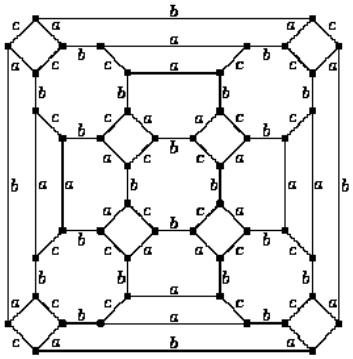


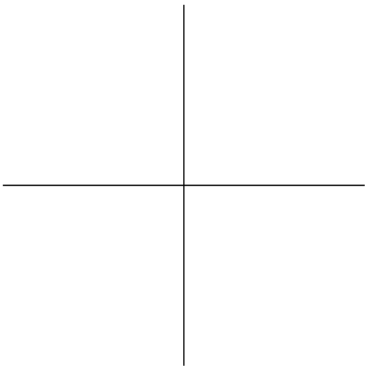


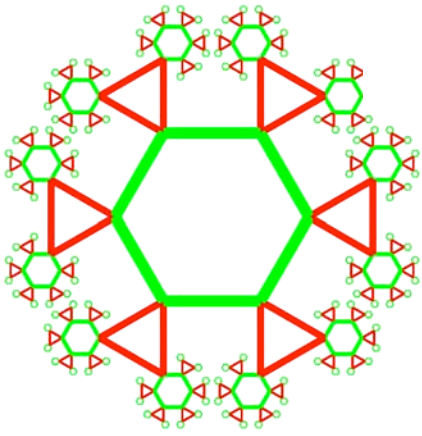




Cayley graphs (2)

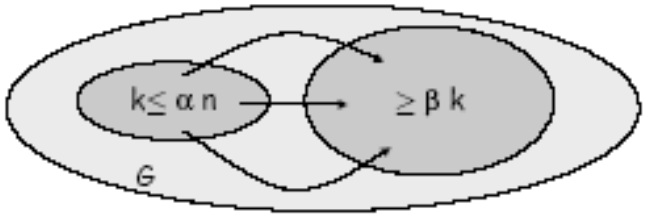






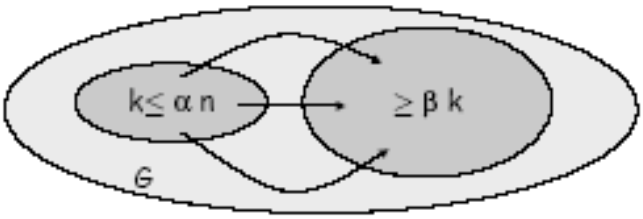
Expander (1)

- Graph *G* with *n* vertices (often fixed degree)
- All subsets of *k* nodes have at least β*k* neighboring vertices
- Expanding constant (isoperimetric constant)

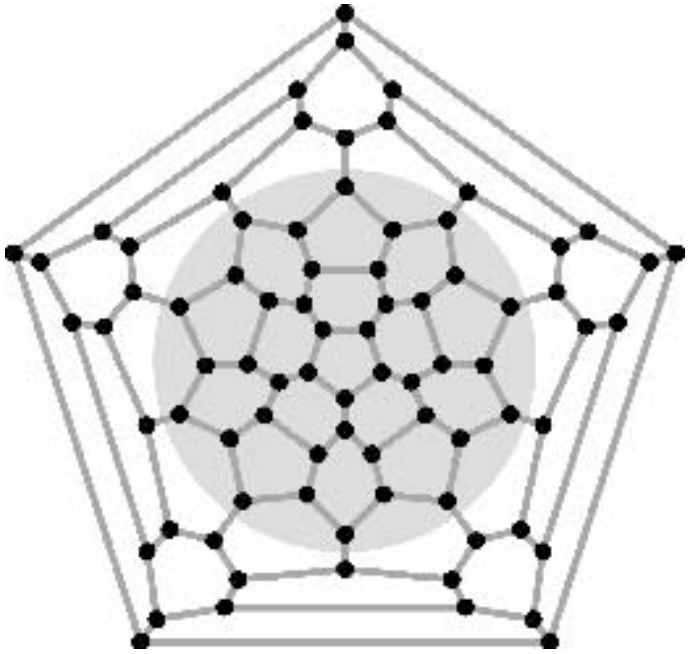


Expander (2)

- (condensers, extractors, (super)concentrators…),
- Many applications:
  - Circuit complexity, randoms, communications, cryptography, data structures …
- Pure mathematics: topology, group theory, measure theory, number theory, graph theory, …



Ramanujan graph



Computing the expanding constant

- Explicit: difficult but very interesting results
- Specific: complete graphs, cycles, …
- Cayley graphs
- Eigenvalue computations: the best ones today (related to random walks)

SL(2,p) (example)

- **Group of invertible matrices with determinant 1**
- Operations modulo *p*

$$A = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$$

A, *B* generate a group, described by a Cayley graph

Hash function based on SL(2,*p*) (Zemor-Tillich: 1994)

- **Group of invertible matrices with determinant 1**
- Operations modulo *p*

$$A = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$$

$C^i = A$ or B if $i = 0$ or 1

$M = m_0 m_1 m_2 \ldots m_{l-2} m_{l-1}$

$C^{m_0} C^{m_1} C^{m_2} \cdots C^{m_{l-2}} C^{m_{l-1}} = h(M)$

Security

- NB: associativity (no other hash functions like this)
- (Close to cascades (Delsarte-Q., 1974))
- Length of $h(M)$ constant (independent of M)
- NP-complete problem (representation of an element of a group given a set of generators)
- Cayley graph
- p with 500 bits or more (large graphs!)

- **Girth (collision):** partial proof of security *(maille, tour de taille): other examples: VSH (Contini, A. Lenstra, Stenfield, based on factorization) and also expanders (Charles, Goren and Lauter, 2005)*
- Diameter of the graph (not too large): and if expander good random walks
- Not very efficient (one bit as exponent)
- Construction similar to expanders based on Cayley graphs (Ramanujan graphs, …)

Secure camcoder: Joye-Q.:JCS, 1997

- Integrity of forms (precomputation)
- Signatures of sequences
- Postprocessing of videos (authenticity)
- Efficiency (?) and easy use thanks to associativity

Better? (1)

- $C^i = A$ or B if $i = 0$ or 1
- Binary -> *n*-ary
- Cayley graphs generated by *n* generators
- (associativity)
- Other groups: well known?
- Symmetric groups, alternating groups (all or all even permutations on *m* elements)?

Better? (2)

- Generating symmetric groups S_m
- Easy?: « any » 2 random elements from S_m generate S_m with high probability (Dixon, 1969)
- Choice related to girth and diameter (security)
- BUT not so many results

Research (Q., 2004)

- Choice n generators generating S_m, A_m, with good diameter and girth
- Expanders?
- Done: family of n generators (n in 2 …log(m) ... m) with good diameter, with good algorithm of representation (oops!)
- Girth?
- Subgroups?

Group generations

- Few results if optimal in complexity
- Golunkov (1971)
- Babai, Kantor (1988)
- JJQ (1983, 1987, 2004, …)

Group generations (applications)

- Gluskov automaton (1965):
- Cellular logic (Elspas, Stone, 1968 …):
- Cryptographic algos (modelling by finite lossless automata): Huffman (1955), Kurmit (1974),
- Analog scrambling: Wyner (1975), Sloane (1982),
- Random access memory: Stone, Aho-Ullmann,
- Access to databases (Klugge, 1977)
- Bubble memories (aso),
- Random generators (Luby-Rackoff, 1986),
- Criteria for Feistel schemes (Even-Goldreich, 1981),
- (Bill Gates) …

Symmetric groups S_n

- Pick at random an element (permutation): cycle structure? Goncharov(1942-1944), Knuth-Pardo (1976),
- Pick at random 2 or more elements: proba of generating S: Netto (1892), Dixon (1969),
- Diameter, girth?
- Cayley graphs

Results

- Easy and flexible constructions of sets of generators for S_m and A_m
- Very diameter (close to lower bound)
- 2, 3, …, log(log(m)), … log(m), … generators (except few cases)
- Hamiltonian paths
- Bipartite graphs
- (too) easy computation of paths from 0

Coming back to expanders

- Such graphs are « good » expanders
- Need of a seed graph in an iterative context (Wigderson et al, 2004)
- …

Conclusions

- Hash functions?
- Interesting results anyway
- Thanks for your interest
- Questions, answers, comments?

Protection de la vie privée sur Internet

Yves Deswarte

Directeur de Recherche au LAAS-CNRS
France

Protection de la vie privée sur Internet

Yves Deswarte
deswarte@laas.fr

LAAS-CNRS, Toulouse

Sécurité & protection de la vie privée

- "Privacy" ≈ confidentialité de données (et méta-données) personnelles
  PII : Personally Identifiable Information
- = sous-ensemble de "sécurité" (CIA)
- Mais…

… "*the devil is in the details*"

- Garder les justificatifs, en cas de litige
- Traçabilité des actions
- Authentification forte
- … danger pour la vie privée !!!

Sommaire

- "Privacy" : Définition, Législation
- Principes de base
- PETs : Privacy Enhancing Technologies
  - Gestion d'identités multiples
  - Protéger les adresses IP
  - Accès anonyme à des services
  - Autorisation respectant la vie privée
  - Gestion des données personnelles
- Projet Prime

"Privacy" : définitions

- Intimité, protection de la vie privée, du domaine privé
- Critères Communs (ISO 15408) :
  une classe fonctionnelle, 4 propriétés :
  - Anonymat : impossibilité (pour d'autres utilisateurs) de déterminer le véritable nom de l'utilisateur associé à un sujet, une opération, un objet
  - "Pseudonymat" : idem, sauf que l'utilisateur peut être tenu responsable de ses actes
  - Non-"chaînabilité" : impossibilité (pour d'autres utilisateurs) d'établir un lien entre différentes opérations faites par un même utilisateur
  - Non-observabilité : impossibilité (pour d'autres utilisateurs) de déterminer si une opération est en cours

Législation

- Guides pour l'utilisation de données personnelles informatisées et leurs transmissions internationales : OCDE en septembre 1980, Assemblée Générale de l'ONU, en décembre 1990.
- Protection des données à caractère personnel : Convention 108 du Conseil de l'Europe (26/01/81), directives 95/46/EC (libre mouvement) et 2002/58/CE (communications électroniques) (remplaçant la directive 97/66/CE)
- Protection des données nominatives -> à caractère personnel : loi "Informatique et Libertés" du 06/01/78, révisée par loi du 6 août 2004 + loi 94-548 (recherche médicale) http://www.cnil.fr/
- Secret professionnel (N^eau Code Pénal, art. 226-13) et secret des correspondances (NCP art. 226-15) + code des postes et télécommunications (secret des correspondances + art. L-32-3-1, inséré par la "Loi relative à la sécurité quotidienne" du 15/11/2001, révisé par la "Loi pour la sécurité intérieure" du 18/03/2003, la "Loi sur l'économie numérique" du 21/06/2004, puis la "Loi relative aux communications électroniques et aux services de communication audiovisuelle" n°2004-669 du 9 juillet 2004, décret du 24/03/06.

1er Principe pour protéger la vie privée :

- "Besoin d'en connaître" ("need-to-know")
  ne transmettre une information qu'à ceux qui en ont besoin pour réaliser la tâche qu'on leur confie
  -> Minimisation des données personnelles
  puis destruction/oubli

- … sur Internet comme dans le monde réel

- …avec des limites : certaines informations personnelles doivent pouvoir être fournies aux autorités judiciaires en cas de litige ou d'enquête (lutte contre le blanchiment d'argent sale, par exemple) : "pseudonymat" plutôt qu'anonymat total

Exemple : commerce électronique (1)

- Parties impliquées :
  un client, un marchand, un service de livraison, des banques, un émetteur de carte de crédit, un fournisseur d'accès Internet, …

- Le marchand n'a pas besoin (en général) de l'identité du client, mais doit être sûr de la validité du moyen de paiement.

- La société de livraison n'a pas besoin de connaître l'identité de l'acheteur, ni ce qui a été acheté (sauf les caractéristiques physiques), mais doit connaître l'identité et l'adresse du destinataire.

Exemple : commerce électronique (2)

- La banque du client ne doit pas connaître le marchand ni ce qui est acheté, seulement la référence du compte à créditer, le montant …
- La banque du marchand ne doit pas connaître le client…
- Le f.a.i. ne doit rien connaître de la transaction, sinon les caractéristiques techniques de la connexion …

2ème Principe pour protéger la vie privée :

- "Auto-détermination" : garder le contrôle sur ses [méta-] données personnelles

  -> stockage sur un dispositif personnel (carte à puce, PDA, PC…)

  -> si ces données sont divulguées à un tiers, imposer des **obligations** sur leur usage

  - Date de péremption
  - Notification en cas de transfert ou d'usage non prévu
  - etc…

PET : Privacy-Enhancing Technology

- Gestion d'identités multiples
- Protéger les adresses IP
- Accès anonyme à des services
- Autorisation respectant la vie privée
- Gestion des données personnelles

1° PET : gestion d'identités multiples

- Réduire les liens entre une personne et les données la concernant (contrôler la *chaînabilité*)
  - Communications et accès anonymes
- Mais : accès personnalisés / privilégiés : pseudonymes
  - Préférences (ex: météo)
  - "Rôles" différents -> pseudonymes différents
    - Ex: contribuable et électeur
  - Authentification adaptée au risque d'usurpation d'identité (et à la responsabilité)
  - Durée de vie liée aux besoins de chaînabilité -> pseudonymes "jetables"
- Identités virtuelles multiples vs. "single-sign-on"
  Liberty Alliance <http://www.projectliberty.org>
  vs. Microsoft Passport

Adresse IP= "donnée nominative"

Exemple :

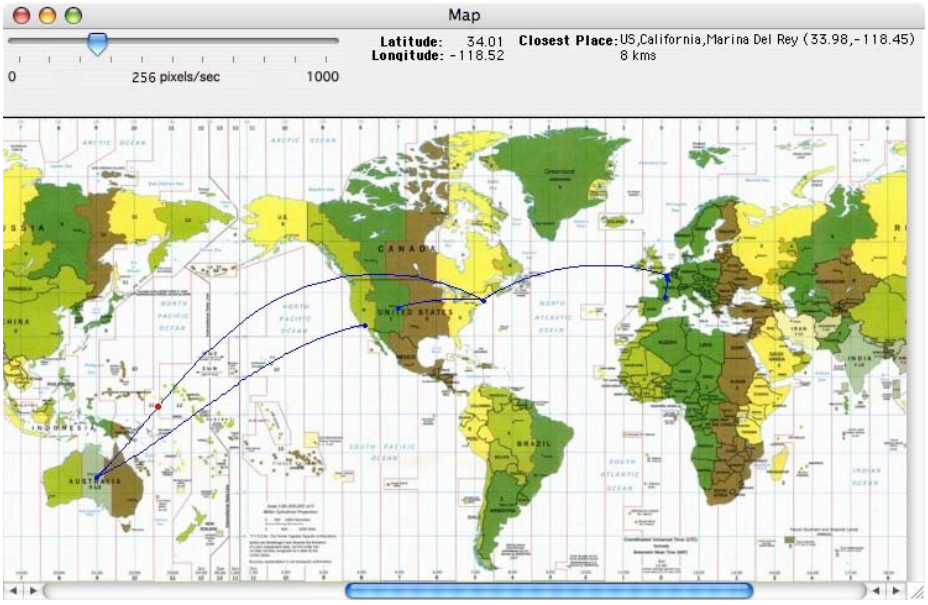
```
Return-Path: <Yves.Deswarte@laas.fr>
Received: from laas.laas.fr (140.93.0.15) by mail.libertysurf.net
(6.5.026)
      id 3D518DEF00116A4D for yves.deswarte@libertysurf.fr; Tue, 13 Aug
2002 13:44:40 +0200
Received: from [140.93.21.6] (tsfyd [140.93.21.6])
    by laas.laas.fr (8.12.5/8.12.5) with ESMTP id g7DBid1D001531
    for <yves.deswarte@libertysurf.fr>; Tue, 13 Aug 2002 13:44:39 +0200
(CEST)
User-Agent: Microsoft-Entourage/10.1.0.2006
Date: Tue, 13 Aug 2002 13:44:38 +0200
Subject: test
From: Yves Deswarte <Yves.Deswarte@laas.fr>
To: <yves.deswarte@libertysurf.fr>
Message-ID: <B97EBDC6.2052%Yves.Deswarte@laas.fr>
Mime-version: 1.0
Content-type: text/plain; charset="US-ASCII"
Content-transfer-encoding: 7bit
```

Adresse IP= "info sensible"

Exemple :

Adresse IP= localisation

Exemple :



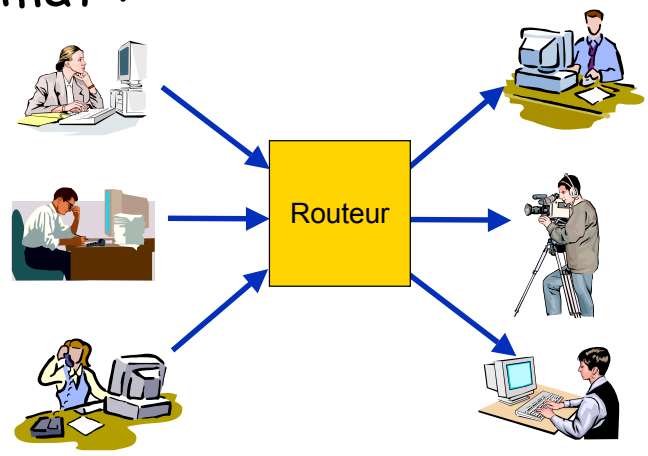
IP V6, réseaux ad hoc, …

- Demain : IP partout (*pervasive/ubiquitous computing, intelligence ambiante, sensor networks, RFID, convergence 4G …*)
- chaque "machin" aura une adresse IP *permanente* (nomade)
- chaque personne aura plusieurs machins …
- … qui se connecteront aux machins proches (réseaux ad hoc)
- … qui s'identifieront, routeront leurs communications, fourniront des infos contextuelles, etc.

2° PET : Protéger les adresses IP

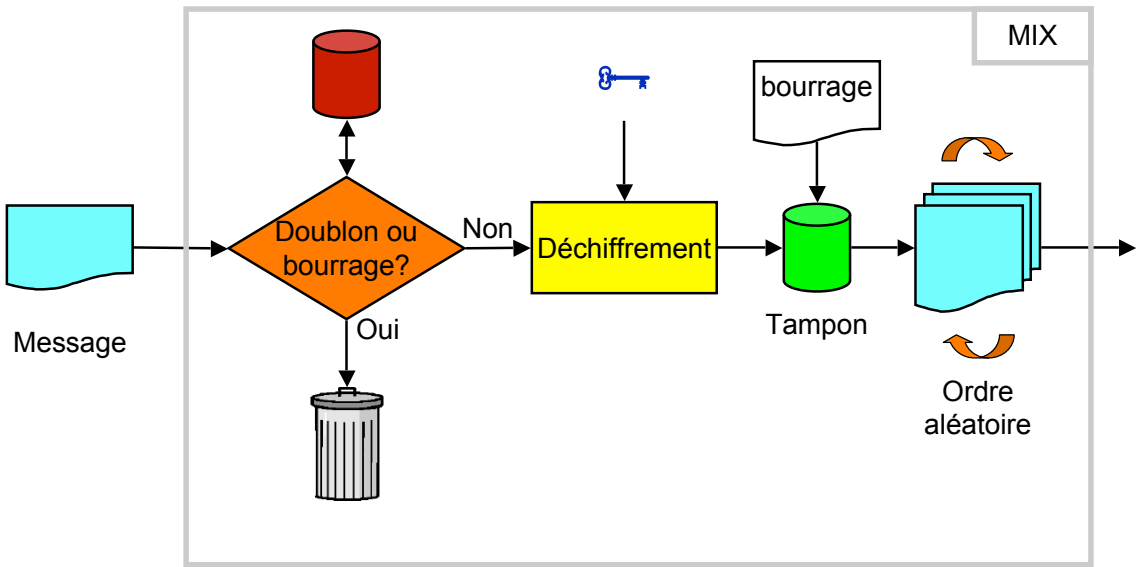
- PET : affectation dynamique des adresses IP (DHCP, PPP, NAT, …)

- Routeurs d'anonymat :
  - MIX
  - Onion Routing
  - Crowds

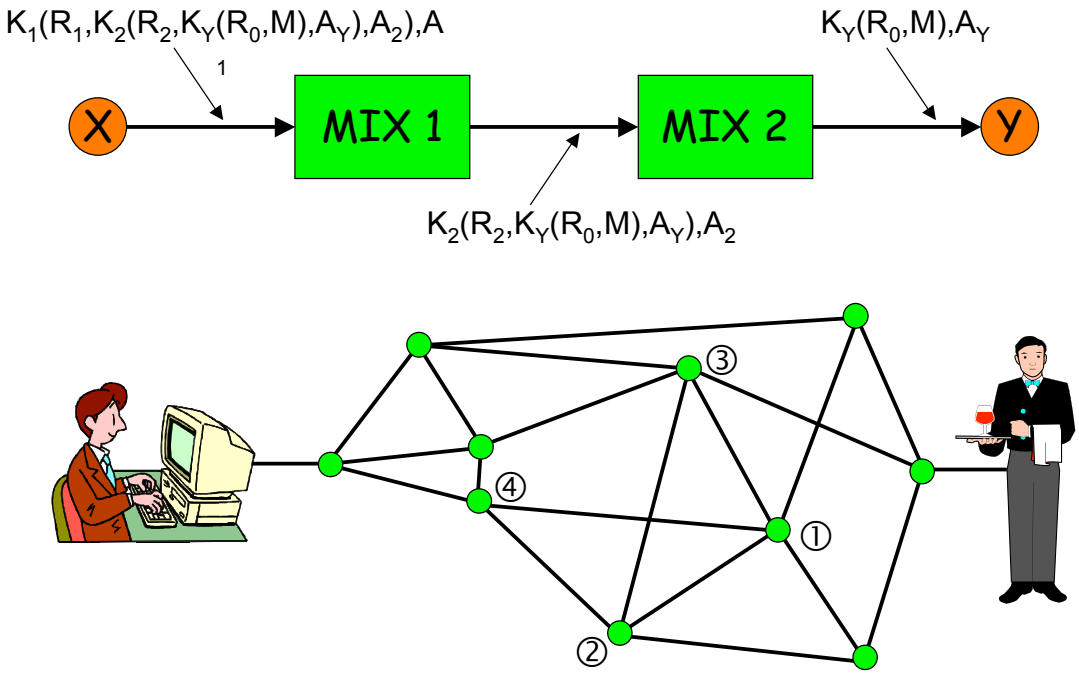


MIX : comment ça marche ?

http://www.inf.tu-dresden.de/

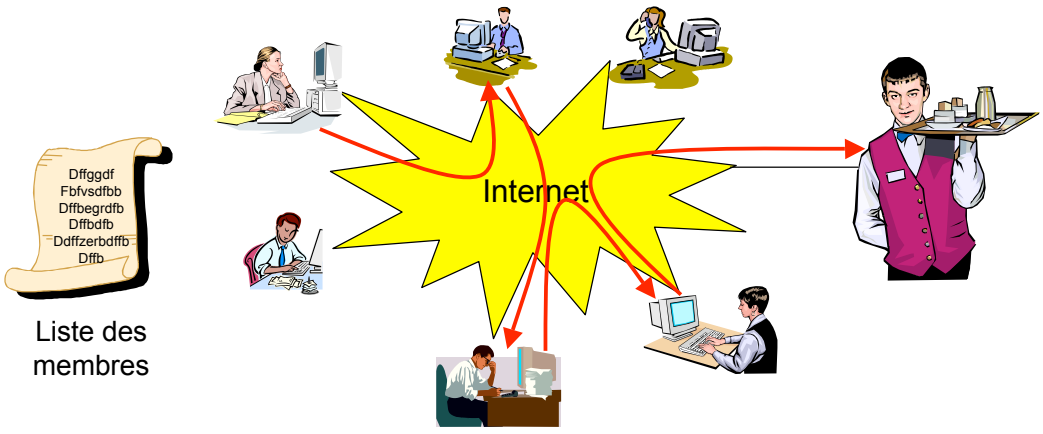


MIX / Onion Routing / Crowds



Crowds/Hords

- Chaque membre est un MIX pour les autres
- Probabilité *p* d'envoi au destinataire
  (1-*p*) d'envoi à un autre membre au hasard



Inconvénients des MIX

- Coût (# de messages, chiffrement, …)
- Vulnérables à la collusion entre les MIX
  --> indépendance entre les MIX ?
- Vulnérables à un observateur global (analyses statistiques)
  --> distribution sur Internet ?
- Interactivité : canal retour + anonymat de relation
- Mal adapté aux réseaux locaux…

Le dîner des cryptographes

- Comment savoir si quelqu'un a payé, sans pouvoir savoir qui ?

  DC-network

Protocole par tour : à chaque tour :

- Chacun diffuse un message ou du bourrage
- Chacun fait le XOR de tout ce qu'il a reçu
- Les bourrages sont générés de façon à s'annuler par XOR

  --> résultat = XOR(messages)
  - Si pas de message : résultat = 0
  - Si un seul message : tous les participants reçoivent le message (en clair)
  - Si plusieurs messages : collision --> résolution "aloha"

Bourrage s'auto-annulant

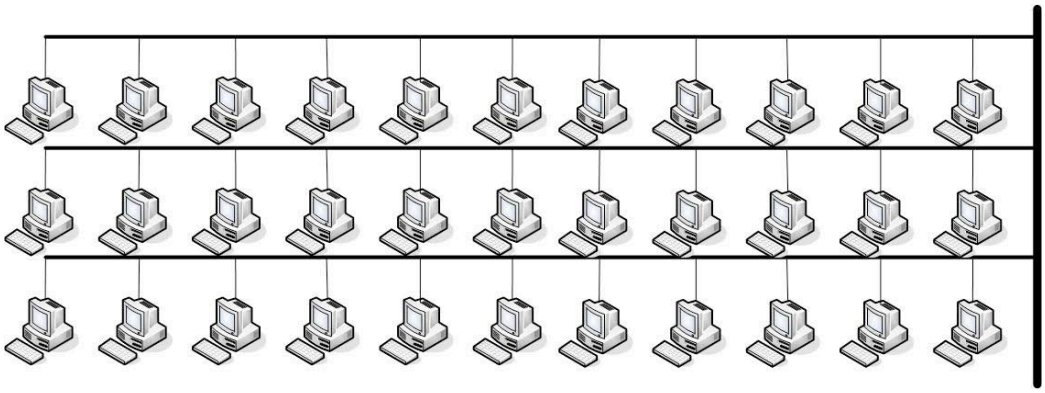
- $\forall$ i, j $\in$ {cryptographes}, i et j partagent une chaîne secrète de bits aléatoires de longueur infinie : $S_{i,j} = S_{i,j}$

- A chaque tour k :
  - Si i ne veut pas émettre de message, il diffuse $B_i = XOR_{i \neq j}$ (k-ième tranche $(S_{i,j})$)
  - Si i veut émettre le message M, il diffuse M XOR B_i
  - $XOR_{i=1..n}(B_i) = 0 \Rightarrow$ résultat = M (si un seul message)

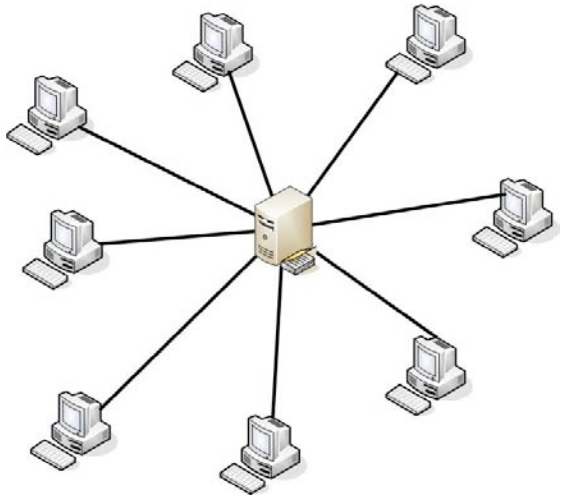
Problème 1 : coût du broadcast

- Tout le monde diffuse à tout le monde --> coûteux dans un réseau switché

- Traitement d'erreur très coûteux



Amélioration 1 : serveur central

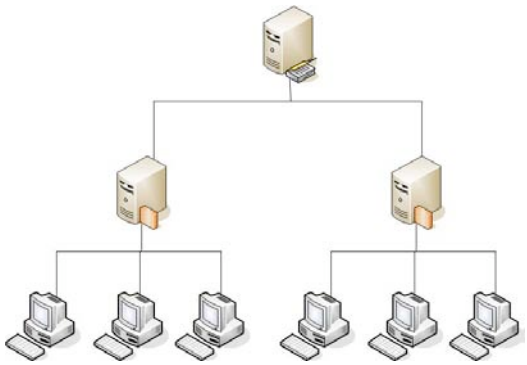


- Serveur de confiance (ex. sysadmin)
- Tous envoient leur message au serveur (point-à-point, traitement d'erreur simple)
- Le serveur fait le XOR et diffuse le résultat à tous (traitement d'erreur simple)

Problème 2 : passage à l'échelle

- Beaucoup d'abonnés : ils émettent à chaque tour (bourrage), même quand ils n'ont rien à dire --> bande passante consommée par abonné
- Serveur : puissance de calcul

Amélioration 2 : proxies



- Hiérarchie de proxies, qui font le XOR par segment (et traitement d'erreur)
- Le serveur fait un XOR réduit, et diffuse le résultat
- Tolérer les défaillances des proxies

Problème 3 : Bande passante

- Le débit est proportionnel au nombre d'abonnés, pas au nombre de communications.
- Le débit est consommé par chacun des abonnés, même s'il ne communique pas.

Amélioration 3 : débit adaptatif

- Tant qu'il n'y a pas de communication, les rounds sont espacés

- Dès qu'un message est détecté : doublement du rythme

- Si collision, doublement encore

- # slots max = # comms simultanées

Débit Max DCnet

- Nombre de XOR par round proportionnel à **n**
  → Débit % **~1/n**
- Video conf seulement pour petits groupes
  (débit-latence)
- Videostreaming (?) ou transferts de gros fichiers
  limités à 8 users max (débit)
- Audio possible pour des centaines d'utilisateurs
  ex: VoIP (débit-latence)

3° PET: Accès anonyme à des services

- Relais d'anonymat (*anonymity proxy*) : unidirectionnels (ou bidirectionnels?)
  - e-mail, news (Usenet)
    - anon.penet.fi (700 000 utilisateurs en 1996 !)
    - Cypherpunks
  - ftp
  - Web : ex: proxify.com
  - ...
- Serveur de pseudonymes :
  - e-mail
  - Identités multiples fournies par des f.a.i. (adresses mél)

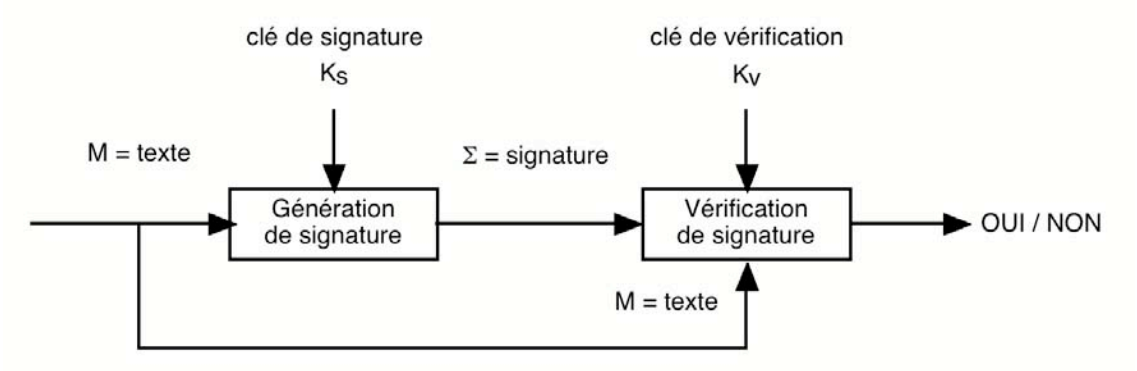
4° PET: Autorisation sur Internet

- Aujourd'hui : client-serveur
  le serveur accorde ou refuse des privilèges au client en fonction de son identité déclarée (éventuellement vérifiée par des mécanismes d'authentification)
- Le serveur doit enregistrer des données personnelles : preuves en cas de litige
- Ces données peuvent être utilisées à d'autres fins (profilage des clients, marketing direct, revente de fichiers clients, chantage...)
- Action P3P (W3C) : *Platform for Privacy Preferences Project*
  vérification automatique de politiques de sécurité/privacy "déclarées"

Ce schéma est dépassé

- Les transactions sur Internet mettent en jeu généralement plus de deux parties (ex : commerce électronique)
- Ces parties ont des intérêts différents (voire opposés) : suspicion mutuelle
- Nocif pour la vie privée : opposé au "besoin d'en connaître"

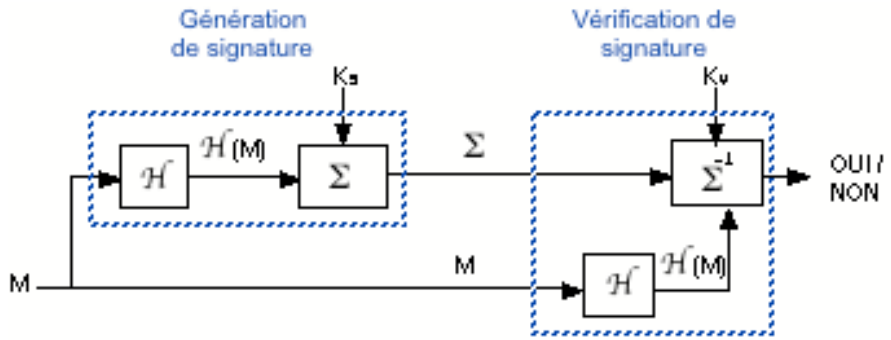
Rappel : signature numérique



- Ks = clé de signature Kv = clé de vérification
- Intégrité :
  - Sans connaître Ks, "impossible" de générer une signature valide
  - Il est "impossible" de trouver Ks, connaissant M et Σ (clair connu)
  - Il est "impossible" de trouver Ks, en choisissant M (clair choisi)

Signatures à clé publique : Ks ≠ Kv

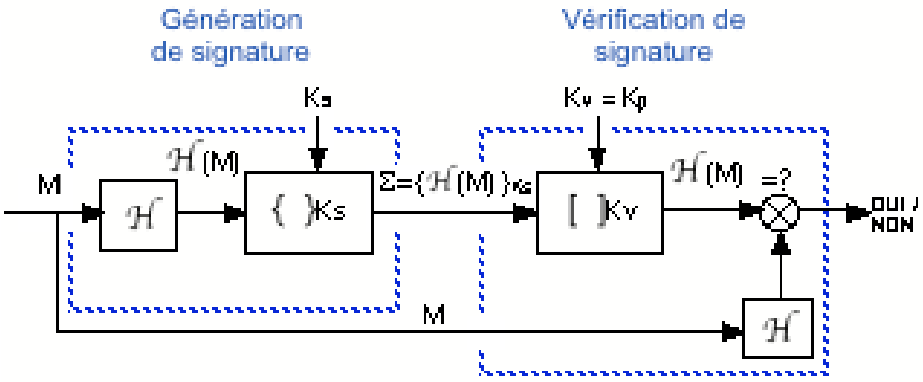
- Exemple : DSA



- Fonction de hachage : SHA-1
- Signature/vérification : el Gamal

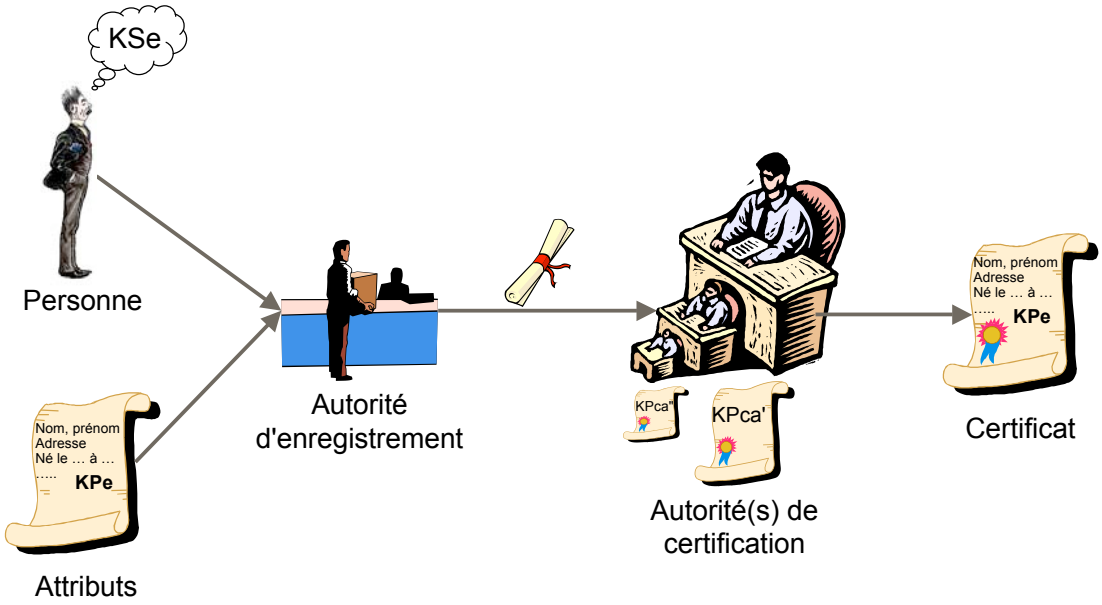
Signature par chiffres à clé publique

- Exemple : RSA



- Ks = clé de signature = clé de chiffrement Kc privée
- Kv = clé de vérification = clé de déchiffrement Kd publique

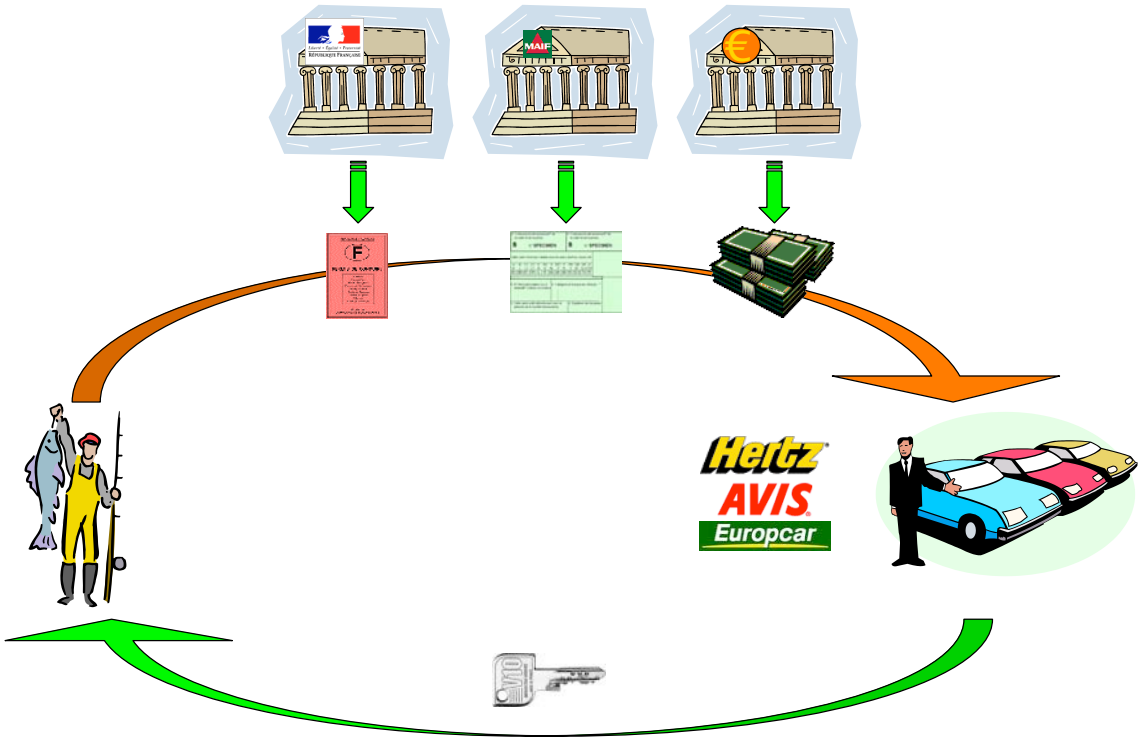
Certificats - IGC (PKI)



Preuves d'autorisation: credentials

- Certificats multiples :
  ex: SPKI : certificats d'attributs/d'autorisation
  - cartes d'abonnement, de membre d'association, ...
  - permis de conduire, carte d'électeur...
- Problèmes: "chaînabilité" (une seule clé publique pour plusieurs certificats?), gestion des certificats/clés, authentification, préservation des preuves, révocation, ...
- Certificats restreints :
  - "Partial Revelation of Certified Identity"
    Fabrice Boudot, CARDIS 2000

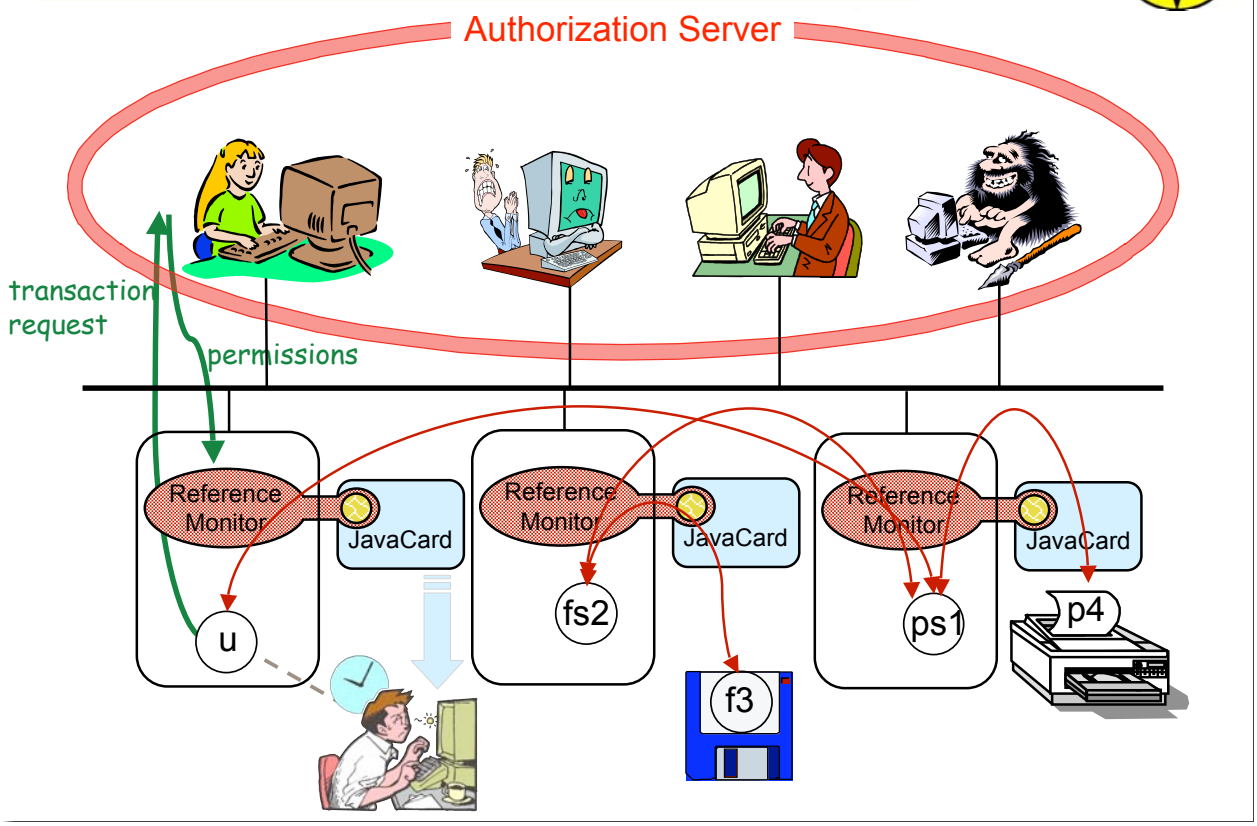
"Anonymous Credentials" (Idemix)



Signature de groupe

- Une clé publique de vérification de signature, *n* clefs privées de génération de signature.
- Le responsable de groupe distribue les clefs privées aux membres du groupe.
- Pour prouver qu'on est membre du groupe (= possède une garantie anonyme), on chiffre un message aléatoire, vérifiable, signé par le groupe.
- La vérification de la signature est une preuve d'appartenance, donc de garantie.
- Seul le responsable de groupe peut vérifier quel membre a signé.

Autorisation dans MAFTIA



e-Cash (1)

- Propriétés souhaitées :
  - Anonymat : un billet n'identifie pas la personne pour laquelle il a été émis
  - Impossibilité de fabriquer des faux
  - Impossibilité de dépenser deux fois
  - Transmissibilité : un billet peut être échangé entre personnes
  - Liquidité : un billet peut être divisé en petites coupures, ou agrégé en coupures supérieures

e-Cash (2) : signature aveugle (blind sign.)

- Alice génère un nombre aléatoire R, le multiplie par un facteur secret S, et l'envoie signé à sa banque: A->B: $[R*S, \text{valeur}]_A$
- La banque débite le compte d'Alice de la valeur, et renvoie le billet signé à Alice : B->A: $[R*S, \text{valeur}]_B$
- Alice "désaveugle" le billet $[R, \text{valeur}]_B$, et le dépense chez un marchand
- Le marchand transmet le billet à la banque : M->B: $[R, \text{valeur}]_B$
- La banque vérifie la signature, enregistre le billet comme dépensé, et crédite le compte du marchand de la valeur, et notifie le marchand, qui donne un reçu à Alice
- Si Alice (ou le marchand) essaye de redépenser le billet, la banque trouvera le billet dans la liste des billets dépensés

5° PET : gestion des données personnelles

- Négociation entre l'individu et l'entreprise
  ex: coupons de réduction en échange d'une publicité ciblée
- Auto-détermination : celui qui fournit des informations sur lui-même doit pouvoir contraindre l'usage qui pourrait en être fait --> Obligations
  ex: à effacer dans 48 h.
- Minimisation des données personnelles
  -> répartition : séparation des pouvoirs, fragmentation des données
  -> anonymisation + appauvrissement
  ex: remplacer le code postal par l'identifiant de la région

  -> Private Information Retrieval (PIR)

Private Information Retrieval (PIR)

- Exemple : PIR "parfaitement" sûr
  - Base de données répliquée
  - Composée de N éléments de taille fixe
  - 2 Requêtes :
    - 1 chaîne S de N bits aléatoires -> serveur 1
    - même chaîne sauf le k-ième bit inversé -> serveur 2
  - Réponse de chaque serveur = XOR de tous les éléments i tels que $S_i = 1$
  - Réponse = XOR des deux réponses
- Avec des méthodes cryptographiques (chiffrements homomorphiques {a + b} = {a}+{b}, résidus quadratiques et non-quadratiques, …), on peut réaliser des PIR "computationnellement" sûrs sans réplication

5°-bis PET : Accès aux données

- Principe du moindre privilège : un individu ne doit avoir que les droits minimaux nécessaires à sa tâche
- Politique de sécurité et mécanismes de protection : le détenteur d'une information en est responsable (art 34 de la loi « informatique et libertés »)
- Ces données peuvent être très critiques : ex: dossiers médicaux
  - Disponibilité : temps de réponse (urgence), pérennité
  - Intégrité : nécessaire à la confiance, éléments de preuve
  - Confidentialité : vie privée <-> intérêts économiques
- Privacy = contrôle d'accès + obligations

Contrôle d'accès aux données

- Séparation entre décision de contrôle d'accès et mise en œuvre
  - Décision : à un niveau élevé (ex. transaction)
    - Cohérence de l'ensemble des opérations
    - Décision sur la « sémantique » de la transaction
    - Moindre privilège : le privilège d'exécuter la transaction est inférieur à celui d'exécuter les opérations élémentaires

    Si OK --> génération de preuves d'autorisation
  - Mise en œuvre : à chaque opération élémentaire : fournir ou bloquer l'accès en fonction de l'opération et de ses paramètres vs. les preuves d'autorisation

Exemple : virement bancaire

- Transaction : virer 2000 € du compte 184-948449 au compte 946448-658
  - Lire le solde du compte 184-948449
  - Tester si le solde est supérieur à 2000 €
  - Si oui :
    - solde := solde - 2000; écrire solde 184-948449
    - Lire le solde du compte 946448-658
    - solde := solde + 2000; écrire solde 946448-658
  - Si non : retourner « solde insuffisant ».

Donner confiance aux utilisateurs…

… que leur vie privée est protégée?

- Certification & labellisation
- Approche Trusted Computing Group (TCG)
  - Support matériel : TPM
  - Bootstrap sûr
  - Vérification sceau S/W avant chargement
  - Vérifiable à distance, sans dévoiler d'identité (DAA)

PRIME (03/2004 - 02/2008)

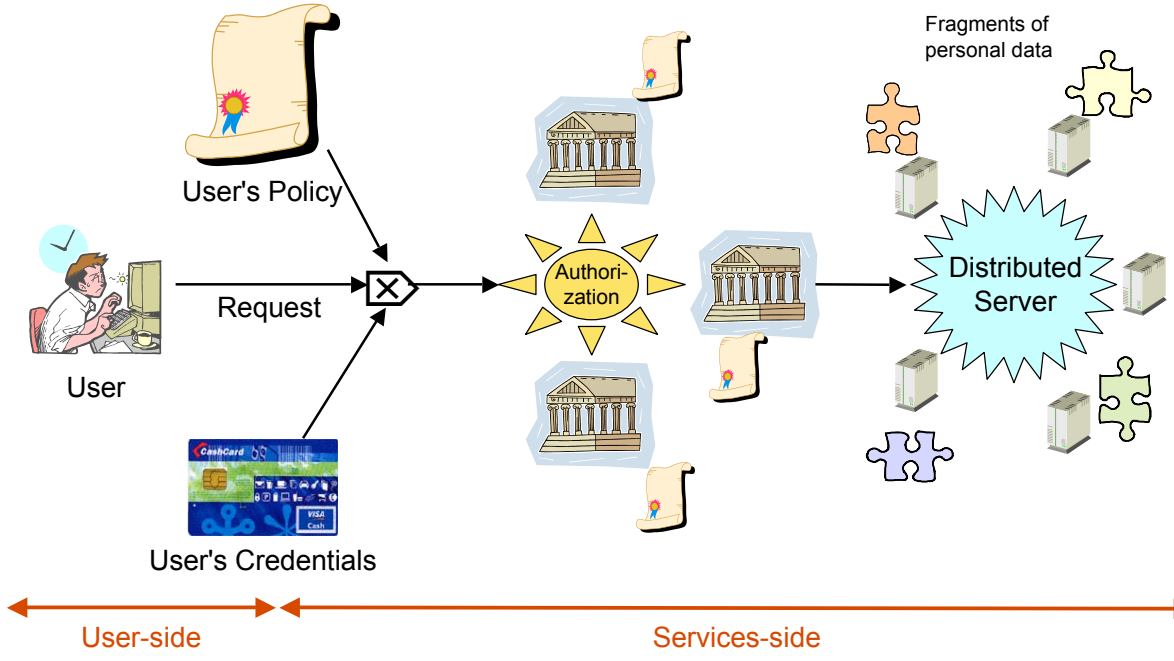
http://www.prime-project.eu.org/

- Privacy and Identity Management for Europe
  - Aspects juridico-socio-économiques
  - PET Côté utilisateur (dévelopt, utilisabilité)
  - PET Côté système, réseau, serveur
  - Applications réelles
- 20 Partenaires, 16 M€, subvention : ~10 M€
  - Fournisseurs (IBM, HP, …)
  - Labos (KUL, U. Dresde, U. Milan, Eurécom, LAAS…)
  - Utilisateurs (Lufthansa, T-Mobile, Swisscom, HSR)

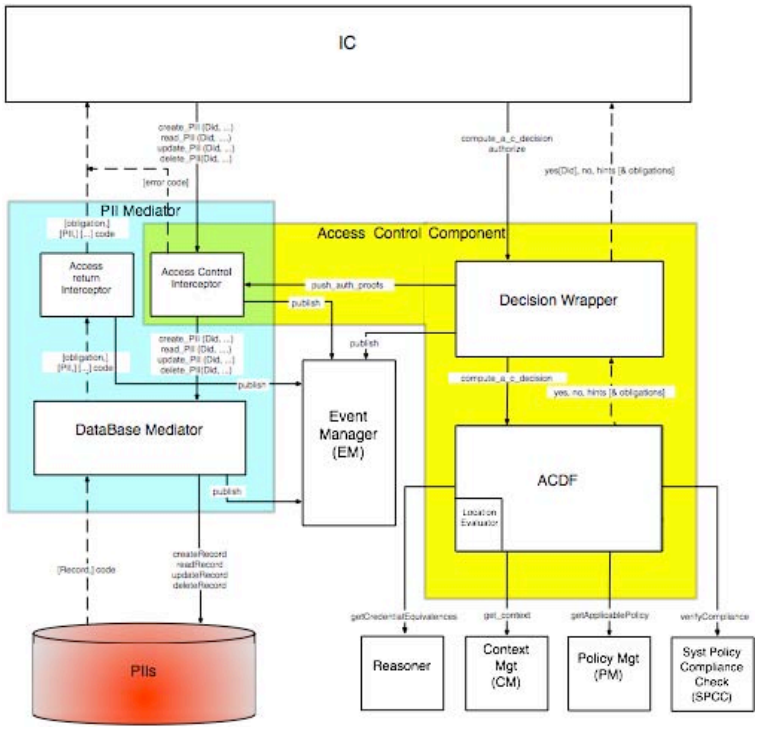
Principe :

- Identités différentes selon les besoins

Exemple d'architecture



Architecture du contrôle d'accès



Bibliographie

- *Sécurité des systèmes d'information V.2*, dir. Ludovic Mé & Yves Deswarte, Traité IC2, série Réseaux et télécommunications, Hermès, ISBN 2-7462-1259-5, 390 pp., à paraître en juin 2006. <http://www.lavoisier.fr/fr/livres/index.asp?texte=2746212590&select=isbn&from=Hermes>
- Simone Fischer-Hübner, *IT-Security & Privacy*, LNCS 1958, Springer, 2001.
- Stefan A. Brands, *Rethinking Public Key Infrastructures and Digital Certificates*, MIT Press, 2000.
- Yves Deswarte, Carlos Aguilar-Melchor, "Current and Future Privacy Enhancing Technologies for the Internet", à paraître dans *Annales des Télécommunications*, 2006.
- Yves Deswarte, Carlos Aguilar-Melchor, Vincent Nicomette, Matthieu Roy, "Protection de la vie privée sur Internet", à paraître dans *Revue de l'Électricité et de l'Électronique (REE)*, 2006.

Analyse de protocoles dans un modèle abstrait

Yannick Chevalier

Maître de Conférences à l'IRIT
France

Analyses de Protocoles Cryptographiques

École de printemps Cryptographie et Sécurité Informatique

Sorrèze 2006

Yannick Chevalier ⟨ychevali@irit.fr⟩

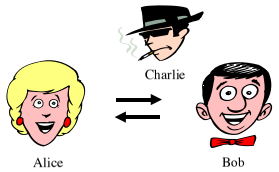
Outline

Outline

Motivation

The world is distributed. . .

- Our basic infrastructures are increasingly based on networked information systems
- Business, finance, communication, transportation, energy distribution, entertainment, . . .



Alice → Bob@Bank: “Transfer 100€ to account X”

Bob@Bank → Alice: “Transfer carried out”

- How does Bob know that he is really speaking with Alice?
- How does Bob know Alice just said it?

Security Problems

Communication in Open Networks

- Alice and Bob exchange Messages
- An Intruder controls the communication channel

Goal: provide private real-life communication facilities

Security Objectives for: A sends message M to B

- Confidentiality (only A and B know M)
- Integrity of data (M is not altered)
- Authenticity (B knows that A has sent M)

Solution: Security Protocols

Problem solved !

- SSL: browsers
- PGP: mail
- SET: E-commerce
- Kerberos:remote login ...

. . . , is it ?

New protocols for:

- Mobile environment
- Dedicated tasks (e-Voting)
- Privacy enhancing

Building Blocks for Security Protocols

Cryptographic Procedures: Encryption or signature of messages

public keys $\{M\}_{K_B}, K_B^{-1} \Rightarrow M$

secret keys $\{M\}_K, K \Rightarrow M$

(Pseudo-)Random Number Generators: to generate “Nonces”, e.g. for “Challenge-Response”

Protocols: recipe for exchanging messages

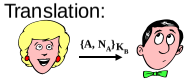
Steps like: A sends B his name together with the message M. The pair $\{A, M\}$ is encrypted with B's key.

$$A \rightarrow B : \{A, M\}_{K_B}$$

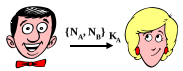
An Authentication Protocol: Needham-Schroeder

1. $A \rightarrow B : \{A, N_A\}_{K_B}$
2. $B \rightarrow A : \{N_A, N_B\}_{K_A}$
3. $A \rightarrow B : \{N_B\}_{K_B}$

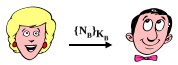
Translation:



“I am Alice and here is a Nonce N_A.”



“Here is your Nonce N_A and I also have one for you.”



“I got it! It is N_B.”

Protocols are typically small and convincing ...

... but also often wrong!

Attack on Needham-Schroeder

$\{A, N_A\}_{K_C}$ $\{A, N_A\}_{K_B}$

$\{N_A, N_B\}_{K_A}$ $\{N_A, N_B\}_{K_A}$

$\{N_B\}_{K_C}$ $\{N_B\}_{K_B}$

B believes he is speaking with A!

What went wrong?

- Problem in step 2: $B \rightarrow A : \{N_A, N_B\}_{K_A}$
  Agent B should also give his name: $\{N_A, N_B, B\}_{K_A}$
- Is this new version correct now?

. . . against this or other kinds of attacks?

Use formal methods (and automated tools)!

- Other moral: There are vulnerabilities beyond cryptography

What are Cryptographic Protocols ?

Protocols are Distributed programs communicating along a fixed message sequence used by agents to provide a security layer for communications over an insecure network

- Distributed programs: parts (Role)evolve independently one from another
- Fixed message sequence: At each step, a role waits for a message satisfying a given pattern and will deterministically compute a response from its execution so far
- The roles are instantiated when executed by an agent
- Security layer: The message sequence must satisfies some properties provided that the network satisfies some assumptions
- No communications between instances through another medium
- A session is defined by a network model and a set of role instances

Common assumption: Dolev-Yao model of an intruder that controls the network

Outline

Messages in Protocols

Messages as strings

In implementations:

- Messages are strings of bit or characters
- Alternatively, they can be treated as numbers

Dolev-Yao model

Abstraction on messages:

- Consider only the relations between valid messages
- Define these relations by cryptographic functions

Defaults of the model

Abstraction from particular values

- Validity of messages is often an issue for cryptographic analysis
- One cannot express that x and $x + 1$ are close one to another

Simple relational model

- Only operations in a given set are possible
- Operations either certainly succeed or certainly fail
- All operations have the same Cost

Messages (1)

Constants

- Unrelated messages are modelled by different constants in an infinite set
- Messages with no properties are modelled by free constants
- Alternatively, one may give to constants some specific properties
- Size doesn't matter

Random values

Assumption that random values really exists:

- Independently of previous and future generated values
- Unrelated to other constants such as agent's name or public key

Modelled either by free constants otherwise unused or relying on skolemization

Messages (2)

Operations – Equational Model (1)

- An operation is modelled by an unique symbol with non-null arity

  Example: Symmetric encryption
  - ⋆ symbol: $\{\_\}^s_\_$
  - ⋆ two arguments: a key and a message

- Messages are constructed with constants and these symbols

Fundamental Example

Operations in the original model

Several operations are usually considered:

- Symmetric encryption ($\{\_\}^s_\_$) and decryption ($\mathsf{Dec}_s(\_,\_)$)
- Asymmetric encryption ($\{\_\}^p_\_$) and decryption ($\mathsf{Dec}_p(\_,\_)$), m^{-1} denotes the inverse key of the key m.
- Message concatenation (pairs) $\langle \_,\_ \rangle$ and projections $\pi_1(\_)$,$\pi_2(\_)$ on components
- Application of hash functions $h(\_)$ for function h

Can be extended with other operations, or distinct operators for e.g. symmetric encryption,...

Messages (4)

Operations – Equational Model (2)

- Must provide relations between operations
- Done using equations between terms

$$\begin{array}{rcl} \mathsf{Dec}_s(\{m_1\}^s_{m_2}, m_2) & = & m_1 \\ \pi_1(\langle m_1, m_2 \rangle) & = & m_1 \end{array} \qquad \begin{array}{rcl} \mathsf{Dec}_p(\{m_1\}^p_{m_2}, {m_2}^{-1}) & = & m_1 \\ \pi_2(\langle m_1, m_2 \rangle) & = & m_2 \end{array}$$

Messages (5)

Operations – Equational Model (3)

- Other equalities may be added
- Equational theory fixed before the analysis

⇒ Main default:

There exists no other relations between messages

In preceding example:

- No ambiguity in the construction of a message

$$\{m_1\}^s_{m_2} \neq \langle m_3, m_4 \rangle$$

- Perfect cryptography assumption

Intruder and Network

In real-life

For communications between distant sites

- A malevolent person may tamper the wires
- The routers may have been attacked
- Someone in my local network may listen to all communications
- Listening even easier for Wireless networks

Control by an intruder

Worst-case assumption: An intruder can

- Intercept messages
- Fake its identity
- Listen to communications

Two first points imply the third one

Intruder's deductions

Dolev-Yao settings

Perfect Cryptography assumption

- Operations may either or not be used by the intruder
- Permitted operations: Public symbols ($\langle \_, \_ \rangle$,$\mathrm{Dec}_s(\_, \_)$)
- Forbidden operations: private symbols ($\_^{-1}$)

A bit too restrictive...

Slight extension

- Separate permitted operations from signature
- Permitted Operations are defined by terms over the signature
- There might be an infinite number of such terms
- Allow an arbitrary equational theory

Expressive power of the extension (1)

In theory...

- Very general setting, e.g. can model primitive recursive algorithms on bit-strings
- Give to the intruder terms describing the functions he may use

Example

- Signature $\{0, S(\_), \langle \_, \_ \rangle, \pi_1(\_), \pi_2(\_)\}$ plus other symbols defined by these
- Trivial encoding of natural integers $0, S(0), \ldots, S^n(0), \ldots$
- Definition of a new function h:
  - $h(0, x) = f(x)$
  - $h(S(x), y) = g(h(x, y), x, y)$
  - Encode arity with pairs and projections

Expressive power of the extension (2)

In practice...

- It's useless to express something you cannot analyze
- Give just enough details to express the desired result
- Perform analysis in this abstraction

Still in the spirit of Dolev and Yao

Notation

For an intruder $\mathcal{I}$ able to apply functions described by terms in a set S defined over a signature $\mathcal{F}$ with equational theory $\mathcal{E}$,

We define $\mathcal{I} = \langle \mathcal{F}, S, \mathcal{E} \rangle$

Example

The Exclusive OR

$$x, y \rightarrow x \oplus y \qquad x, y \rightarrow 0$$

and

$$\begin{aligned} (x \oplus y) \oplus z &= x \oplus (y \oplus z) \\ x \oplus y &= y \oplus x \\ x \oplus 0 &= x \\ x \oplus x &= 0 \end{aligned}$$

Deduction relation

- defined by the ground instances of deduction rules
- Notation: $\overline{E}$ denotes the infinite set of terms deducible from E
- Example ?

Example of a Sequence of Deductions

Derivations

$a, a \oplus b \rightarrow_{\mathcal{L}} a, a \oplus b, (a \oplus b) \oplus a =_{\mathcal{E}} a, a \oplus b, b$

- Initially, the intruder knows a and $a \oplus b$
- His knowledge doesn't decrease (infinite memory intruder)
- He uses the 'function' $x, y \mapsto x \oplus y$ with x instantiated by $a \oplus b$ and y instantiated by a
- The result is rewritten in a simpler notation, b

Temporary Conclusion

We have defined:

- An abstract intruder model
- This model is symbolic: The intruder operates on terms rather than on bit-strings
- Doesn't take into account flaws in the cryptographic layer
- Doesn't take into account cryptanalysis, statistical attacks,. . .

Ok if one looks for flaws, but insufficient for validation

Outline

Secrecy

A Basic Property ?

Informally. . .

- When one buys newspapers with a credit card, one doesn't want anyone else (including the vendor) to know the secret code
- This secret code is in a finite set of values, and can anyway be guessed
- But one only has three attempts for a correct guess

⇒ Simplification: the secrecy of the card code is preserved if it cannot be guessed at the first attempt

Secrecy

What does "guess" mean ?

- We rule out the lucky guess, so we look for a deterministic algorithm permitting to produce this secret code
- From the intruder model, this algorithm consists in:
  - ⋆ Some composition of functions available to the intruder
  - ⋆ Some interactions with vendor(s) and/or client(s)
- In other words, we're searching an execution of the protocol such that, at the end, the intruder is able to deduce the secret card code

⇒ Reachability property

Confidentiality

Example: e-Voting

- Assume a yes/no vote
- The problem here is not to produce the yes/no values, but to determine what a given person has voted
- Complex property in general (see Peter Ryan's presentation)
- A necessary condition is that an intruder should not be able to tell the difference between sequences of messages corresponding to yes or no votes

Confidentiality

Tell the difference ???

Assume t_y (resp. t_n) is a sequence of messages corresponding to a yes (resp. no) vote

- From the non-collision property, we can assume t_y and t_n contain different messages
- Thus, an intruder sees different sequences of messages
- The problem is whether he knows epistemically whether t_y and t_n contain the same vote or not

⇒ Equivalence property

Authentication

Let's go phishing...

Consider the following situation:

- You connect to the web site of your bank
- You login securely to this site using your account number and password
- You're assured that none other than you and the distant site know the value of your password

Secrecy is not sufficient

Authentication

- One wants to be certain of the identity of its correspondent
- Certainty means here that, after one point of the execution of the protocol, it is impossible to communicate with someone who is not the claimed correspondent

$\Rightarrow$ Again a reachability property

Non-repudiation

No plausible deny

- Assume you want to be sure that your correspondent has received a letter
- You can require that before possessing the letter your correspondent signs a reception acknowledgement
- The postal office guarantees that this acknowledgement is produced if and only if the letter was received
- You can keep this acknowledgement as an evidence for further processing

Non-repudiation

Non-repudiation

- Close, but not equal to authentication
- Non-repudiation states that an agent cannot deny having performed an action
- It is based on an evidence which is uniquely related to the person and the action
- For protocols, based on messages sent by agents, and thus needs assumptions on secrecy of some data
- Based on the existence of a trusted third party

Non-repudiation can be seen as mutual authentication and secrecy properties of an exchange

Outline

Outline

Framework for Reachability Analysis

Goal:

- Help developers to rule out flaws during the design of a protocol
- Automatic, fast and exact. . .

. . . in not raising false alarms

Framework

- Finite instances for correctness
- Fast tools already exist

Models for Protocols

Input specification

- Narrations describing the aimed messages exchange

$$\begin{array}{lll} 1. & A \to B: & \{A, N_A\}_{K_B} \\ 2. & B \to A: & \{N_A, N_B\}_{K_A} \\ 3. & A \to B: & \{N_B\}_{K_B} \end{array}$$

- One has to give an operational semantics to this narration Protocol compilation
- The operational semantics for the intruder actions is defined by his deduction rules
- Then one defines an execution scenario by a finite instance

Operational Semantics

A.k.a compilation

First attempt

Define step by step the actions of a role on the received message in order to construct the response

$$\begin{array}{lll} 1. & A \to B: & \{A, N_A\}^p_{K_B} \\ 2. & B \to A: & \{N_A, N_B\}^p_{K_A} \end{array}$$

- B decrypts the received message with her private key, gets the nonce N_A and the sender A
- B then constructs the response to A by creating a new nonce N_B and encrypting the two nonces with A's public key

Compilation, cont'd

To sum up...

- action of B: $x \Rightarrow \left\{\left\langle \pi_2(\mathsf{Dec}_p(x, {K_B}^{-1})), N_B\right\rangle\right\}^p_{K_{\pi_1(\mathsf{Dec}_p(x,{K_B}^{-1}))}}$
- Note that nonces require a special handling
- Depending on the setting, B may also check that:
  - ⋆ The received message matches the right pattern:

$$x = \{\langle v_1, v_2\rangle\}^p_{K_B}$$

  - ⋆ That she hasn't received the nonce N_A before (If she keeps a list of received nonces)
  - ⋆ ...

Notes on Compilation

Output Formalisms

- π-calculus: Explicit constructions
- Multiset-rewriting rules: simplified result

Simplification of $\begin{cases} x & \Rightarrow & \left\{\left\langle \pi_2(\mathsf{Dec}_p(x, {K_B}^{-1})), N_B\right\rangle\right\}^p_{K_{\pi_1(\mathsf{Dec}_p(x,{K_B}^{-1}))}} \\ x & = & \{\langle v_1, v_2\rangle\}^p_{K_B} \end{cases}$

1. Replace x by its pattern in the rule:
$$\{\langle v_1, v_2\rangle\}^p_{K_B} \Rightarrow \left\{\left\langle \pi_2(\mathsf{Dec}_p(\{\langle v_1, v_2\rangle\}^p_{K_B}, {K_B}^{-1})), N_B\right\rangle\right\}^p_{K_{\pi_1(\mathsf{Dec}_p(\{\langle v_1,v_2\rangle\}^p_{K_B},{K_B}^{-1}))}}$$

2. Simplify with equational theory:
$$\{\langle v_1, v_2\rangle\}^p_{K_B} \Rightarrow \{\langle v_2, N_B\rangle\}^p_{K_{v_1}}$$

Permits a simpler analysis with no equational theory!

Consequence of the Compilation

On a role

- A role is an ordered sequence of Receive ⇒ Send rules $\mathcal{R}_1, \ldots, \mathcal{R}_n$
- Under reasonable assumptions, one can prove that if a variable is in the RHS of rule R_i, it is in LHS of rule R_j for some $j \leq i$
- This property remains true when we gather rules from several roles

Cornerstone to every (known) proof of decidability results

Execution of a protocol

- One selects a rule that has smaller rule and remove it from the set of protocol rules
- The intruder constructs a message matching the LHS
- He receives the response

and iterate

Example of compilation

Session: Role A played by a, role B played by b, instantiated initial knowledge

1. $A \to B$: $\{A, N_A\}_{K_B}$
2. $B \to A$: $\{N_A, N_B\}_{K_A}$
3. $A \to B$: $\{N_B\}_{K_B}$

$(a,1)$ $\quad \emptyset \Rightarrow \{a, na\}kb$
$(b,1)$ $\quad \{a, x_{Na}\}kb \Rightarrow \{x_{Na}, nb\}ka$
$(a,2)$ $\quad \{na, x_{Nb}\}ka \Rightarrow \{x_{Nb}\}kb$
$(b,2)$ $\quad \{nb\}kb \Rightarrow \emptyset$

Outline

Simplified setting

Another simplification

- Assume the principals can check exactly the received messages
- Then variables can only be instantiated by constants

Permits to bound the "size" of messages

Results in this setting

- Session with a bounded number of protocol rules: Decidable
- Session with an unbounded number of protocol rules: Undecidable

Decidability

Reduction to problems without variables

- Finite number of constants once nonces are created
- First step: choose a mapping from variables to constants
- Instantiate the rules according to this mapping

Can be done quickly by model-checker

Well-formed derivations

- Assume the intruder knows the set of terms E and has to build message m
- One can prove that if m can be deduced from E, then it can be deduced using only subterms of E and t
- Bounds the length of a derivation

Undecidability

Sketch of the undecidability proof

- Existential Horn clauses:

$$\forall x_1, \ldots, x_n, P_1(x_1, \ldots, x_n) \wedge \ldots \wedge P_k(x_1, \ldots, x_n) \Rightarrow (\exists y_1, \ldots, y_l, Q(x_1, \ldots, x_n, y_1, \ldots, y_l$$

- Can be translated to role rules:

$$\{P_1, x_1, \ldots, x_n\}_k^s, \ldots, \{P_k, x_1, \ldots, x_n\}_k^s \Rightarrow (\exists y_1, \ldots, y_l, \{Q, x_1, \ldots, x_n, y_1, \ldots, y_l\}_k^s)$$

where k is not known by the intruder

- Existential quantifier corresponds to the creation of nonces
- Implication problem: Can an atomic fact be derived from a set of clau
- This problem is undecidable, and thus secrecy for an unbounded number of participants is undecidable

Analysis of the undecidability

Facts

- Implication problem for Horn clauses without existential is decidable
- Idea: remove the nonces
- Result: protocols with an infinite set of role rules, but a finite set of constants
- Add new clauses for intruder deductions

Result

- Secrecy problem can be reduced to DATALOG problem
- Decidable in DEXPTIME

But most protocols use nonces !

Abstraction techniques for Protocol validation

Nonces from Lutece

- Idea: re-use the nonces in different rules
- Try to be clever enough to not find secrecy problems for all protocols:
  - ⋆ By indexing a nonce by the sender and the receiver
  - ⋆ By showing that attacks on a protocol, if any, can be mounted with less than n nonces

Authentication

- In any case, authentication is almost ruled out by this abstraction
  - ⋆ $\Rightarrow$ Replace authentication by weak authentication, where one only care of the origin of a message (and not of its freshness)
  - ⋆ Or ensure that abstracted nonces are not relevant to authentication

Efficient model-checking

Symbolic or lazy approach

Three ingredients:

- Variables are ordered according to their apparition time
- Intruder knowledge increases along this order
- Variables are instantiated as late as possible

Advantages

- Very efficient method
- Can be combined with methods for validation
- Main remaining difficulty: Order of messages

Outline

Introduction

Needham-Schroeder Protocol with El-Gamal encryption

$A \rightarrow B : (\alpha_B{}^{x_1}\ ,\ \alpha_B{}^{k_B \cdot x_1} \oplus (A, N_a))$

$B \rightarrow A : (\alpha_B{}^{x_2}\ ,\ \alpha_A{}^{k_A \cdot x_2} \oplus (N_a, N_b))$

$A \rightarrow B : (\alpha_B{}^{x_3}\ ,\ \alpha_B{}^{k_B \cdot x_3} \oplus (N_b))$

- $\alpha_{A|B}$, $\alpha^k_{A|B}$ public
- $k_{A|B}$ secret
- x_1, x_2 and x_3 nonces
- multiplication operators $\cdot$

Need of four disjoint theories

- $\oplus$
- pairing
- abelian group
- exponentiation

A negative result – 1

Simulatability relation

- Goal: Prove that Dolev-Yao like abstraction is faithfull to underlying cryptographic primitives
- Technique: Define conditions on cryptographic operations such that cryptographic proofs go up to symbolic level
- Achievements: Conditions for asymmetric and symmetric encryption and signature
- Permits cryptographically sound proofs, still largely by hand, at the symbolic level of some protocols

Permits to provide more confidence w.r.t. validation

A negative result – 2

Drawbacks

- No good model for hash functions (at the moment)
- A proof that there exists no way to handle exotic operations at cryptographic level

Handling of algebraic operators is good to find more errors, but may be useless for validation

A positive result

Independance of operators

- Symmetric operations are defined independently of asymmetric operations
- Formally they are defined on disjoint signatures
- In this case, reachability problems are modular

Application

- Permits to simplify proofs by considering operators in isolation
- Drawback: requires that functions employed by the intruder are applicable on all terms:
  - ⋆ Good: $x, y \mapsto \mathsf{Dec}_s(x, y)$
  - ⋆ Bad: $\{x\}^s_y, y \mapsto y$

Example of operations

Sheer algebraic operations

- Generalization of decryption/encryption operators
- Exclusive or is the first and simplest non-trivial example
- Abelian groups

Algebraic + something else

- Exponentiation, which can be reduced to abelian groups
- adding homomorphism to Xor: The Xor of a pair is equal to the pair of the xor
- Negative result: homomorphism and associative-commutative theory is not decidable!

In practice

Theory and practice

- Addition of algebraic operators often doesn't change the complexity class of reachability problems
- In practice the combinatorics is much higher
- Doesn't mix well with Lazy intruder Strategies

Few attempts

- Lazy strategy can be extended to Xor operator (same technique seems to apply for exponential)
- Implementation of incomplete strategies in a few tools

Work in progress, but techniques slowly percole to the level of tools

Outline

Outline

Voting procedure in France

Example of a yes/no voting procedure

- Publicly, take the two possible choices and an envelope
- In a box, put one of the choices in the envelope
- Publicly, put the envelope in the urn

Why does it work ?

- The first step ensures that an observer in the voting office will not be able to guess your vote from your selection
- The second step ensures that no one see you put something in the envelope
- We assume that an envelope permits to keep its content secret
- After being put in the jar, we assume that the envelope doesn't contain anything that could identify you

Is it correct ?

One could physically. . .

- Mark the envelopes
- have translucent envelopes
- Put a video camera in the box
- . . .

What does confidentiality mean ?

- In the protocol setting, we again model messages by terms
- We have to define for the intruder, again, a set of possible actions and a set of terms at his disposal
- Goal: Check that, given a set of actions and terms, an intruder cannot distinguish between a voter voting yes and a voter voting no.

Consequence: The goal of the intruder is no more to have specific terms at his disposal

Observational equivalence

Gaining information

- In the symbolic setting, $\langle a, b\rangle$ and a have nothing in common
- $\{a\}_k^s$ and a also have nothing in common
- But if k is not at intruder's disposal, the intruder knows that a is a part of $\langle a, b\rangle$, but doesn't know that a is a part of $\{a\}_k^s$
- Difference: In the first case, he can apply $\pi_1(\_)$ on $\langle a, b\rangle$ and see that he gets a

Conclusion: Someone learns something when, after applying some functions to the messages, one finds an equality

This permits to give a formal basis to define what someone knows

Definition of Knowledge

Frame

- A Frame is an ordered sequence of messages
- It models:
  - ⋆ Terms at the disposal of the intruder
  - ⋆ The trace of an execution of a protocol
- Indistinguishability of two frames is modelled by logical equivalence of the two frames

We need to define this logic

A logic for the observer

- A propositional logic with usual connectors
- An equality predicate $\_ = \_$
- Arguments of this predicate: Terms build from terms in the frame by applying deduction rules

Two frames φ_1 and φ_2 are equivalent iff for any formula A, $\varphi_1 \models A$ iff $\varphi_2 \models A$

Simplification...

Evaluation on atoms

- Given the quantification on all formulas, one is easily convinced that it suffices to check equivalence on atoms
- We formalize deductions performed by contexts:
  - ⋆ Write a frame $\varphi = \lceil x_1 = t_1, \ldots, x_n = t_n \rceil$
  - ⋆ A context M is a term build from variables $x_1, \ldots, x_n$ by application of deduction rules:

$$x_1, \ldots, x_n \rightarrow x_1, \ldots, x_n, \mathsf{Dec}_s(x_1, x_3) \rightarrow x_1, \ldots, x_n, \mathsf{Dec}_s(x_1, x_3), \langle x_2, \mathsf{Dec}_s(x_1, x_3)\rangle$$

  - ⋆ We can write atomic formula $M = N$ where M and N are contexts
  - ⋆ $\varphi \models M = N$ iff $M\varphi = N\varphi$
    where $M\varphi$ is the term where a variable x_i of M, if present, is replaced by t_i

Outline

The Show so Far

Confidentiality

- Formalized by indistinguishability of two traces
- Formalization w.r.t a passive intruder
- One can decide indistinguishability iff one can decide, given two frames φ_1 and φ_2, whether there exists two contexts M and N such that $M\varphi_1 = N\varphi_1$ but $M\varphi_2 \neq N\varphi_2$

We still have to decide such problems for some intruders

Deciding indistinguishability

A several step process

- Double implication: It suffices to decide whether for all M, N we have:

$$M\varphi_1 = N\varphi_1 \Rightarrow M\varphi_2 = N\varphi_2$$

- Prove it is possible to reduce the quantification to a finite set of contexts
- Prove that equality can be decided
- The second part is the most complex one, requires to find some basis of all possibly deducible terms
- Sort of one-sided compilation, where variables act as this basis

Computational soundness

Cryptographic definition of a good primitive

Example: Asymmetric encryption, IND-CCA2

- Given two messages $\{m\}_k^p$ and $\{m'\}_k^p$, it is not possible to tell which one contains m, even with access to encryption/decryption oracles
- Simulates randomness property of encryption

In general, yes/no game-based definition of good cryptography in which the intruder should not be able to perform significantly better than random guess

Relating Computational soundness and Indistinguishability

Idea

- Re-state the properties of the symbolic model by suffixing a negligible probability
- Translate these properties into cryptographic properties
- Do a lot of math to relate all these properties

Restriction

- This is valid for an off-line intruder that doesn't meddle in the executions of the protocol
- Indistinguishability with an on-line intruder is still an open problem
- Permits to prove equivalence of indistinguishability and computational soundness of algebraic operators such as the Xor when applied on nonces

For example there exists a computationally sound implementation of the Xor, but there does not exist any good symbolic abstraction of the Xor

Relating secrecy and confidentiality

Theorems valid for a passive intruder

- It has been shown that with asymmetric primitives and hash function, confidentiality can be reduced to secrecy
- Extra hypothesis needed (Random Oracle Model, non-reachability of a hashed term,. . .
- But this result only holds for passive intruders

Other similar results are awaited

Outline

Tools for automated Analysis

Goal

- Assist protocol developers to eliminate design flaws

Find rapidly and automatically possible flaws

- Assist protocol standardization organisms to validate protocols
  - ⋆ Prove the absence of flaws in isolation from other protocols
  - ⋆ Prove the absence of leaks on the cryptographic level

Outline

Presentation

Basics

- Based on Horn clauses (prolog)
- Performs abstraction on nonces (but does not limit the size of messages
- Initially released as a protocol validation tool

Achievements

- Can treat protocols with an unbounded number of messages
- Permits to prove the security of some protocols
- Used by Microsoft to verify some Web Services

Properties handled

Main advantage of ProVerif: Its versatility!

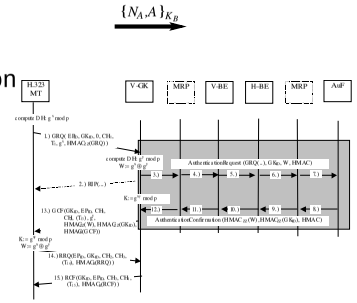
- authentication (mostly weak agreement)
- trace reconstruction (tries to build a trace in the un-abstracted model) corresponding to an attack in the abstraction
- dictionary attacks
- observational equivalence of processes, confidentiality

Outline

The State of the Art...

Several (semi-)automated protocol analyzers

- For example, Clark/Jacob protocol library: NSPK, NSSK, Otway-Rees, Yahalom, Woo-Lam, Denning-Sacco, ...
- Most tools come with their own specification language and user interface.
- Scaling up to large-scale Internet security protocols is a considerable scientific and technological challenge.



...and beyond: the AVISPA Project

- A rich specification language for formalizing industrial strength security protocols and their properties
- Advance state-of-the-art analysis techniques to scale up to this complexity.
- An integrated tool supporting the protocol designer in the debugging and validation of protocols via a uniform and user-friendly interface. $\Longrightarrow$ AVISPA Tool
- Tool assessed on a large collection of practically relevant, industrial protocols $\Longrightarrow$ AVISPA Library
- Migration of this technology to companies and standardization organizations.

AVISPA Tool

- Push-button security protocol analyzer
- Supports the specification of security protocols and properties by means of a rich protocol specification language
- Integrates different back-ends implementing a variety of state-of-the-art automatic analysis techniques
- User interaction facilitated by:
  - ⋆ XEmacs mode.
  - ⋆ Web interface.

AVISPA Tool: Web Interface

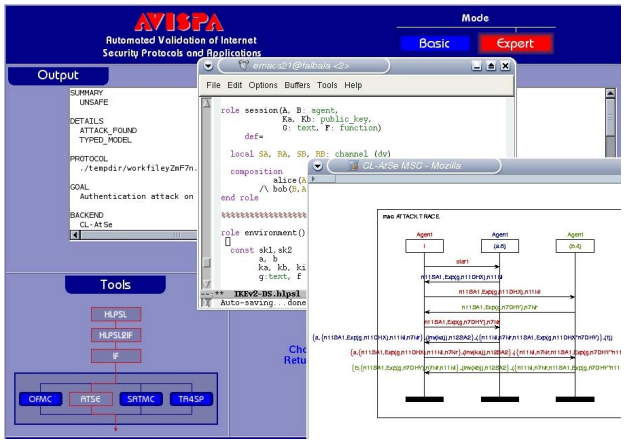
- The AVISPA tool is freely accessible on the web:

  http://www.avispa-project.org

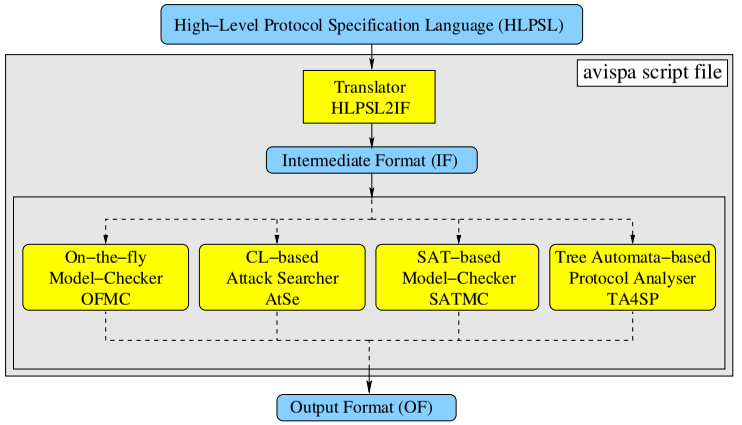
- The interface features:
  - ⋆ A simple editor for HLPSL specifications
  - ⋆ Basic/Expert user modes
  - ⋆ Attacks are graphically rendered with message-sequence charts

just a quick look. . .

Screenshot



AVISPA Tool: Architecture



AVISPA Tool: HLPSL

(High-Level Protocol Specification Language)

A powerful specification language:

- modular, role-based: basic roles (participants) and composed roles (sessions, instances)
- various cryptographic bases: symmetric keys (non-atomic), public/private keys, hash functions, nonces
- typed information (or not): simple and compound types
- algebraic properties: concatenation, exclusive-or, exponentiation

HLPSL (2)

Language properties (cont'd)

- channels: for exchanging messages (Dolev-Yao)
- flow control: guarded transitions
- studied properties: secrecy, weak and strong authentication

Explicit semantics:

- a declarative semantics based on a fragment of Lamport's temporal logic of actions (TLA)
- an operational semantics based on a translation into a rewriting-based formalism: Intermediate Format (IF)

AVISPA Tools for falsification

OFMC: On-the-fly Model-Checker
employs several symbolic techniques to explore the state space in a demand-driven way

AtSe: Constraint-Logic-based Attack Searcher
applies constraints solving with simplification heuristics and redundancy elimination techniques

SATMC: SAT-based Model-Checker
builds a propositional formula encoding all the possible attacks (of bounded length) on the protocol and feeds the result to a SAT solver

AVISPA tools for verification

Only one tool so far...

TA4SP: Tree Automata-based on Automatic Approximations for the Analysis of Security Protocols
approximates the intruder knowledge by using regular tree languages and rewriting to produce under- and over-approximations

But open to other tools, such as ProVerif

AVISPA Library

- Beyond Clark/Jacob (a few seconds for the entire library, with new attacks)
- Selection of a substantial set of security problems associated with protocols that have recently been or are currently being standardized by the IETF
- Formalization in HLPSL of a large subset of these protocols
  $\Longrightarrow$ AVISPA Library
- At present the AVISPA Library comprises 112 security problems derived from 33 protocols

Assessment of AVISPA Tools

- AVISPA Tool assessed by running it against the AVISPA Library
- **Some protocols:** AAAMobileIP, CHAPv2, CRAM-MD5, DHCP-delayed-auth, EKE2, IKEv2-CHILD, IKEv2-DS, IKEv2-MAC, ISO-9798, Kerb-Basic, Kerb-Cross-Realm, Kerb-PKinit, Kerb-Preauth, LPD-MSR, PBK, SRP, TLS, UMTS_AKA
- Also: TA4SP establishes in a few minutes that a number of protocols (EKE, EKE2, TLS, UMTS_AKA, CHAPv2) guarantee secrecy

AVISPA Tool: How to use it?

Main steps: Given a RFC document

1. Write the messages exchange in an Alice&Bob notation
2. Write it in the point of view of each participant
3. Specify a basic role for each participant . . .
4. Specify composition roles: one role representing a session, one environment role representing the required instances
5. Specify properties to check: goal section, plus goal predicates in some transitions of basic roles

Run the AVISPA Tool: the translator will verify the syntax; the chosen back-end will verify the properties.

Conclusion and Future Work

Outcomes of the AVISPA Project

- AVISPA Tool: a state-of-the-art, integrated environment, for automated analysis and validation of Internet security protocols.
- Information on http://www.avispa-project.org:
  - ⋆ Web interface for demos;
  - ⋆ AVISPA package v1.0: officially released in a couple of days!
  - ⋆ Scientific papers, Slides,. . .

Conclusion and Future Work

Future Work

- Collaborations: Siemens AG (IETF), France Telecom.
  Large distribution of the AVISPA Tool $\Longrightarrow$ many future collaborations?
- Diversification of the activities: more properties (non-repudiation, fairness,. . .); more kinds of protocols (contributing protocols, web services, . . .).

Let's go to work !

Demonstration on demand, or online:

http://www.avispa-project.org

Outline

Conclusion

Hot topics

- Relate symbolic analysis with cryptographic analysis
- Indistinguishability w.r.t. an active intruder
- More tools !
- More (better) intruder theories

Questions ?

Network Working Group
Request for Comments: 2945
Category: Standards Track

T. Wu
Stanford University

September 2000

The SRP Authentication and Key Exchange System

Status of this Memo

Copyright Notice

Abstract

This document describes a cryptographically strong network authentication mechanism known as the Secure Remote Password (SRP) protocol. This mechanism is suitable for negotiating secure connections using a user-supplied password, while eliminating the security problems traditionally associated with reusable passwords. This system also performs a secure key exchange in the process of authentication, allowing security layers (privacy and/or integrity protection) to be enabled during the session. Trusted key servers and certificate infrastructures are not required, and clients are not required to store or manage any long-term keys. SRP offers both security and deployment advantages over existing challenge-response techniques, making it an ideal drop-in replacement where secure password authentication is needed.

1. Introduction

The lack of a secure authentication mechanism that is also easy to use has been a long-standing problem with the vast majority of Internet protocols currently in use. The problem is two-fold: Users like to use passwords that they can remember, but most password-based authentication systems offer little protection against even passive attackers, especially if weak and easily-guessed passwords are used.

Eavesdropping on a TCP/IP network can be carried out very easily and very effectively against protocols that transmit passwords in the clear. Even so-called "challenge-response" techniques like the one described in [RFC 2095] and [RFC 1760], which are designed to defeat

simple sniffing attacks, can be compromised by what is known as a "dictionary attack". This occurs when an attacker captures the messages exchanged during a legitimate run of the protocol and uses that information to verify a series of guessed passwords taken from a precompiled "dictionary" of common passwords. This works because users often choose simple, easy-to-remember passwords, which invariably are also easy to guess.

Many existing mechanisms also require the password database on the host to be kept secret because the password P or some private hash h(P) is stored there and would compromise security if revealed. That approach often degenerates into "security through obscurity" and goes against the UNIX convention of keeping a "public" password file whose contents can be revealed without destroying system security.

SRP meets the strictest requirements laid down in [RFC 1704] for a non-disclosing authentication protocol. It offers complete protection against both passive and active attacks, and accomplishes this efficiently using a single Diffie-Hellman-style round of computation, making it feasible to use in both interactive and non-interactive authentication for a wide range of Internet protocols. Since it retains its security when used with low-entropy passwords, it can be seamlessly integrated into existing user applications.

2. Conventions and Terminology

The protocol described by this document is sometimes referred to as "SRP-3" for historical purposes. This particular protocol is described in [SRP] and is believed to have very good logical and cryptographic resistance to both eavesdropping and active attacks.

This document does not attempt to describe SRP in the context of any particular Internet protocol; instead it describes an abstract protocol that can be easily fitted to a particular application. For example, the specific format of messages (including padding) is not specified. Those issues have been left to the protocol implementor to decide.

The one implementation issue worth specifying here is the mapping between strings and integers. Internet protocols are byte-oriented, while SRP performs algebraic operations on its messages, so it is logical to define at least one method by which integers can be converted into a string of bytes and vice versa.

An n-byte string S can be converted to an integer as follows:

```
i = S[n-1] + 256 * S[n-2] + 256^2 * S[n-3] + ... + 256^(n-1) * S[0]
```

where i is the integer and S[x] is the value of the x'th byte of S. In human terms, the string of bytes is the integer expressed in base 256, with the most significant digit first. When converting back to a string, S[0] must be non-zero (padding is considered to be a separate, independent process). This conversion method is suitable for file storage, in-memory representation, and network transmission of large integer values. Unless otherwise specified, this mapping will be assumed.

If implementations require padding a string that represents an integer value, it is recommended that they use zero bytes and add them to the beginning of the string. The conversion back to integer automatically discards leading zero bytes, making this padding scheme less prone to error.

The SHA hash function, when used in this document, refers to the SHA-1 message digest algorithm described in [SHA1].

3. The SRP-SHA1 mechanism

This section describes an implementation of the SRP authentication and key-exchange protocol that employs the SHA hash function to generate session keys and authentication proofs.

The host stores user passwords as triplets of the form

```
{ <username>, <password verifier>, <salt> }
```

Password entries are generated as follows:

```
<salt> = random()
x = SHA(<salt> | SHA(<username> | ":" | <raw password>))
<password verifier> = v = g^x % N
```

The | symbol indicates string concatenation, the ^ operator is the exponentiation operation, and the % operator is the integer remainder operation. Most implementations perform the exponentiation and remainder in a single stage to avoid generating unwieldy intermediate results. Note that the 160-bit output of SHA is implicitly converted to an integer before it is operated upon.

Authentication is generally initiated by the client.

```
  Client                              Host
--------                            ------
U = <username>              -->
                               <--    s = <salt from passwd file>
```


Upon identifying himself to the host, the client will receive the salt stored on the host under his username.

```
a = random()
A = g^a % N                 -->
                                       v = <stored password verifier>
                                       b = random()
                               <--     B = (v + g^b) % N

p = <raw password>
x = SHA(s | SHA(U | ":" | p))

S = (B - g^x) ^ (a + u * x) % N    S = (A * v^u) ^ b % N
K = SHA_Interleave(S)              K = SHA_Interleave(S)
(this function is described
 in the next section)
```

The client generates a random number, raises g to that power modulo the field prime, and sends the result to the host. The host does the same thing and also adds the public verifier before sending it to the client. Both sides then construct the shared session key based on the respective formulae.

The parameter u is a 32-bit unsigned integer which takes its value from the first 32 bits of the SHA1 hash of B, MSB first.

The client MUST abort authentication if B % N is zero.

The host MUST abort the authentication attempt if A % N is zero. The host MUST send B after receiving A from the client, never before.

At this point, the client and server should have a common session key that is secure (i.e. not known to an outside party). To finish authentication, they must prove to each other that their keys are identical.

```
M = H(H(N) XOR H(g) | H(U) | s | A | B | K)
                                  -->
                                  <--    H(A | M | K)
```

The server will calculate M using its own K and compare it against the client's response. If they do not match, the server MUST abort and signal an error before it attempts to answer the client's challenge. Not doing so could compromise the security of the user's password.

If the server receives a correct response, it issues its own proof to the client. The client will compute the expected response using its own K to verify the authenticity of the server. If the client responded correctly, the server MUST respond with its hash value.

The transactions in this protocol description do not necessarily have a one-to-one correspondence with actual protocol messages. This description is only intended to illustrate the relationships between the different parameters and how they are computed. It is possible, for example, for an implementation of the SRP-SHA1 mechanism to consolidate some of the flows as follows:

```
 Client                                Host
--------                              ------
 U, A                          -->
                               <--     s, B
 H(H(N) XOR H(g) | H(U) | s | A | B | K)
                               -->
                               <--     H(A | M | K)
```

The values of N and g used in this protocol must be agreed upon by the two parties in question. They can be set in advance, or the host can supply them to the client. In the latter case, the host should send the parameters in the first message along with the salt. For maximum security, N should be a safe prime (i.e. a number of the form N = 2q + 1, where q is also prime). Also, g should be a generator modulo N (see [SRP] for details), which means that for any X where 0 < X < N, there exists a value x for which g^x % N == X.

3.1. Interleaved SHA

The SHA_Interleave function used in SRP-SHA1 is used to generate a session key that is twice as long as the 160-bit output of SHA1. To compute this function, remove all leading zero bytes from the input. If the length of the resulting string is odd, also remove the first byte. Call the resulting string T. Extract the even-numbered bytes into a string E and the odd-numbered bytes into a string F, i.e.

```
E = T[0] | T[2] | T[4] | ...
F = T[1] | T[3] | T[5] | ...
```

Both E and F should be exactly half the length of T. Hash each one with regular SHA1, i.e.

```
G = SHA(E)
H = SHA(F)
```


Interleave the two hashes back together to form the output, i.e.

```
result = G[0] | H[0] | G[1] | H[1] | ... | G[19] | H[19]
```

The result will be 40 bytes (320 bits) long.

3.2. Other Hash Algorithms

SRP can be used with hash functions other than SHA. If the hash function produces an output of a different length than SHA (20 bytes), it may change the length of some of the messages in the protocol, but the fundamental operation will be unaffected.

Earlier versions of the SRP mechanism used the MD5 hash function, described in [RFC 1321]. Keyed hash transforms are also recommended for use with SRP; one possible construction uses HMAC [RFC 2104], using K to key the hash in each direction instead of concatenating it with the other parameters.

Any hash function used with SRP should produce an output of at least 16 bytes and have the property that small changes in the input cause significant nonlinear changes in the output. [SRP] covers these issues in more depth.

4. Security Considerations

This entire memo discusses an authentication and key-exchange system that protects passwords and exchanges keys across an untrusted network. This system improves security by eliminating the need to send cleartext passwords over the network and by enabling encryption through its secure key-exchange mechanism.

The private values for a and b correspond roughly to the private values in a Diffie-Hellman exchange and have similar constraints of length and entropy. Implementations may choose to increase the length of the parameter u, as long as both client and server agree, but it is not recommended that it be shorter than 32 bits.

SRP has been designed not only to counter the threat of casual password-sniffing, but also to prevent a determined attacker equipped with a dictionary of passwords from guessing at passwords using captured network traffic. The SRP protocol itself also resists active network attacks, and implementations can use the securely exchanged keys to protect the session against hijacking and provide confidentiality.

SRP also has the added advantage of permitting the host to store passwords in a form that is not directly useful to an attacker. Even if the host's password database were publicly revealed, the attacker would still need an expensive dictionary search to obtain any passwords. The exponential computation required to validate a guess in this case is much more time-consuming than the hash currently used by most UNIX systems. Hosts are still advised, though, to try their best to keep their password files secure.

5. References

[RFC 1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[RFC 1704] Haller, N. and R. Atkinson, "On Internet Authentication", RFC 1704, October 1994.

[RFC 1760] Haller, N., "The S/Key One-Time Password System", RFC 1760, Feburary 1995.

[RFC 2095] Klensin, J., Catoe, R. and P. Krumviede, "IMAP/POP AUTHorize Extension for Simple Challenge/Response", RFC 2095, January 1997.

[RFC 2104] Krawczyk, H., Bellare, M. and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[SHA1] National Institute of Standards and Technology (NIST), "Announcing the Secure Hash Standard", FIPS 180-1, U.S. Department of Commerce, April 1995.

[SRP] T. Wu, "The Secure Remote Password Protocol", In Proceedings of the 1998 Internet Society Symposium on Network and Distributed Systems Security, San Diego, CA, pp. 97-111.

6. Author's Address

Thomas Wu
Stanford University
Stanford, CA 94305

EMail: tjw@cs.Stanford.EDU

7. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

École de Printemps Cryptographie et Sécurité Informatique
Analyses de Protocoles Cryptographiques

Yannick Chevalier

Sorrèze, 24-28 avril 2006

1 Cryptographic Protocol Models

1.1 Protocols

Bibliographical background. The security problems have been the object of numerous panels in the CSFW workshop serie. These panel are announced by a 2-3 pages paper describing the position of participants. I suggest reading of these papers, as they are non-technical, simple to understand and often instructive. In particular I suggest [73, 5] for secrecy and confidentiality, and [52] for authentication. For this latter property, note however that the "must-read" paper is by G. Lowe [62], where the different notions of authentication (called agreement) are defined.

Privacy. There is no mention fo anonymity in this list because it corresponds to a more off-line and accounting property not directly related to the protocols, though protocols may be used to ensure privacy. In fact it could be argued that the techniques described here do not apply for privacy analysis, since privacy analysis is concerned with the behavior of *a priori* honest users, whereas we are mainly concerned here with the behavior of an external aggressor, even if this aggressor may impersonate (play the role of) an honest user.

The protocol. This protocol was proposed by Needham and Schroeder in 1978 [68]. Though the message sequence originally defined is flawed, it has been used as the basis of numerous protocols, including Kerberos [32]. The justification presented was formalized in BAN Logic [27], thus giving a "proof" of the protocol. The absence of patch on this prove was the main ground for the adoption of symbolic analysis of protocols based on messages exchange instead of

intended results of encryption. The flaw presented here was found automatically by G. Lowe [58, 59] among other flaws [60].

In fact, the correction is not fullproof. While in [59] no other attacks was found, Cathy Meadows [65], using her own analyzer NRL [66], found a flaw depending on the possibility of type confusion between a nonce and a principal's name. More recently, Bogdan Warinschi showed [77] that the El Gamal encryption scheme should not be used for this protocol as it would open the possibility of cryptographic attacks.

Finally, let us mention that the proposed description of a protocol is based on [61] and [49], but we will come back to this latter.

1.2 Dolev-Yao model

This model was defined as an abstraction over bit-strings by Dolev and Yao, though they only considered public key encryption and pairing [49][1]. In later development, some assumptions, such as the fact that there exists a secure public key infrastructure, where removed because the purpose of some protocols is precisely to establish such an infrastructure. The fact that there is no obvious relation between to seemingly close terms is to be related to the properties required on some cryptographic algorithm, such that *e.g.* to change only one bit in the key must change at least half of the bits in the encoded message (see for example the description of DES in [75][2]).

The model permits to define robustness with respect to a polynomial-time adversary, but also permits to give an intuitive interpretation of what the research of an *attack* is. It is the synthesis of a program employing cryptographic primitives which is able to interfere on-line with the desired properties of the protocol. It is quite different from the cryptographer's definition of an attack, where one looks for off-line attacks possibly relying on a limited use of on-line activity.

However, the two approaches have much in common. First, the assumption of non-collision of messages constructed differently is expressed, at the cryptographic level, by stating that the probability of such a collision is *negligible*, that is decrease more rapidly than the inverse of any polynomial of a security parameter k (*e.g.* the length of the keys used). The fact that the length of a message leaks is also taken into account by Michael Backes [14].

The fact that no other equality may be found is the main difference with the cryptographic level of analysis. Indeed, cryptanalysis is concerned with the existence of equalities with small probability. Other types of equalities can be easily introduced, for example to model block-encryption [33].

[1] No idea on who introduced symmetric encryption and hash functions

[2] Beware, there exists several errors in the french translation of this book!

Concerning the possible control of intruder on the network, one can read with interest litterature on the distributed denial of services, on the experiments using snort and so on to be persuaded that the control described here is not unlikely strong. The case of Wifi networks, is a bit more complicated, though [78] gives a good account on how to mount attacks on the WEP protocol. Some softwares are already available *via* sourceforge, *e.g.* WEP Crack [79] and Air Snort [9].

Finally, intruder's deductions were first defined in the multiset rewritting framework by Iliano Cervesato *et al.* in [31]. The presentation given here is a simplification for theoretical purposes defined in [72], though it is there described without equations, but using the simplifications described later. The extension (using terms instead of symbols) was first used for ease-of-proof reasons in [37], and was generalized in [38].

1.3 Properties

I will not present any further the authentication property, and refer to [62] for in-depth covering. I will instead give more details on confidentiality in Section ?? in order to focus here on non-repudiation. This property encompasses several classes defined by the non-repudiable action : Non-repudiation of reception, non-repudiation of emission,.... J. Zhou and D. Gollmann were the first in [81] to formalize this notion from the working drafts released by ISO/IEC. An example of reduction of non-repudiation to several authentication goals is given in [74], for a protocol proposed by J. Zhou and D. Gollmann [80]. In the meantime, analysis was conducted using dedicated procedures, such as games [55]. Finally, it is worth mentionning that a different operational notion of authentication was defined in [54], where the emphasis was on a gradual exchange of informations.

2 Reachability Analysis

2.1 Operational semantics

I present here an operational semantics based on the multiset rewriting formalism, that was proposed for the analysis of protocols by Cervesato *et al* [31]. I have abstracted from the details of the implementation, hiding the terms describing the current state of the principals and relying instead on sharing of variables between messages of a same principal (to check correspondance on received messages), ordering on the rules (to model the state in the execution flow of the principals) and existentials, which are explicitely used in [31] to model nonces as new values that have never occurred before. It is worth mentioning that the current Intermediate Format defined in the AVISPA project is extremely close to the one defined in [31].

There has been much work on the definition of formal models. I believe the first one was (again) by Gavin Lowe [61]. It has been ameliorated in [53] to enable roles to check again messages that were previously accepted. This situation occurs for example when one first receives a message encrypted by the symmetric key k, and later the key k itself. Such a situation is the basis of the non-repudiation protocols discussed earlier. Several other articles have been published on the topic [24, 29] independently from these first two. Finally, let us mention [33], where the compilation process was extended to take into account exclusive or, exponentiation and commutative encryption.

2.2 Reachability analysis

First, one shall note that the hypothesis of well-typed messages is not used to prove the decidability in the case of a finite number of messages, but it is mandatory to have decidability in the case of an infinite number of messages, even if only a finite number of constants are permitted. In fact, one can easily encode a Post Correspondance Problem using only pairs and a symmetric encryption operator with a key k unknown to the intruder.

It seems the first algorithm was given by M. Boreale (see [23] for a longer version), and was proven by R. Amadio and D. Lugiez [10] (see [11] for an extended version including asymmetric encryption and improvements on the algorithm).

Case of a bounded number of messages. These original algorithms were then extended to handle non-atomic keys independently ([34] for a description of an implementation and [67] for a decidability proof). The analysis was shown to be NP-complete in [72][3]. Several papers were subsequently in the same line without improving significantly either the implementation or the proof. I feel that the simplest complexity proof, and the only one based on lazy constraint solving to our knowledge, is by H. Comon-Lundh [40]. I strongly suggest the reading of it for clarity purposes. A small improvement in [33] is to show that the algorithm covers all possible attacks (and thus outputs a more precise answer than a simple attack/no attack) and that it is possible to combine it with normalization (which is anyway done in all implementations), and in [19] that it is possible to decide whether the intruder has the same set of possible actions in two instances of the protocol. In [17] another improvement (constraints differenciation) is introduced, though it does not seem to be effective in all cases [12]. Finally, a lazy intruder implementation with the exclusive or is proposed in [35] and is implemented in CL-Atse by M. Turuani[4].

[3] NP-hardness was given in [10]

[4] Unfortunately, no description is available yet.

Unbounded number of messages. The sketch of proof of the undecidability presented here is extracted from [51], long version of an article published in 1999), as well as the DEXPTIME result for the case with a bounded number of constants. The authors also present an extension where the principals can check that a nonce has not been received before, but give a DEXPTIME complexity, thought it was later shown to be NEXPTIME-complete.

The restriction to a finite number of nonces was an inspiration to many authors. Among the subsequent work, the most proeminent work is the first implementation of ProVerif by B. Blanchet [21] together with the proof of the termination of the analysis for tagged protocols, in collaboration with A. Podelski [22]. There are a number of other different proofs of decidability of secrecy for tagged protocols, see for example [71]. Another line of work was initiated by G. Lowe and P. Broadfoot and focused on proving that it suffices to analyze a protocol with respect to a bounded number of constants (bound determined from the protocol) [26].

Efficient model-checking. The conditions given are minimal at the moment. They are used by L. Mazaré [64] to give an extension of the analysis algorithm to the case where the variables are not totally ordered. For the analysis of bounded sessions, some experimental results are given in [50, 13, 12]. For the unbounded analysis, the most efficient tool at this time seems to be ProVerif [21], which is also very versatile with respect to the proved properties. Finally, though the implementation is now out of date, I dare to mention [39] which presents, to the best of my knowledge, the only attempt at proving strong authentication (injective aggreement) in the unbounded setting.

2.3 Algebraic and probabilistic primitives

First let us recall that this implementation of Needham-schroeder protocol has been shown to be computationally incorrect by B. Warinschi [77].

Theories treated. The three theories presented here were the most studied ones. The case of the Exclusive or with respect to a passive intruder was treated in [41] where a polynomial bound was given for the ground reachability problem. Independently, [37] proved this bound and the NP-completeness for the combination of the xor and the usual Dolev-Yao operators. I recall also the implementation proposition of [35].

The abelian group (multiplication, here) was also treated in the case of a passive intruder in [41]. The active case was first completed in [76]. It is also considered as an application example in [38].

Finally, the exponentiation operator with respect to an active intruder has been treated under different assumptions in [36, 76]. Note that the proof of the

latter seems incomplete (from the author's web site).

Negative result. The simulatability notion was introduced by B. Pfitzmann and M. Waidner in [70]. This notion was applied to the analysis of cryptographic protocols with M. Backes in [15], as a way to bring to the symbolic level the cryptographic games used to prove computational correctness. Note that, in parallel, a similar notion of Universal Composability was developed by R. Canetti [30] to the same goal. Numerous protocols and properties were proved with this relation (the list is too long to cite). One shall note that the simulatibility relation between an implementation satisfying some requirements (often around the resistance to chosen-ciphertext attacks) and a symbolic representation of this implementation as a term[5] It was proved in [16] that there exists no symbolic representation (under very wide hypotheses) of the cryptographic Xor that holds a simulatability relation with a xor implementation. This can be seen either as the fact that the simulatability relation requires too much from the symbolic level, or as the fact that one should validate protocol using the xor, as the validation may not reduce to a validation at the cryptographic level.

Positive results. See ??. P. Lafourcade, D. Lugiez and R. Treinen have first treated the case of symbolic xor and homomorphism with respect to a passive intruder [56]. In [48] and with S. Delaune prove that this intruder theory is decidable in presence of an active intruder. However, homomorphism properties are hard to handle, and in [47] it is proved that if one replace xor with abelian groups, the problem becomes undecidable.

Implementations. They are hard to find ! I only know three tools that handle not trivial algebraic properties :

- ProVerif, in which it is possible to specify an equational theory between terms ;
- Ofmc, which can also be parameterized by user-specified axioms [18] ;
- CL-Atse, that has complete support (in its most recent version) of xor terms and partial (excluding list unification) support for lists and sets.

Note that for the two first tools, termination is an issue. For the second tool, technically involved unaddressed issues[6] make the tool both non-terminating and incomplete.

[5] In the simulatibility relation, it is admissible that the length of a message leaks.

[6] *I.e.* the fact that narrowing modulo an underlying theory is used, and it is well known that this method is not complete semi-decision procedure for unification.

3 Equivalence Analysis

This part is certainly very incomplete. It should be considered as an outsider's point of view. However, there is at least some minimal things that can be explained.

Defining knowledge. The definition of process equivalence given here was defined by M. Abadi and A. Gordon in their seminal paper on the π-calculus [6]. One can also read the papers of Halpern on the topic of knowledge in a distributed environment for an intuitive explanation.

Indistinguishability. The problem of deciding indistinguishability with respect to a passive intruder was solved by M. Abadi and V. Cortier in two steps, first in the case of a Dolev-Yao intruder [3], and then for more general intruder theories [4]. The problem is very similar to compilation, since in this latter case one desires to find a "most general frame" such that for any instance of the variables in this frame, the resulting frame satisfies all equations satisfied by a correct run of the protocol. This case was solved for Dolev-Yao intruder augmented with xor and exponential operations in [33], though no justification is given for the computation of the "most general frame". The idea of the procedure given is, given a finite set of term E, to compute a set of terms E' such that any term that can be constructed from E can be constructed from E'.

The connection between observational equivalence and computational soundness was first established by M. Abadi and Rogaway in [7, 8]. There is a nice development of this relation in [20], where it is proved that observational equivalence between frames permits to define a computationally sound abstraction of the exclusive or (among other primitives). This result is to be related to the one presented earlier that there exists no abstraction of Xor in a simulatability-like relation with its implementation. Finally, let me mention [42], where a computationally sound abstraction of hash functions and asymmetric encryption primitives is presented (in the Random Oracle Model) but also where the problem of frame equivalence is reduced to reachability problems.

4 Tools for Protocol Analysis

4.1 ProVerif

ProVerif [21] was first released as a tool permitting the validation o cryptographic protocols under a finite number of nonces abstraction. It has since been considerably enriched, and now also permits analysis of confidentiality [1], and thus validation that extends to the underlying cryptographic system. It now

used in the Samoa Project to analyze web services [63], though proofs have to be partly conducted by hand [2].

4.2 AVISPA – a tutorial

I will give an example of how to write an HLPSL specification from a RFC. The protocol (SRP) was chosen only because it is defined by a very short RFC. This tutorial and AVISPA Project presentation was handed by **Laurent Vigneron**, and I thank him here.

Références

[1] Martín Abadi and Bruno Blanchet. Analyzing Security Protocols with Secrecy Types and Logic Programs. *Journal of the ACM*, 52(1) :102–146, January 2005.

[2] Martín Abadi and Bruno Blanchet. Computer-Assisted Verification of a Protocol for Certified Email. *Science of Computer Programming*, 58(1–2) :3–27, October 2005. Special issue SAS'03.

[3] Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under equational theories. In Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, *ICALP*, volume 3142 of *Lecture Notes in Computer Science*, pages 46–58. Springer, 2004.

[4] Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under (many more) equational theories. In CSFW2005 [46], pages 62–76.

[5] Martin Abadi, Riccardo Focardi, Catherine Meadows, Jonathan Millen, and Dennis M. Volpano. Formalization and proof of secrecy properties. In *CSFW*, pages 92–95, 1999.

[6] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols : The spi calculus. *Inf. Comput.*, 148(1) :1–70, 1999.

[7] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In Jan van Leeuwen, Osamu Watanabe, Masami Hagiya, Peter D. Mosses, and Takayasu Ito, editors, *IFIP TCS*, volume 1872 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2000.

[8] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *J. Cryptology*, 15(2) :103–127, 2002.

[9] Airsnort software. Implementation of an attack against WEP : `http://airsnort.shmoo.com/`.

[10] Roberto M. Amadio and Denis Lugiez. On the reachability problem in cryptographic protocols. In Catuscia Palamidessi, editor, *CONCUR*, volume 1877 of *Lecture Notes in Computer Science*, pages 380–394. Springer, 2000.

[11] Roberto M. Amadio, Denis Lugiez, and Vincent Vanackère. On the symbolic reduction of processes with cryptographic functions. *Theor. Comput. Sci.*, 290(1) :695–740, 2003.

[12] Alessandro Armando, David A. Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, Paul Hankes Drielsma, Pierre-Cyrille Héam, Olga Kouchnarenko, Jacopo Mantovani, Sebastian Mödersheim, David von Oheimb, Michaël Rusinowitch, Judson Santiago, Mathieu Turuani, Luca Viganò, and Laurent Vigneron. The avispa tool for the automated validation of internet security protocols and applications. In Kousha Etessami and Sriram K. Rajamani, editors, *CAV*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285. Springer, 2005.

[13] Alessandro Armando, David A. Basin, Mehdi Bouallagui, Yannick Chevalier, Luca Compagna, Sebastian Mödersheim, Michaël Rusinowitch, Mathieu Turuani, Luca Viganò, and Laurent Vigneron. The aviss security protocol analysis tool. In Brinksma and Larsen [25], pages 349–353.

[14] Michael Backes. *Cryptographically Sound Analysis of Security Protocols*. PhD thesis, Saarland University, 2002.

[15] Michael Backes, Christian Jacobi 0002, and Birgit Pfitzmann. Deriving cryptographically sound implementations using composition and formally verified bisimulation. In Lars-Henrik Eriksson and Peter A. Lindsay, editors, *FME*, volume 2391 of *Lecture Notes in Computer Science*, pages 310–329. Springer, 2002.

[16] Michael Backes and Birgit Pfitzmann. Limits of the cryptographic realization of dolev-yao-style xor. In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, *ESORICS*, volume 3679 of *Lecture Notes in Computer Science*, pages 178–196. Springer, 2005.

[17] David A. Basin, Sebastian Mödersheim, and Luca Viganò. An on-the-fly model-checker for security protocol analysis. In Einar Snekkenes and Dieter Gollmann, editors, *ESORICS*, volume 2808 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2003.

[18] David A. Basin, Sebastian Mödersheim, and Luca Viganò. Algebraic intruder deductions. In Geoff Sutcliffe and Andrei Voronkov, editors, *LPAR*, volume 3835 of *Lecture Notes in Computer Science*, pages 549–564. Springer, 2005.

[19] Mathieu Baudet. Deciding security of protocols against off-line guessing attacks. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *ACM Conference on Computer and Communications Security*, pages 16–25. ACM, 2005.

[20] Mathieu Baudet, Véronique Cortier, and Steve Kremer. Computationally sound implementations of equational theories against passive adversaries. In Caires et al. [28], pages 652–663.

[21] Bruno Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In CSFW2001 [45], pages 82–96.

[22] Bruno Blanchet and Andreas Podelski. Verification of cryptographic protocols : Tagging enforces termination. In Andrew D. Gordon, editor, *FoSSaCS*, volume 2620 of *Lecture Notes in Computer Science*, pages 136–152. Springer, 2003.

[23] Michele Boreale. Symbolic trace analysis of cryptographic protocols. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *ICALP*, volume 2076 of *Lecture Notes in Computer Science*, pages 667–681. Springer, 2001.

[24] Sébastien Briais and Uwe Nestmann. A formal semantics for protocol narrations. In Rocco De Nicola and Davide Sangiorgi, editors, *TGC*, volume 3705 of *Lecture Notes in Computer Science*, pages 163–181. Springer, 2005.

[25] Ed Brinksma and Kim Guldstrand Larsen, editors. *Computer Aided Verification, 14th International Conference, CAV 2002,Copenhagen, Denmark, July 27-31, 2002, Proceedings*, volume 2404 of *Lecture Notes in Computer Science*. Springer, 2002.

[26] Philippa J. Broadfoot and A. W. Roscoe. Capturing parallel attacks within the data independence framework. In *CSFW*, pages 147–159. IEEE Computer Society, 2002.

[27] Michael Burrows, Martín Abadi, and Roger M. Needham. A logic of authentication. *ACM Trans. Comput. Syst.*, 8(1) :18–36, 1990.

[28] Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors. *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*. Springer, 2005.

[29] Carlos Caleiro, Luca Viganò, and David A. Basin. Deconstructing alice and bob. *Electr. Notes Theor. Comput. Sci.*, 135(1) :3–22, 2005.

[30] Ran Canetti. Universally composable security : A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.

[31] Iliano Cervesato, Nancy A. Durgin, Patrick Lincoln, John C. Mitchell, and Andre Scedrov. A meta-notation for protocol analysis. In *CSFW*, pages 55–69, 1999.

[32] Iliano Cervesato, Catherine Meadows, and Dusko Pavlovic. An encapsulated authentication logic for reasoning about key distribution protocols. In CSFW2005 [46], pages 48–61.

[33] Y. Chevalier. *Résolution de problèmes d'accessibilité pour la compilation et la validation de protocoles cryptographiques*. Thèse de doctorat, Université Henri Poincaré, Nancy, décembre 2003.

[34] Y. Chevalier and L. Vigneron. Towards Efficient Automated Verification of Security Protocols. In *Proceedings of the Verification Workshop (VERIFY'01) (in connection with IJCAR'01), Università degli studi di Siena, TR DII 08/01*, pages 19–33, Siena (Italy), June 2001.

[35] Yannick Chevalier. A simple constraint-solving decision procedure for protocols with exclusive or. Technical Report RR-5224, INRIA, June 2004.

[36] Yannick Chevalier, Ralf Küsters, Michaël Rusinowitch, and Mathieu Turuani. Deciding the security of protocols with diffie-hellman exponentiation and products in exponents. In Pandya and Radhakrishnan [69], pages 124–135.

[37] Yannick Chevalier, Ralf Küsters, Michaël Rusinowitch, and Mathieu Turuani. An np decision procedure for protocol insecurity with xor. In LICS2003 [57], pages 261–270.

[38] Yannick Chevalier and Michaël Rusinowitch. Combining intruder theories. In Caires et al. [28], pages 639–651.

[39] Yannick Chevalier and Laurent Vigneron. Automated unbounded verification of security protocols. In Brinksma and Larsen [25], pages 324–337.

[40] Hubert Comon-Lundh. Short proof for complexity of reachability analysis in a finite number of sessions. available at `http://www.lsv.ens-cachan.fr/~comon/CRYPTO/`.

[41] Hubert Comon-Lundh and Vitaly Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In LICS2003 [57], pages 271–.

[42] Véronique Cortier and Bogdan Warinschi. Computationally sound, automated proofs for security protocols. In Shmuel Sagiv, editor, *ESOP*, volume 3444 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.

[43] *Ninth IEEE Computer Security Foundations Workshop, March 10 - 12, 1996, Dromquinna Manor, Kenmare, County Kerry, Ireland*. IEEE Computer Society, 1996.

[44] *10th Computer Security Foundations Workshop (CSFW '97), June 10-12, 1997, Rockport, Massachusetts, USA*. IEEE Computer Society, 1997.

[45] *14th IEEE Computer Security Foundations Workshop (CSFW-14 2001), 11-13 June 2001, Cape Breton, Nova Scotia, Canada*. IEEE Computer Society, 2001.

[46] *18th IEEE Computer Security Foundations Workshop, (CSFW-18 2005), 20-22 June 2005, Aix-en-Provence, France*. IEEE Computer Society, 2005.

[47] Stéphanie Delaune. An undecidability result for agh. Technical Report LSV-06-02, Laboratoire Spécification et Vérification, ENS Cachan, France, February 2006.

[48] Stéphanie Delaune, Pascal Lafourcade, Denis Lugiez, and R. Treinen. Symbolic protocol analysis in presence of a homomorphism operator and exclusive or. Technical Report LSV-05-20, Laboratoire Spcification et Vrification, ENS Cachan, France, November 2005.

[49] Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2) :198–207, 1983.

[50] Ben Donovan, Paul Norris, and Gavin Lowe. Analyzing a library of security protocols using casper and fdr. In *In Proceedings of the Workshop on Formal Methods and Security Protocols*, page???, 1999.

[51] Nancy A. Durgin, Patrick Lincoln, and John C. Mitchell. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2) :247–311, 2004.

[52] Roberto Gorrieri, Paul F. Syverson, Martín Abadi, Riccardo Focardi, Dieter Gollmann, Gavin Lowe, and Catherine Meadows. Panel introduction : Varieties of authentication. In *CSFW*, pages 79–82, 1998.

[53] Florent Jacquemard, Michaël Rusinowitch, and Laurent Vigneron. Compiling and verifying security protocols. In Michel Parigot and Andrei Voronkov, editors, *LPAR*, volume 1955 of *Lecture Notes in Computer Science*, pages 131–160. Springer, 2000.

[54] Steve Kremer, Olivier Markowitch, and Jianying Zhou. An intensive survey of fair non-repudiation protocols. *Computer Communications*, 25(17) :1606–1621, 2002.

[55] Steve Kremer and Jean-François Raskin. A game-based verification of non-repudiation and fair exchange protocols. In Kim Guldstrand Larsen and Mogens Nielsen, editors, *CONCUR*, volume 2154 of *Lecture Notes in Computer Science*, pages 551–565. Springer, 2001.

[56] Pascal Lafourcade, Denis Lugiez, and Ralf Treinen. Intruder deduction for *ac*-like equational theories with homomorphisms. In Jürgen Giesl, editor, *RTA*, volume 3467 of *Lecture Notes in Computer Science*, pages 308–322. Springer, 2005.

[57] *18th IEEE Symposium on Logic in Computer Science (LICS 2003), 22-25 June 2003, Ottawa, Canada, Proceedings*. IEEE Computer Society, 2003.

[58] Gavin Lowe. An attack on the needham-schroeder public-key authentication protocol. *Inf. Process. Lett.*, 56(3) :131–133, 1995.

[59] Gavin Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. In Tiziana Margaria and Bernhard Steffen, editors, *TACAS*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer, 1996.

[60] Gavin Lowe. Some new attacks upon security protocols. In CSFW1996 [43], pages 162–169.

[61] Gavin Lowe. Casper : A compiler for the analysis of security protocols. In CSFW1997 [44], pages 18–30.

[62] Gavin Lowe. A hierarchy of authentication specification. In CSFW1997 [44], pages 31–44.

[63] Kevin D. Lux, Michael J. May, Nayan L. Bhattad, and Carl A. Gunter. Wsemail : Secure internet messaging based on web services. In *IEEE International Conference on Web Services (ICWS'05)*, pages 75–82, 2005.

[64] Laurent Mazaré. Satisfiability of dolev-yao constraints. *Electr. Notes Theor. Comput. Sci.*, 125(1) :109–124, 2005.

[65] Catherine Meadows. Analyzing the needham-schroeder public-key protocol : A comparison of two approaches. In Elisa Bertino, Helmut Kurth, Giancarlo Martella, and Emilio Montolivo, editors, *ESORICS*, volume 1146 of *Lecture Notes in Computer Science*, pages 351–364. Springer, 1996.

[66] Catherine Meadows. Language generation and verification in the nrl protocol analyzer. In CSFW1996 [43], pages 48–61.

[67] Jonathan K. Millen and Vitaly Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *ACM Conference on Computer and Communications Security*, pages 166–175, 2001.

[68] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12) :993–999, 1978.

[69] Paritosh K. Pandya and Jaikumar Radhakrishnan, editors. *FST TCS 2003 : Foundations of Software Technology and Theoretical Computer Science, 23rd Conference, Mumbai, India, December 15-17, 2003, Proceedings*, volume 2914 of *Lecture Notes in Computer Science*. Springer, 2003.

[70] Birgit Pfitzmann and Michael Waidner. Composition and integrity preservation of secure reactive systems. In *ACM Conference on Computer and Communications Security*, pages 245–254, 2000.

[71] R. Ramanujam and S. P. Suresh. Tagging makes secrecy decidable with unbounded nonces as well. In Pandya and Radhakrishnan [69], pages 363–374.

[72] Michaël Rusinowitch and Mathieu Turuani. Protocol insecurity with finite number of sessions is np-complete. In CSFW2001 [45], pages 174–.

[73] Peter Ryan. A genealogy of non-interference. In CSFW1996 [43], pages 158–.

[74] J. Santiago and L. Vigneron. Study for Automatically Analysing Non-repudiation. In *Actes du 1er Colloque sur les Risques et la Sécurité d'Internet et des Systèmes, CRiSIS*, pages 157–171, Bourges, France, October 2005.

[75] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, 1996.

[76] Vitaly Shmatikov. Decidable analysis of cryptographic protocols with products and modular exponentiation. In David A. Schmidt, editor, *ESOP*, volume 2986 of *Lecture Notes in Computer Science*, pages 355–369. Springer, 2004.

[77] Bogdan Warinschi. A computational analysis of the needham-schröeder-(lowe) protocol. In *CSFW*, pages 248–. IEEE Computer Society, 2003.

[78] Description of how to mount attacks in a wifi setting, and more specifically how to mount attacks on wep. Survey on the techniques that can be used to attack a 802.11 protocol : `http://www.isaac.cs.berkeley.edu/isaac/wep-faq.html`.

[79] Wep crack. Implementation of an attack against WEP : `http://wepcrack.sourceforge.net/`.

[80] Jianying Zhou and Dieter Gollmann. A fair non-repudiation protocol. In *IEEE Symposium on Security and Privacy*, pages 55–61. IEEE Computer Society, 1996.

[81] Jianying Zhou and Dieter Gollmann. Observations on non-repudiation. In Kwangjo Kim and Tsutomu Matsumoto, editors, *ASIACRYPT*, volume 1163 of *Lecture Notes in Computer Science*, pages 133–144. Springer, 1996.

Cryptologie asymétrique, factorisation et logarithme discret

Jean-Marc Couveignes

Professeur au GRIMM
France

Cryptologie asymétrique, factorisation, logarithme discret

Jean-Marc COUVEIGNES

G.R.I.M.M. — Université Toulouse II

Quelques notions de complexité

- Problèmes décisionnels (e.g. primalité) / Problèmes fonctionnels (e.g. factorisation)
- Langage décidé par une machine de Turing
- décidabilité / recursive énumérabilité
- fonction calculable
- PTIME = $\cup_{d\geq 0}$ TIME $(n \mapsto n^d)$
- machines de Turing déterministes / non-déterministes
- NPTIME = $\cup_{d\geq 0}$ NTIME $(n \mapsto n^d)$
- co-NP
- PRIME ?

Arithmétique dans $\mathbb{Z}$

- addition, soustraction : linéaires
- multiplication, division euclidienne : quadratique mais ...
- multiplication rapide : Karatsuba, FFT
- Algorithme d'Euclide
- Calcul dans $(\mathbb{Z}/N\mathbb{Z}, +, \times)$
- Autres corps finis ?
- $\mathbb{F}_p[X]$
- $\mathbb{F}_p[X]/P(X)\mathbb{F}_p[X]$
- Complexité bilinéaire pour p fixé et $\deg(P) \to \infty$?

Montrer qu'un nombre est premier

- PRIME $\in$ NP ?
- P premier ssi $\#(\mathbb{Z}/P\mathbb{Z})^* = P - 1$
- exhiber un générateur g
- $P - 1 = \prod_i q_i$
- $m_i = (P-1)/q_i$ diviseur maximal
- $g^{m_i} \neq 1 \bmod P - 1$
- preuve de primalité pour q_i
- et recursivement ...
- certificat de primalité !

Montrer qu'un nombre est composé

- exhiber un facteur ! PRIME $\in$ co-NP
- Fermat : si $x \neq 0 \bmod P$ alors $x^{P-1} = 1 \bmod P$
- Miller-Rabin : $x^{P-1} - 1 = (x^{\frac{P-1}{2}} - 1)(x^{\frac{P-1}{2}} + 1) = (x^{\frac{P-1}{4}} - 1)(x^{\frac{P-1}{4}} + 1)(x^{\frac{P-1}{2}} + 1) = \cdots$
- Témoin de non-primalité
- Monier : il y a au plus $\frac{\phi(P)}{4}$ faux témoins
- En fait cela montre que PRIME $\in$ co-RP
- L $\in$ RP : il existe un algorithme probabiliste tel que
- $x \in L \Rightarrow$ OUI avec proba $\geq \frac{1}{2}$
- $x \notin L \Rightarrow$ NON tout le temps

Que sait-on de PRIME ?

- dans NP $\cap$ co-NP
- dans co-RP
- Adleman et Huang : dans RP
- Agrawal, Kayal, Saxena : dans P
- En pratique ? Miller-Rabin ou pour les artistes :
- Cyclotomie : Lenstra, Pomerance
- Courbes elliptiques (Atkin-Morain)
- Remplacer $(\mathbb{Z}/P\mathbb{Z})^*$ ensemble des solutions de $y^2 = x^3 + x$ modulo P par exemple.
- Si $P = a^2 + b^2$ alors il y a $P + 1 - t$ solutions avec $t = \pm 2a$ ou $\pm 2b$

Exercices

- c'est quoi co-P ?
- c'est quoi FNP ? exemple ?
- P $\subset$ NP $\subset$ PSPACE $\subset$ EXPTIME
- GRAPH-ISOMORPHISM $\in$ NP
- GRAPH-ISOMORPHISM $\in$ co-NP ?
- vecteur le plus court d'un réseau ?
- ZPP = RP $\cap$ co-RP (Las Vegas)
- RP (Monte Carlo)
- BPP : erreur avec probabilité $\frac{1}{3}$
- ZPP $\subset$ RP $\subset$ BPP ?

classe IP

- L$\in$ IP : protocole $\mathcal{P}$ avec prouveur A dans EXPTIME et vérifieur B polynomial non-déterministe
- $x \in L \Rightarrow \mathcal{P}(A,B)[x,w] =$ OUI pour tout w,
- $x \notin L \Rightarrow$ pour tout prouveur A' on a $\mathcal{P}(A',B)[x,w] =$ NON avec proba $\geq \frac{1}{2}$.
- graph non-isomorphism ?
- IP=PSPACE (Lund, Fortnow, Karloff, Nisan, Shamir)
- factorisation ? vecteur le plus court ?

Réductions et problèmes complets

- $L_1 \subset \{0,1\}^*$ se réduit à $L_2 \subset \{0,1\}^*$ s'il existe une fonction de réduction $\mathcal{R} : \{0,1\}^* \to \{0,1\}^*$ dans FLOGSPACE telle que $\mathcal{R}(x) \in L_2$ ssi $x \in L_1$
- problème dur ou complet pour une classe
- SAT est NP-complet
- 3SAT, 3DM, 3COLORS, sac à dos, ...
- vecteur le plus court : en norme L^∞ (Van Emde Boas), en norme L^2 mais réduction probabiliste (Ajtai)
- oracle, réduction many-to-one : racine carrée modulo N et factorisation

Factorisation et racines carrées

- si $N = pq$ alors $(\mathbb{Z}/N\mathbb{Z})^* = (\mathbb{Z}/p\mathbb{Z})^* \times (\mathbb{Z}/q\mathbb{Z})^*$ a quatre racines carrées de 1.
- exemple : $p = 7$ et $q = 11$: on a $7 \times 8 = 1 \bmod 11$ et $11 \times 2 = 1 \bmod 7$
- $r = 1 \times 2 \times 11 - 1 \times 8 \times 7 = 43$
- $< -1 > \times < 43 >= (\mathbb{Z}/N\mathbb{Z})^*[2] = \{1, -1, 43, 34\}$
- réciproquement $\mathrm{pgcd}(N, r-1) = \mathrm{pgcd}(N, 33) = 11$
- Fermat : chercher a et b tels que $a^2 = b^2 \bmod N$
- $N = 45$, $\sqrt{N} \approx 6.708$, on pose $a = 7$ et $a^2 - N = 4 = b^2$ avec $b = 2$. Donc $(a-b)(a+b) = N = 5 \times 9$

Une fonction asymétrique ?

- $(p, q) \mapsto N = pq$
- trial division : $T \approx N^{\frac{1}{2}}$
- Méthodes ρ et $p-1$ de Pollard : $T \approx p^{\frac{1}{2}}$
- Cribles : Pomerance, Lenstra, Pollard, . . . : $T \approx \exp((\log N)^{\frac{1}{2}})$ ou même $\exp((\log N)^{\frac{1}{3}})$
- idée : calcul de $\#(\mathbb{Z}/N\mathbb{Z})^* = (p-1)(q-1)$ par générateurs et relations
- générateurs : $\ell \leq B$ petits premiers
- relations : $x = x + N \bmod N$, et $x = \prod_{\ell \leq B} \ell^{e_\ell}$ et $x + N = \prod_{\ell \leq B} \ell^{f_\ell}$

Crible quadratique

- congruences plus astucieuses
- $N = 21311$ et $m = \lceil \sqrt{N} \rfloor = 146$
- $(m+a)^2 = m^2 - N + 2am + a^2 = 5 + 292a + a^2 \bmod N$
- On choisit $B = 13$ et on explore les petites valeurs de a :

a	$5 + 292a + a^2$
-27	$-2.5^2.11.13$
-5	$-2.5.11.13$
-1	$-2.11.13$
0	5
60	$5^3.13^2$

Algèbre linéaire

a	$5+292a+a^2$
-27	$-2.5^2.11.13$
-5	$-2.5.11.13$
-1	$-2.11.13$
0	5
60	$5^3.13^2$

Matrice

	-1	2	5	11	13
-27	1	1	0	1	1
-5	1	1	1	1	1
-1	1	1	0	1	1
0	0	0	1	0	0
60	0	0	1	0	0

Noyau et relations

Matrice :

	-1	2	5	11	13
-27	1	1	0	1	1
-5	1	1	1	1	1
-1	1	1	0	1	1
0	0	0	1	0	0
60	0	0	1	0	0

Noyau :

-27	-5	-1	0	60
1	0	1	0	0
1	1	0	1	0
1	1	0	0	1

$(2.5.11.13)^2 \equiv (146-27)^2.(146-1)^2 \pmod{21311}.$

Exponentielle et logarithme discrets

- G groupe cyclique, e.g. $G = (\mathbb{Z}/P\mathbb{Z})^*$ avec $P = 7$
- g générateur de G, exemple $g = 3$, $g^2 = 2$, $g^3 = 6$, $g^4 = 4$, $g^5 = 5$
- $x \mapsto g^x$ définit $exp_g : \mathbb{Z}/(P-1)\mathbb{Z} \to (\mathbb{Z}/P\mathbb{Z})^*$
- $0 \mapsto 1$, $1 \mapsto 3$, $2 \mapsto 2$, $3 \mapsto 6$, $4 \mapsto 4$, $5 \mapsto 5$
- application réciproque $\log_g : h = g^x \mapsto x$
- exponentiation rapide : $g_0 = g$, $g_1 = g^2$, $g_2 = g_1^2 = g^4$, $\ldots g_k = ?$
- $g_0 g_2 g_4 g_5 = g^{53}$
- Pas de log rapide !

Propriétés du log discret

- G groupe cyclique de cardinal γ, g générateur
- $g^{x+y} = g^x g^y$ donc $\exp_g(x+y) = \exp_g(x)\exp_g(y)$ et $\log_g(uv) = \log_g(u) + \log_g(v)$
- changement de base : h autre générateur, $\log_h(u) = \frac{\log_g(u)}{\log_g(h)}$
- H ensemble des générateurs de G
- $A = (\mathbb{Z}/\gamma\mathbb{Z})^* = \mathrm{Aut}(G)$ agit sur H
- $a.h = h^a$, action fidèle et libre $A \times H \to H$
- espace homogène difficile
- exemple : $G = (\mathbb{Z}/7\mathbb{Z})^*$, alors $A = \{1, 5\}$ et $H = \{3, 5\}$.

Complexité du logarithme discret

- G groupe cyclique quelconque de cardinal γ :
- algorithme naïf : $T \approx \gamma$
- algorithme de Shanks : $T \approx \sqrt{\gamma}$ et pas mieux (Maurer)
- $G = (\mathbb{Z}/N\mathbb{Z}, +)$, trivial
- $G = ((\mathbb{Z}/N\mathbb{Z})^*, \times)$, cribles $T \approx \exp((\log N)^{\frac{1}{2}})$
- courbe elliptique générique : apparemment comme groupe générique
- autres groupes : groupes de classes (Buchman), variétés abéliennes plus compliquées (genre 2 et 3)

Génération de secret et identification

- G cyclique de cardinal γ et H ensemble des générateur et A ensemble des automorphismes
- Alice choisit $g \in H$ et $a \in A$ et calcule $h = g^a$. Elle publie G, g, h.
- pour se prouver auprès de Bob, elle choisit un exposant b au hasard dans A et calcule $k = h^b = g^{ab}$ et elle envoie k à Bob.
- Bob tire $\epsilon \in \{0, 1\}$ au hasard
- si $\epsilon = 0$ Bob demande à Alice quel est l'automorphisme qui envoie g sur k et Alice doit répondre $c = ab \bmod \phi(\gamma)$
- si $\epsilon = 1$ Bob demande à Alice quel est l'automorphisme qui envoie h sur k et Alice doit répondre b
- preuve sans apport d'information

Un exemple d'identification

Alice choisit $G = (\mathbb{Z}/11\mathbb{Z})^*$ cyclique de cardinal $\gamma = 10$ et $H = \{2, 8, 7, 6\}$ ensemble des générateurs :

```
? vector(11,k,lift(Mod(2,11)^k))
%1 = [2, 4, 8, 5, 10, 9, 7, 3, 6, 1, 2]
```

et $A = \{1, 3, 7, 9\} = (\mathbb{Z}/10\mathbb{Z})^*$ ensemble des automorphismes.

Alice choisit $g = 2$ et $a = 7$ et elle calcule $h = g^a = 7$. Elle publie $(11, 2, 7)$.

Pour se prouver auprès de Bob, elle choisit un exposant $b = 3$ au hasard dans A et calcule $k = h^b = 7^3 = 2$ et elle envoie $k = 2$ à Bob.

Bob tire $\epsilon \in \{0, 1\}$ au hasard.

Un exemple d'identification (suite)

Bob tire $\epsilon \in \{0, 1\}$ au hasard.

Si $\epsilon = 0$ Bob demande à Alice quel est l'automorphisme qui envoie $g = 2$ sur $k = 2$ et Alice doit répondre $c = 3 \times 7 = 1 \bmod 4$.

Si $\epsilon = 1$ Bob demande à Alice quel est l'automorphisme qui envoie 7 sur 2 et Alice doit répondre $b = 3$.

Partage de secret

- G groupe cyclique quelconque de cardinal γ et H ensemble des générateur et A ensemble des automorphismes (exposants)
- Alice choisit $g \in H$ et $a \in A$ et calcule h comme g^a. Elle publie G, g, h
- Bob choisit $b \in A$ et calcule $k = g^b$. Il publie k
- Alice calcule $S = k^a$ et Bob calcule $S' = h^b$ et on vérifie que $S = k^a = g^{ba} = g^{ab} = h^b = S'$
- le méchant a vu : G, g, h, k
- utile pour l'échange de clé

Exemple de partage de secret

On a encore $G = (\mathbb{Z}/11\mathbb{Z})^*$ cyclique de cardinal $\gamma = 10$ et $H = \{2, 8, 7, 6\}$ ensemble des générateurs et $A = \{1, 3, 7, 9\} = (\mathbb{Z}/10\mathbb{Z})^*$ ensemble des automorphismes.
Alice choisit $g = 2$ et $a = 7$ et elle calcule $h = g^a = 7$. Elle envoie G, g et $h = 7$ à Bob.
Bob choisit $b = 9$ et calcule $k = g^b = 6 \bmod 11$. Il envoie $k = 6$ à Alice.
Alice calcule $S = k^a = 6^7 = 8 \bmod 11$ et Bob calcule $S' = h^b = 7^9 = 8 \bmod 11$.

Chiffrement Vernam (masque jetable)

- Idée : brouiller le message avec un bruit connu du destinataire
- $m = 010101101011\ldots$ et $k = 100111010\ldots$ clé aussi longue que le message
- chiffré $c = m \oplus k = 11001\ldots$ donc déchiffrement par $m = c \oplus k$
- si c connu alors k et m sont équivalents : sécurité inconditionnelle
- ne surtout pas utiliser plusieurs fois la même clé : pas très pratique
- générateur pseudo-aléatoire ? comment faire beaucoup de bruit avec peu d'information ...

Chiffrement de El Gamal

- Alice a choisi G et g et $a \in A$. Elle a calculé $h = g^a$ et a publié G, g et h. Le triplet (G, g, h) est la clé publique
- Bob veut envoyer m à Alice. On suppose que m est un élément de G. Bob trouve (G, g, h) dans l'annuaire. Il choisit b dans A et l'applique à g ce qui donne $k = g^b$. Il calcule aussi $t = h^b$ et $c = mt$
- Bob envoie (k, c)
- Alice calcule $k^a = g^{ba} = g^{ab} = h^b = t$ et $m = c/t$
- Le chiffré est deux fois plus long que le clair mais il n'y a qu'un échange

Exemple de chiffrement de El Gamal

Alice a choisi $G = (\mathbb{Z}/11\mathbb{Z})^*$ et $g = 7$ et $a = 3 \in A = \{1, 3, 7, 9\}$.
Elle a calculé $h = g^a = 7^3 = 2 \bmod 11$ et a publié G, $g = 7$ et $h = 2$.
Bob veut envoyer $m = 4$ à Alice. Bob trouve (G, g, h) dans l'annuaire. Il choisit $b = 9$ dans A et l'applique à g ce qui donne $k = g^b = 7^9 = 8$. Il calcule aussi $t = h^b = 2^9 = 6$ et $c = mt = 4 \times 6 = 2$.
Bob envoie $(k, c) = (8, 2)$ à Alice.
Alice calcule $k^a = 8^3 = 6 = t$ et $m = c/t = 2/6 = 4 \bmod 11 = m$!

RSA : une autre fonction trappe

- $N = pq$ entier composé et $G = (\mathbb{Z}/N\mathbb{Z})^*$
- $\gamma = \#G = (p-1)(q-1) = \varphi(N)$ connu si p et q connus
- Lagrange : si $x \in G$ alors $x^\gamma = 1$ et $A = \mathrm{Aut}(G) = (\mathbb{Z}/\gamma\mathbb{Z})^*$
- Soit e dans A (souvent on choisit $e = 3$)
- L'application $m \mapsto m^e$ est bijective et asymétrique si l'on ne connaît pas γ
- Si on connaît γ alors soit $f = e^{-1} \bmod \gamma$. Alors $(m^e)^f = m^{ef} = m^{1+*\gamma} = m$.
- Pour connaître γ il faut connaître p et q.

RSA

- Génération de clé : Alice choisit deux grands nombres premiers (il y en a ? comment les trouver ?) et elle les multiplie $N = pq$. Elle choisit e premier à $\gamma = \varphi(N) = (p-1)(q-1)$. Elle calcule $f = e^{-1} \bmod \gamma$ et publie N, e et garde secrets p, q, γ et f.
- Chiffrement : Bob veut envoyer le message m à Alice. On suppose que m est un entier entre 0 et N et premier à N. Bob trouve N, e dans l'annuaire et calcule $c = m^e \bmod N$. Il envoie c à Alice.
- Déchiffrement : Alice calcule $c^f = m^{ef} = m \bmod N$.

Un exemple de RSA

Génération de clé : Alice choisit deux grands nombres premiers $p = 5$ et $q = 11$ et elle les multiplie $N = pq = 55$. Elle choisit $e = 3$ premier à $\gamma = \varphi(N) = (p-1)(q-1) = 40$. Elle calcule $f = e^{-1} \bmod \gamma = 27$ et publie $N = 55$, $e = 3$ et garde secrets p, q, γ et f.
Chiffrement : Bob veut envoyer le message $m = 23$ à Alice. Bob trouve $N = 55$, $e = 3$ dans l'annuaire et calcule $c = m^e \bmod N = 23^3 = 12$. Il envoie $c = 12$ à Alice.
Déchiffrement : Alice calcule $c^f = 12^{27} = 23 \bmod N = m$.

Arithmétique appliquée I

Jean-Marc Couveignes

15 septembre 2001

Présentation

La première partie de ce cours rassemble une exposition rapide des notions essentielles de la théorie de la complexité et une introduction à la théorie algorithmique des nombres. Il s'agit de préparer l'introduction de la cryptographie à clés publiques dans la deuxième partie du cours. Cette deuxième partie comportera aussi les rudiments indispensables de géométrie algorithmique, sur les courbes algébriques et leurs jacobiennes en particulier.

Table des matières

Chapitre 1

Quelques notions de complexité

1.1 Introduction

La théorie de la complexité compte les opérations élémentaires et mesure l'espace nécessaire à la résolution d'un problème donné. La question n'a de sens que pour une famille infinie de problèmes similaires qui doivent être résolus par le même algorithme.

Cela sous-entend que l'on se restreint à des problèmes décidables (il existe un algorithme, même très maladroit, qui résout le problème).

La mémoire ('espace') est mesurée en bits et le temps en opérations logiques ou arithmétiques élémentaires telles que l'addition ou la multiplication de deux bits. Un accès à la mémoire, pour lire ou écrire, est supposé requérir un temps constant.

La complexité en temps (resp. espace) d'un algorithme est une fonction de $\mathbb{N}$ dans $\mathbb{N}$ qui pour toute instance ('entrée') x du problème donne un majorant du temps d'exécution de l'algorithme (resp. de la mémoire nécessaire) qui dépend uniquement de la taille $|x|$ de x. La complexité n'est donc pas unique mais si l'on se restreint à une famille totalement ordonnée de fonctions naturelles, on peut souvent définir la complexité comme le minimum ou la borne inférieure parmi toutes les fonctions majorant le temps (resp. l'espace) dans cette famille.

Exercice 1 *On s'intéresse aux algorithmes de tri. Soit donc $L = (L_1, L_2, ..., L_n)$ une liste (vecteur) de réels. On cherche une liste $M = (M_1, M_1, ..., M_n)$ ordonnée i.e. $M_1 \leq M_2 \leq ... \leq M_n$ telle que $\{L_1, ..., L_n\} = \{M_1, ..., M_n\}$.*

Le principe du tri à bulles est de parcourir la liste initiale de gauche à droite et de permuter deux éléments successifs chaque fois qu'ils ne sont pas ordonnés. On réitère jusqu'à ce que la liste soit triée (ce qui se traduit par l'absence de permutations dans la dernière passe).

Montrer que cet algorithme s'arrête. Pour mesurer sa complexité, on majore le nombre de transpositions nécessaires pour trier une liste de n éléments. Expliquer

pourquoi cette convention est raisonnable. Montrer que ce nombre est borné par une constante fois n^2.

Si donc on mesure la complexité par une fonction de la forme $n \mapsto An^\alpha$ avec A et α réels positifs, on peut choisir $\alpha = 2$. On se désintéresse de A. On dit que le tri à bulles est de complexité quadratique en temps.

La complexité intrinsèque d'un problème est la complexité du meilleur algorithme qui résolve ce problème. Cela signifie encore que l'on se restreint à une famille ordonnée raisonnable de fonctions.

Exercice 2 *Une autre façon de trier consiste à sélectionner le plus grand élément par éliminations successives, à la manière d'un tournoi. On groupe donc les éléments par paires et on désigne le plus grand de chaque paire. Les vainqueurs de ce premier tour sont alors confrontés par paires et ainsi de suite, jusqu'à la finale. Cet algorithme détermine seulement le plus grand des éléments. Pour le transformer en algorithme de tri, on imagine une structure hierarchique en forme d'arbre binaire. Au sommet de l'arbre, le chef, qui a deux sous-chefs, chacun ayant deux sous-sous-chefs etc. On suppose qu'au début de l'algorithme toutes les places de l'organigramme sont vacantes sauf celles du plus bas niveau hiérarchique, où l'on place la liste à trier. Ces places du niveau inférieur sont les feuilles de l'arbre binaire. Dans chaque paire de feuilles on choisit la plus grande valeur et on la promeut au niveau supérieur. Les entrées du deuxième niveau sont ensuite comparées par paires et la meilleure est promue au troisième niveau. Et ainsi de suite. Mais on prend bien garde de pourvoir en cascade chaque place laissée vacante par une promotion en promouvant ensuite le meilleur des deux subordonnés du dernier promu, et ainsi de suite récursivement. Ce processus sélectif par paliers se poursuit jusqu'au niveau suprême et détermine la plus grande valeur de la liste (le premier secrétaire du Parti). On retire alors cette valeur de l'arbre (mort du premier secrétaire) et on organise son remplacement récursif en promouvant le meilleur des deux secrétaires adjoints puis en pourvoyant la place de secrétaire adjoint ainsi libérée et ainsi de suite jusqu'à la base.*

Montrer que cet algorithme trie une liste de taille n avec moins de $An \log n$ comparaisons pour un réel A qu'on ne cherchera pas à déterminer.

Au vu de cet exercice et du précédent, que peut on dire de la complexité du problème du tri ?

Il est souvent difficile de déterminer la complexité exacte d'un problème. Chaque algorithme prouvé donne une borne supérieure. Les bornes inférieures proviennent souvent de considération naïves de théorie de l'information.

Exercice 3 *Prouver qu'un algorithme de tri ne peut trier toute les listes de n éléments en moins de $n - 1$ comparaisons.*

S'il s'agit de trier, la différence entre les algorithmes efficaces comme le tri par sélection et les algorithmes naïfs comme le tri à bulles s'exprime en disant que ceux-ci sont quadratiques et ceux-là linéaires (on néglige le terme $\log n$).

Bien que cette différence d'efficacité soit spectaculaire pour n de l'ordre de 10^9 (penser à la sécurité sociale en Chine), les deux algorithmes ont une complexité polynomiale en n c'est-à-dire polynomiale en la taille des données du problème.

On considère maintenant le problème du chemin dans un graphe. On se donne un graphe fini sous la forme d'un ensemble fini V de sommets et d'un sous-ensemble E de $V \times V$ qui représente les arêtes. On se donne aussi deux sommets A et B et on demande (s'il existe) un chemin de A vers B. Un tel chemin est donné par une succession de sommets $v_0 = A$, v_1, ..., $v_I = B$ telle que pour tout i, il existe une arête entre v_i et v_{i+1}. L'algorithme naïf explore systématiquement (en arbre) tous les chemins partant de A. Il est donc de complexité 2^n exponentielle en $n = \#V$.

Exercice 4 *Donner un algorithme polynomial pour le problème du chemin dans un graphe et donner sa complexité i.e. le plus petit réel α tel que le nombre d'opérations élémentaires soit $O(n^\alpha)$.*

La complexité de certains problèmes simples et naturels est parfois imprévue.

Par exemple il est naturel de se donner un groupe G et un ensemble d'éléments σ_0, σ_1, ..., σ_n, et de se demander si σ_0 appartient au sous-groupe de G engendré par σ_1, ..., σ_n. C'est le problème de l'appartenance.

Si G est le groupe de permutations de k lettres alors ce problème se résout en temps polynomial en n et k par d'intéressantes et utiles techniques générales introduites par des topologues du siècle dernier (voir [9, chapitre 2] et [7, chapitre 8]).

Si G est le groupe $GL_2(\mathbb{Z})$ des matrices inversibles à coefficients entiers, alors ce problème est indécidable, c'est-à-dire qu'il n'existe même pas d'algorithme pour le résoudre. Il en va de même si G est un groupe donné par générateurs et relations (voir [5] pour un historique).

Revenant au cas où G est un groupe fini, un problème classique consiste à calculer le cardinal du sous-groupe engendré par des éléments donnés. Ce problème est aussi difficile que celui de l'appartenance. En effet, si je sais calculer le cardinal d'un sous-groupe, je peux calculer le cardinal de $< \sigma_1, \sigma_2, ..., \sigma_n >$ et celui de $< \sigma_0, \sigma_1, ..., \sigma_n >$. Ils sont égaux si et seulement si $\sigma_0 \in < \sigma_1, \sigma_2, ..., \sigma_n >$. Cette méthode qui résout le problème de l'appartenance si l'on sait résoudre celui du cardinal est appelée une *réduction*. L'existence de telles réductions définit une relation d'ordre entre les problèmes (ici le cardinal est aussi difficile que l'appartenance pour les groupes finis). Encore faut-il préciser quels types de réduction on autorise (cela dépend du contexte). Par exemple, pour résoudre une instance du problème $\mathcal{A}$, s'autorise-t'on à poser un, deux ou plusieurs instances du problème $\mathcal{B}$? Et quelle puissance de calcul nécessite la solution d'une instance du problème $\mathcal{A}$ si l'on suppose résolu le problème B ?

Ici par exemple, on utilise deux instances du problème du cardinal pour résoudre une instance du problème de l'appartenance. Et le temps nécessaire pour la traduction entre les deux problèmes est linéaire en la taille des entrées.

Un autre problème célèbre dit problème des mariages consiste, étant donnés deux ensembles A (les mariés) et B (les mariéEs) finis de même cardinalité n et un sous-ensemble M de $A \times B$ (les mariage arrangeables), à trouver s'il en existe une bijection μ de A vers B telle que pour tout $a \in A$ le couple $(a, \mu(a))$ soit dans M (μ est un ensemble de mariages monogames qui ne laisse personne seul).

Ce problème est appelé 2DM (2-dimentional matching) et il se résout en temps polynomial de façon non-triviale [19, Chapitre I].

Un autre problème très proche consiste à se donner trois ensembles A, B, C de même taille n et un sous-ensemble M de $A \times B \times C$ (l'ensemble des mariage à trois possibles). On cherche, s'il en existe, deux bijections $\mu : A \to B$ et $\nu : A \to C$ telles que pour tout $a \in A$ le mariage à trois $(a, \mu(a), \nu(a))$ soit dans M. On ne connaît pas d'algorithme polynomial pour résoudre ce problème et nous verrons qu'il y a peu de chances pour qu'il en existe un. Ce problème est appelé 3DM (3-dimentional matchnig).

Cependant, notons une propriété intéressante de ce problème : il est facile de vérifier la validité d'une solution.

Cela nous conduit à quelques définitions classiques.

On distingue premièrement deux types de problèmes. Les problèmes dont la réponse ne peut être que OUI ou NON sont appelés problèmes de *décision* (par exemple, *ce nombre est il premier ?* ou *ce graphe est il connexe ?*. Les autres problèmes dont la réponse est plus longue (par exemple, *quel est le produit de ces deux nombres ?* ou *quel est le cardinal de ce groupe ?*) sont appelés problèmes fonctionnels.

Si un problème de décision (resp. fonctionnel) admet un algorithme polynomial, on dit qu'il appartient à la classe de complexité **P** (resp. **FP**).

Si la réponse à un problème fonctionnel peut être vérifiée en temps polynomial, avec le secours éventuel d'une preuve, on dit que le problème est dans la classe de complexité **FNP**.

Par exemple, soit G est un groupe cyclique de cardinal $\#G$. On suppose qu'il est possible de calculer dans G (représenter, comparer, multiplier et inverser des éléments) en temps polynomial en $\log \#G$. On peut choisir $G = (\mathbb{Z}/p\mathbb{Z})^*$ avec p entier premier. Soit g un générateur de G et h un élément quelconque de G. Le problème fonctionnel consistant à trouver un entier k tel que $h = g^k$ est appelé *logarithme discret* dans G. Ce problème est dans **FNP** car la réponse se vérifie aisément à l'aide d'une exponentiation rapide.

Si la réponse OUI à un problème de décision peut être vérifiée en temps polynomial, avec le secours éventuel d'une preuve, on dit que le problème est dans la classe de complexité **NP**.

Si la réponse NON à un problème de décision peut être vérifiée en temps polynomial, avec le secours éventuel d'une preuve, on dit que le problème est

dans la classe de complexité **coNP**.

Par exemple, la primalité (i.e. le problème *ce nombre est-il premier ?*) est dans **coNP** car si la réponse est non, on en donne une preuve aisément vérifiable en exhibant un facteur.

Exercice 5 *Montrer que la primalité est dans* **NP**. *Il faut montrer l'existence de preuves simples de primalité. Or un nombre n est premier si et seulement si le groupe $(\mathbb{Z}/n\mathbb{Z})^*$ a pour cardinal $n-1$. Il suffit donc d'exhiber un générateur g de ce groupe et une preuve que g est bien d'ordre $n-1$. Développer cet argument.*

Exemple 1 *Un graphe $\mathcal{G}$ est donné sous la forme d'un couple (V, E) avec V ensemble des sommets et $E \subset V \times V$ ensemble des cotés. Soient $\mathcal{G}_1 = (V_1, E_1)$ et $\mathcal{G}_2 = (V_2, E_2)$ deux graphes finis. On demande s'ils sont isomorphes i.e. s'il existe une bijection de V_1 dans V_2 qui induise une bijection de E_1 dans E_2. Ce problème de décision est dans* **NP** *(exercice) mais il n'est pas connu pour être dans* **coNP**. *En d'autres termes, on ne sait pas prouver efficacement que deux graphes ne sont pas isomorphes.*

Les classes de complexité regroupent les problèmes selon leur difficulté. Il est possible de leur donner un statut d'ensembles à l'aide de la théorie des machines de Turing que nous effleurerons bientôt. Les classes que nous avons rencontrées jusqu'alors correspondent à des complexités en temps mais on pourrait aussi bien considérer des classes associées à des complexités en espace. Pour être rigoureux on devrait noter **PTIME** la classe **P** afin de la distinguer de la classe **PSPACE** qui rassemble les problèmes solubles à l'aide d'un espace mémoire polynomial en la taille des données (sans restriction sur le temps d'exécution).

Exercice 6 *Une restriction en espace sous-entend une restriction en temps car la mémoire d'une machine ne peut pas se trouver deux fois dans le même état au cours d'un même calcul (sinon on boucle ....). En déduire que* **PSPACE** *est contenue dans la classe* **EXPTIME** *des problèmes solubles en temps exponentiel $\exp(n^A)$ où n est la taille des entrées et A un réel quelconque dépendant du problème.*

1.2 Machines de Turing

Il n'a pas échappé au lecteur sourcilleux que les définitions du paragraphe précédent souffrent d'un certain manque de rigueur. Si l'on se restreint à un cadre assez étroit (les algorithmes de tri, les algorithmes de transformée de Fourier, etc.), le point de vue adopté jusqu'alors est justifié : il suffit de compter les opérations les plus coûteuses, celles qui sont le pain quotidien des algorithmes concernés. Mais si l'on prétend définir des classes plus générales de problèmes, il

est nécessaire de disposer d'un modèle uniforme de calcul. C'est le mérite des machines de Turing de fournir un tel modèle. Il existe hélas plusieurs définitions possibles des machines de Turing (avec une ou plusieurs bandes etc.) qui conduisent à des définitions légèrement différentes de la complexité de tel ou tel problème. Cependant, les classes générales de complexité comme **P** et **NP** sont indépendantes de ces variantes.

Une machine de Turing est un automate déterministe muni d'une bande mémoire infinie à droite sur laquelle elle peut lire ou écrire à l'aide d'un unique curseur. Le temps est discret. Le comportement de la machine est déterminé par une fonction de transition qui, connaissant l'état de la machine et le caractère lu sur la bande à l'endroit où se trouve le curseur, détermine l'état suivant, le caractère à écrire à la place du caractère courant, et le mouvement du curseur.

Plus précisément une machine de Turing est la donnée d'un quadruplet $M = (S, \Sigma, \delta, s)$ où S est l'ensemble fini des états de la machine, $s \in S$ est l'état initial, Σ est l'alphabet utilisé par M et contient au moins les deux symboles $\triangleright$ (symbole initial) et $\square$ (blanc) (souvent $\Sigma = \{0, 1, \triangleright, \square\}$), et δ est la fonction de transition qui va de $S \times \Sigma$ dans $S \cup \{\text{STOP, OUI, NON}\} \times \Sigma \times \{\leftarrow, \rightarrow, \bullet\}$. Cette fonction détermine l'état suivant (qui peut être l'un des trois états finaux OUI, NON ou STOP) le caractère à écrire et le déplacement du curseur (vers la gauche, vers la droite ou pas de déplacement). Initialement, le curseur est sur la première case de la bande mémoire et cette case contient le symbole initial $\triangleright$ suivi des entrées (ou données du problème). Le reste de la bande est vierge (il ne contient que des blancs $\square$). La machine n'est pas autorisée à effacer le symbole $\triangleright$ et ce symbole l'oblige à se déplacer vers la droite, de sorte qu'elle ne sort jamais de la bande mémoire. Ces contraintes s'expriment par la condition que pour tout état p il existe un état q tel que $\delta(p, \triangleright) = (q, \triangleright, \rightarrow)$.

Les machines de Turing sont bien adaptées à l'étude des problèmes de décision. Si $L \subset \{0,1\}^*$ est un langage, on dit que la machine de Turing T décide L si pour toute entrée $x \in \{0,1\}^*$ la machine s'arrête dans l'état OUI si $x \in L$ et dans l'état NON si $x \notin L$. Si un langage est décidé par une machine de Turing, on dit qu'il est décidable. Il existe des langages non-décidables (ne serait-ce que pour des raisons simples de cardinalité) mais la théorie de la complexité se restreint d'ordinaire aux problèmes décidables. Comme certains problèmes très naturels sont indécidables (comme le problème de l'appartenance dans les groupes de présentation finie) il est prudent de ne pas méconnaître ces notions cependant.

Les logiciens introduisent une notion plus faible que la décidabilité. On dit qu'un langage L est récursivement énumérable s'il existe une machine de Turing T qui s'arrête pour toute entrée $x \in L$ et qui ne s'arrête pas si $x \notin L$. Cette notion est plus faible que la décidabilité car dans le cas où $x \notin L$ on n'est jamais fixé. Un exemple naturel et fameux est celui des équations diophantiennes (équations algébriques dont les inconnues sont des nombre entiers). L'existence d'une solution à une équation diophantienne est un problème de décision évidemment récursivement énumérable (il suffit d'essayer toutes les solutions) mais un résultat

fameux de Matiyasevich prouve que ce problème n'est pas décidable, résolvant ainsi le dixième problème de Hilbert [16].

Un langage est décidable si et seulement si lui et son complémentaire sont récursivement énumérables.

Les machines de Turing sont utiles aussi pour l'étude des problèmes fonctionnels. Si une machine de Turing M initialisée avec l'entrée x s'arrête dans l'état STOP, on appelle sortie et on note $M(x)$ la chaîne présente alors sur la bande mémoire de M à droite du $\triangleright$ initial. On dit qu'une fonction $f : \{0,1\}^* \to \{0,1\}^*$ est récursive si elle est calculée par une machine de Turing. Le temps de calcul est le nombre d'étapes avant l'arrêt de la machine.

Si $f : \mathbb{N} \to \mathbb{N}$ est une fonction on note $\mathbf{TIME}(f)$ l'ensemble des langages reconnus par une machine de Turing en temps $\leq f(n)$ où n est la taille des entrées. Cette définition ne présente pas un très grand intérêt car la moindre altération dans la définition des machines de Turing en modifierait le sens. En revanche, la classe $\mathbf{PTIME}$ est définie comme l'union de $\mathbf{TIME}(n \mapsto n^d)$ pour $d \geq 1$ et cette définition résiste à toutes les variations raisonnables dans les définition des machines de Turing.

Remarque 1 *Comme on voit, les machines de Turing permettent de définir les grandes classes de complexité introduites à la section précédente avec la rigueur qui y manquait. Il reste que des distinctions importantes comme celles entre algorithmes linéaires, cubiques ou quadratiques sont mal exprimées par cette théorie. Cela tient au caractère un peu primitif de ces machines. Un modèle plus réaliste est donné par les Random Access Machines. En fait, pour une étude plus fine de la complexité d'une famille donnée d'algorithmes, le point de vue de la section précédente (compter les opérations principales) est préférable.*

Les machines de Turing que nous venons d'introduire sont dites déterministes. On définit les machines de Turing non-déterministes de la même manière à ceci près que la fonction de transition δ n'est pas une fonction mais une relation quelconque ce qui signifie que plusieurs transitions sont possibles. On dit qu'une machine de Turing non-déterministe accepte un langage L si pour toute entrée x, il existe une exécution (succession de transitions possibles) qui conduit à l'état OUI si et seulement si $x \in L$. On peut simuler une machine de Turing non-déterministe par une machine déterministe qui recevrait outre l'entrée x une *preuve* p. Cette preuve p indique à la machine quelle transition effectuer et la guide ainsi vers la bonne réponse. En somme, une machine de Turing non-déterministe se borne à vérifier la validité d'une preuve.

Ainsi, pour une fonction $f : \mathbb{N} \to \mathbb{N}$ on note $\mathbf{NTIME}(f)$ la classe des problèmes de décision résolus par une machine non-déterministe en temps $\leq f(|x|)$. Cela signifie que si la réponse pour l'entrée x est OUI, il en existe une preuve vérifiable en temps $\leq f(|x|)$.

On définit ainsi la classe $\mathbf{NP}$ comme l'union des $\mathbf{NTIME}$ $(n \mapsto n^d)$ pour tout $d \geq 1$.

Exercice 7 *Montrer que* $\mathbf{P} \subset \mathbf{NP} \subset \mathbf{PSPACE} \subset \mathbf{EXPTIME}$

Une notion importante est celle d'*oracle*. Un oracle est un auxiliaire d'une machine de Turing qui lui fournit en une seule unité de temps la réponse à un problème donné. Soit $f : \mathbb{N} \to \mathbb{N}$ une fonction et Q un problème de décision. On dit qu'un langage L est dans $\mathbf{TIME}^{\,Q}(f(n))$ si et seulement s'il est reconnu en temps $\leq f(n)$ par une machine de Turing déterministe ayant recours à un oracle capable de résoudre en une unité de temps toute instance du problème Q. Dans la même veine, un langage L est dans $\mathbf{NP}^{\,3DM}$ si et seulement s'il est reconnu en temps polynomial par une machine non-déterministe ayant recours à un oracle capable de résoudre en une unité de temps toute instance du problème 3DM des mariages à trois.

Plus généralement, un langage L est dans $\mathbf{NP}^{\,\mathbf{NP}}$ si et seulement s'il existe un problème Q dans $\mathbf{NP}$ tel que L est dans $\mathbf{NP}^{\,Q}$.

Les machines de Turing permettent aussi de définir rigoureusement la notion de réduction.

Un problème de décision i.e. un langage L_1 se réduit à un problème de décision L_2 si et seulement s'il existe une fonction de réduction $\mathcal{R} : \{0,1\}^* \to \{0,1\}^*$ dans $\mathbf{FLOGSPACE}$ telle que pour toute instance $x \in \{0,1\}^*$ du premier problème on ait $x \in L_1$ si et seulement si $\mathcal{R}(x) \in L_2$.

Ici la classe $\mathbf{FLOGSPACE}$ est celle des fonction calculables en espace $O(\log |x|)$.

On dit qu'un problème est *complet* pour une classe de complexité s'il est dans cette classe et s'il est aussi difficile que tous les problèmes de cette classe (il existe des réductions de tous les problèmes de la classe au problème en question).

On dit qu'un problème est *dur* pour une classe de complexité s'il est aussi difficile que tous les problèmes de cette classe sans être nécessairement dedans.

On connaît un grand nombre de problèmes $\mathbf{NP}$-complet (i.e. complets pour la classe $\mathbf{NP}$) parmi lesquels le problème 3DM des mariages à trois. Une référence très complète est [8].

1.3 Modèles de calcul probabilistes et interactifs

Soit le problème fonctionnel suivant : étant donné un nombre premier impair P trouver un non-résidu quadratique modulo P.

Exercice 8 *Montrer que ce problème est dans* $\mathbf{FNP}$.

Puisque $(P-1)/2$ résidus modulo P sont non-quadratiques il suffit de choisir un résidu non nul r au hasard et de voir si $r^{\frac{P-1}{2}}$ est différent de 1. On a une chance sur deux de réussir. Il s'agit donc d'un problème facile et un ordinateur muni d'un générateur aléatoire le résout sans difficultés. Il reste qu'une machine de

Turing déterministe ne dispose pas d'un tel générateur aléatoire. De plus, nul ne peut garantir un temps d'exécution puisqu'il est possible, quoique peu probable, d'enchaîner une suite très longue de tirages malchanceux. La probabilité d'echec décroit exponentiellement avec le temps d'exécution (le nombre d'essais).

Il est important de signaler que l'on ne connaît pas d'algorithme polynomial déterministe qui donne un non-résidu quadratique modulo P. On ne sait pas exhiber un sous-ensemble de taille polynomiale de $\mathbb{Z}/P\mathbb{Z}$ qui contienne à coup sûr un non-résidu quadratique.

Un exemple plus frappant est fourni par la primalité. On a vu que ce problème est dans **NP**. On sait pourtant le résoudre efficacement à l'aide du test dit de Miller-Rabin. Ce test est une variante du test de Fermat. Soit $P-1 = 2^k M$ avec M impair. Si P est premier, pour tout résidu non nul a on a, d'après le petit théorème de Fermat

$$0 = a^{P-1} - 1 = (a^M - 1)(a^M + 1)(a^{2M} + 1)(a^{4M} + 1)...(a^{2^{k-1}M} + 1)$$

dans le corps $\mathbb{Z}/P\mathbb{Z}$ si bien que l'un au moins des facteurs de droite est nul. Si tel n'est pas le cas on en déduit que P n'est pas premier et on dit que a est un témoin de non-primalité de P.

Un théorème de Monier [21, 18] montre que si P est composé alors au moins trois quarts des résidus non-nuls modulo P sont des témoins de non-primalité.

Cela conduit à définir la classe de complexité **RP** (comme *random polynomial*) formée des langages L tels qu'il existe une machine de Turing non-déterministe polynomiale en temps qui pour un mot x dans L accepte x au moins une fois sur deux (pour au moins une exécution possible sur deux) et qui rejette toujours un mot x qui n'est pas dans L.

On voit que l'ensemble des nombres composés forment un langage dans **RP**.

On dit qu'un langage est dans **coRP** si son complémentaire est dans **RP**.

En suivant des idées originales d'Atkin [4], Adleman et Huang [3] on montré que la primalité est aussi dans **RP** de sorte qu'elle est dans **RP** $\cap$ **coRP** . Cette dernière classe est parfois appelée **ZPP**.

Présentons une dernière classe probabiliste, la classe **BPP**. Un langage est dans **BPP** s'il existe une machine de Turing non-déterministe qui résout le problème de l'appartenance à ce langage avec une probabilité $\leq 1/3$ de donner une réponse fausse. On a **ZPP** $\subset$ **RP** $\subset$ **BPP**.

Très souvent, un problème dans la classe **ZPP** sera dit polynomial à Las Vegas. De même un problème dans la classe **RP** sera dit polynomial à Monte Carlo. On omet souvent de préciser que les algorithmes présentés ont recours à de l'aléa. Par exemple le problème de la factorisation des polynômes sur un corps fini est souvent considéré comme polynomial alors qu'il est seulement connu pour être dans **FRP**.

On a vu une première façon de formaliser la notion de réduction. C'est la réduction *one to one*. On dit que le problème de décision A admet une réduction

many to one au problème B si A est dans $\mathbf{P}^B$. On dit que A admet une réduction probabiliste *many to one* au problème B si A est dans $\mathbf{ZPP}^B$. Mêmes définitions pour les problèmes fonctionnels. On dit que deux problèmes sont équivalents si chacun se réduit à l'autre.

Exercice 9 *On considère le problème du calcul d'une racine carrée d'un résidu r modulo un nombre entier composé N. Montrer que ce problème est équivalent au problème de la factorisation des entiers pour les réductions polynomiales probabilistes many to one.*

Exercice 10 (Lenstra) *On dit qu'un groupe se prête au calcul si l'on sait représenter, comparer, multiplier, diviser des éléments en temps polynomial en le logarithme de la taille du groupe. Il est courant d'ajouter à ces conditions la possibilité de tirer au hasard un élément du groupe avec probabilité uniforme. On dit qu'on dispose d'une boîte noire pour le groupe si l'on n'en sait pas plus. Le mérite de cette notion est de fournir le cadre le plus large possible pour l'algorithmique des groupes. Un algorithme valide dans ce cadre est utile pour tous les groupes. On peut définir de même les anneaux, corps par boîtes noires. Étant donné un corps premier fini par boîte noire, on s'intéresse au problème de déterminer ce corps ce qui revient à déterminer sa caractéristique. Montrer que s'il existe un algorithme polynomial pour résoudre ce problème alors la factorisation d'entiers est dans* **ZPP** *(on admet que la primalité est dans* **ZPP** *).*

Exercice 11 *Soit N un entier et $G = \mathbb{Z}/N\mathbb{Z}$. Soient G_1 et G_2 deux sous-groupes de G qui ne sont pas connus mais pour lesquels on dispose d'un oracle donnant un élément aléatoire du groupe avec probabilité uniforme. On veut savoir si $G_1 \subset G_2$. Montrer que ce problème est dans* **BPP**.

Nous terminons en évoquant la classe **IP** des langages reconnus par un protocole interactif.

Définition 1 *Un protocole de preuve interactive est un protocole $\mathcal{P}$ entre deux machines de Turing A (comme Alice) et B (comme Bob). Alice est une machine déterministe et Bob est une machine non-déterministe. Alice est appelée le prouveur et Bob le vérifieur. Un mot x est transmis à Alice qui échange ensuite des messages de longueur polynomiale avec Bob. Après un nombre polynomial d'echanges, Bob répond OUI ou NON. Si w est l'aléa utilisé par Bob, la réponse de Bob au protocole est notée $\mathcal{P}(A,B)[x,w]$.*

Nous sommes en mesure de définir la classe **IP**.

Définition 2 *Un langage L appartient à la classe* **IP** *si et seulement s'il existe un protocle $\mathcal{P}$, un prouveur A déterministe dans* **EXPTIME** *et un vérifieur B polynomial non-déterministe tel que si un mot x est dans L alors pour tout w $\mathcal{P}(A,B)[x,w] = OUI$ et si x n'est pas dans L alors pour tout prouveur A' la probabilité que $\mathcal{P}(A',B)[x,w] = NON$ est au moins $1/2$.*

En somme, un langage est dans **IP** s'il existe un vérifieur polynomial et un prouveur très puissant capable de le convaincre de l'appartenance d'un mot au langage. On suppose aussi que tout prouveur malhonnête A' a peu de chances de tromper le vérifieur.

On a déjà rencontré dans l'exemple 1 le problème de l'isomorphisme de deux graphes. Ce problème est dans **NP** mais on ne connaît pas de preuve polynomiale que deux graphes ne sont pas isomorphes. En revanche on connaît des preuves interactives polynomiales du non-isomorphisme de deux graphes. Donc le non-isomorphisme de deux graphes est dans **IP**.

Exercice 12 *Soient $\mathcal{G}_1$ et $\mathcal{G}_2$ deux graphes. Le prouveur A veut convaincre B qu'ils sont non-isomorphes. Le vérifier B tire un bit b au hasard. Si $b = 0$ il forme deux graphes $\mathcal{U}_1$ et $\mathcal{U}_2$ isomorphes à $\mathcal{G}_1$ en appliquant deux permutations aléatoires aux sommets de $\mathcal{G}_1$. Si $b = 1$ il forme un graphe $\mathcal{U}_1$ isomorphe à $\mathcal{G}_1$ en appliquant une permutation aléatoire aux sommets de $\mathcal{G}_1$. Il forme ensuite un deuxième graphe $\mathcal{U}_2$ isomorphe à $\mathcal{G}_2$ en appliquant une permutation aléatoire aux sommets de $\mathcal{G}_2$.*

Ensuite B transmet $\mathcal{U}_1$ et $\mathcal{U}_2$ au prouveur A et lui demande s'ils sont isomorphes.

Montrer à l'aide de ce protocole que le non-isomorphisme de graphes est dans **IP**.

Lund, Fortnow, Karloff, Nisan, et Shamir ont montré dans [15, 22] que **IP** =**PSPACE**. La preuve repose sur des techniques d'interpolation de formules booléennes par des polynômes de faible degré en plusieurs variables.

Chapitre 2

Factorisation et logarithmes discrets

2.1 Introduction

On sait que tout nombre entier naturel s'écrit de façon unique comme produit de nombres premiers. Ce théorème est effectif. On connaît de nombreux algorithmes, plus ou moins efficaces, pour factoriser un entier naturel en produit de facteurs premiers. Aucun n'est polynomial. La difficulté de factoriser des entiers est au cœur de plusieurs protocoles cryptographiques dont le fameux RSA.

La problème dit du *logarithme discret* est assez voisin. Étant donnés un entier naturel N et deux résidus x et y modulo N, on cherche un entier ℓ tel que x^ℓ soit congru à y modulo N. C'est un problème supposé difficile. On ne connaît pas d'algorithme polynomial pour le résoudre. Les algorithmes connus sont proches des algorithmes de factorisation et ils sont d'une efficacité comparable.

Le problème du logarithme discret se généralise à tout groupe fini et même un peu au delà comme nous verrons. Il est lui aussi très utile en cryptographie.

2.2 Rappels sur les entiers

On s'intéresse maintenant à la complexité de quelques algorithmes pour la manipulation des entiers relatifs.

Exercice 13 (Addition, multiplication et division scolaires) *Montrer que la méthode enseignée à l'école primaire pour ajouter deux nombres entiers a une complexité linéaire. Peut-on concevoir un algorithme qui ajoute deux entiers de taille n en temps $o(n)$?*

Montrer que la méthode enseignée à l'école primaire pour multiplier deux nombres entiers a une complexité quadratique en la taille n des entiers. Même question pour la division euclidienne.

Un entier naturel est dit premier s'il n'admet pas d'autre diviseur que 1 et lui même.

Exercice 14 *En utilisant les propriétés de la division euclidienne, donner la liste de tous les sous-groupes de $\mathbb{Z}$.*

Si a et b sont deux entiers relatifs non nuls on note $pgcd(a,b)$ le générateur positif du sous-groupe de $\mathbb{Z}$ engendré par a et b.

Exercice 15 *Montrer que $pgcd(a,b)$ est le plus grand des diviseurs communs à a et b.*

On note $ppcm(a,b)$ l'unique générateur positif du sous-groupe de $\mathbb{Z}$ intersection du groupe engendré par a et du groupe engendré par b.

Exercice 16 *Montrer que $ppcm(a,b)$ est le plus petit des multiples positifs communs à a et b et que $ppcm(a,b) \times pgcd(a,b) = |ab|$.*

Exercice 17 *(Bezout-Bachet-Euclide-Gauss) Démontrer le théorème de Bezout-Bachet selon lequel si a et b sont deux entiers non nuls il existe des entiers λ et μ tels que $\lambda a + \mu b = pgcd(a,b)$.*

En déduire le théorème de Gauss selon lequel, si un nombre premier p divise ab alors il divise a ou b.

Montrer alors le théorème fondamental de l'arithmétique selon lequel tout entier naturel non nul est un produit de nombres premiers, de façon unique à permutation près.

Donner un algorithme polynomial pour le calcul de $pgcd(a,b)$, λ et μ.

De nombreux algorithmes pour multiplier rapidement deux nombres ont été découverts dans les années 60.

Nous décrivons brièvement une variante due à Knuth [11, 4.3.3] d'un algorithme de Karatsuba [10]. Cet algorithme exploite l'existence de morphismes naturels (dits *morphismes d'évaluation*) entre l'anneau $\mathbb{Z}[X]$ des polynômes à coefficients entiers et l'anneau $\mathbb{Z}$ lui même.

Soit $b > 1$ un entier appelé la base et considérons le morphisme ϵ_b d'anneaux de $\mathbb{Z}[X]$ dans $\mathbb{Z}$ qui à X associe b. Représenter un entier N en base b c'est trouver un polynôme P à coefficients entiers dans $[0, b-1]$ tel que $\epsilon_b(P) = P(b) = N$. Très souvent $b = 2$ ou 10. Le passage d'une base à une autre requiert en général de nombreuses divisions euclidiennes mais dans le cas particulier où la nouvelle base est une puissance de la première, il se fait sans calcul. En effet, posons $B = b^n$ où $n > 1$ est un entier. L'écriture B-cimale d'un entier N s'obtient à partir de l'écriture b-cimale en regroupant les b-cimales par paquets de n en commençant par la droite. Par exemple, si $b = 10$, $n = 2$ et $N = 12345$ alors l'écriture de N en base 100 est $N = 45 + 23 \times 100 + 1 \times 100^2$.

Remarque 2 *Les morphismes d'évaluation sont d'une grande utilité pour le calcul des polynômes. Par exemple, soient $\mu(X)$ et $\nu(X)$ deux polynômes de degré fixé r, à coefficients entiers relatifs. Le calcul de $\mu(X)\nu(X)$ par la méthode enseignée naguère au collège (et désormais en licence) requiert $(r+1)^2$ multiplications dans $\mathbb{Z}$ et à peu près autant d'additions. Si les coefficients sont de taille h (leur logarithme est borné par h) et si la méthode de multiplication dans $\mathbb{Z}$ requiert $T(h)$ opérations élémentaires pour multiplier deux nombres de taille h, alors le temps de calcul sera $(1+o(1))T(h)(r+1)^2$.*

Au lieu de celà, considérons les morphismes d'évaluation en 0, 1, ..., $2r$ et leur produit

$$\Pi = \prod_{0\leq i\leq 2r} \epsilon_i : \mathbb{Q}[X] \to \mathbb{Q}^{2r+1}.$$

La restriction de Π aux polynômes de degré inférieur ou égal à $2r$ est un isomorphisme de $\mathbb{Q}$-espaces vectoriels. La matrice de Π dans la base canonique est une matrice de Vandermonde. On a même des formules explicites pour son inverse qui résout le problème dit de l'interpolation des polynômes.

Supposant désormais que r est fixé, le produit $\mu(X)\nu(X)$ est obtenu en calculant d'abord les $\mu(i)$ pour $0 \leq i \leq 2r$, puis les $\nu(i)$, puis les $2r+1$ produits $\mu(i)\nu(i)$. On applique alors le vecteur $(\mu(i)\nu(i))_i$ à la matrice d'interpolation et on obtient les coefficients de $\mu(X)\nu(X)$. Le tout au prix de $2r+1$ multiplications entre nombres de taille $(1+o(1))h$, et moins de $8(r+1)^2$ additions entre des nombres de taille $(2+o(1))h$ et moins de $8(r+1)^2$ multiplications entre des constantes (dépendant de r) et des nombres de taille $(2+o(1))h$. Au total le temps de calcul est donc $(1+o(1))(2r+1)T(h)+O(h)$ soit $(1+o(1))(2r+1)T(h)$ pour r fixé, au lieu de $(1+o(1))(r+1)^2T(h)$ pour la méthode naïve.

Supposons donc qu'une base b est donnée (en fonction de la représentation des entiers dans le contexte ou la machine concernés) et fixons un entier $r > 1$. Pour multiplier deux entiers naturels M et N dont le nombre de b-cimales est borné par $n(r+1)$ on commence par les représenter en base $B = b^n$ ce qui revient à exhiber (sans aucun calcul) des polynômes $\mu(X)$ et $\nu(X)$ de degrés inférieurs ou égaux à r et à coefficients entiers dans $[0, B-1]$ tels que $\mu(B) = M$ et $\nu(B) = N$. Plutôt que de calculer MN on calcule le produit $\pi(X) = \mu(X)\nu(X)$ selon la procédure exposée plus haut. Le produit MN est obtenu en évaluant $\pi(X)$ en $B = b^n$ ce qui requiert $2r$ multiplications par une puissance de B (opération gratuite en base b) et $2r$ additions entre nombres de taille $2n(r+1)$. On fixe r et on applique récursivement la même méthode aux multiplications qui interviennent dans l'algorithme. Le temps de calcul $T(n(r+1))$ du produit de deux entiers de taille $(r+1)n$ est alors majoré par $(2r+1)T(n)$ plus une fonction affine en n soit

$$T((r+1)n) \leq (2r+1)T(n) + \alpha n + \beta.$$

Exercice 18 *En déduire que la méthode récursive décrite ci-dessus multiplie deux entiers de taille n en temps $O(n^{\frac{\log(2r+1)}{\log(r+1)}})$ pour r fixé. Ce qui montre le*

Théorème 1 *Pour tout $\gamma > 1$ il existe un algorithme qui multiplie deux entiers de taille n en temps $O(n^\gamma)$.*

Ainsi la multiplication de deux entiers n'est pas du tout de complexité quadratique. Elle est presque linéaire.

2.3 Rappels sur les congruences

Si N est un entier naturel, l'anneau $\mathbb{Z}/N\mathbb{Z}$ se prête bien au calcul. Les classes de congruences sont représentées par leur plus petit élément positif. On le trouve à l'aide d'une division euclidienne. Les opérations de base (comparaison, addition, mutiplication, inversion lorsqu'elle est possible) sont polynomiales en la taille de N.

De même, si $k = \mathbb{F}_p$ est un corps premier fini et $f(X)$ un polynôme irréductible de degré sur k, on peut traiter des congruences modulo $f(X)$ et construire ainsi un modèle calculatoire du corps fini à p^d éléments.

Cela suppose que l'on sache vérifier l'irreducibilité de $f(X)$ ou plus généralement factoriser un polynôme à coefficients dans $\mathbb{F}_p$.

Cela se fait en trois étapes.

On cherche d'abord à écrire f comme produit de polynômes sans facteurs carrés. On observe que si la décomposition (inconnue) en facteurs irréductibles de f est $f(X) = \prod_i u_i^{e_i}(X)$ alors le plus grand commun diviseur de f et de sa dérivée f' est $\prod_i u_i^{e_i-\kappa(e_i)}$ où $\kappa(e) = 0$ si p divise e et 1 sinon.

Exercice 19 *En déduire un algorithme polynomial déterministe pour factoriser un polynôme en produit de polynômes sans facteurs carrés.*

On se ramène ainsi au cas d'un polynôme sans facteurs carrés. On remarque alors que si f est un tel polynôme et k un entier, alors le plus grand commun diviseur de f et de $X^{p^k} - X$ est le produit de tous les facteurs irréductibles de f de degré divisant k. On dit qu'un polynôme est equanime s'il est sans facteurs carrés et si tous ses facteurs irréductibles ont le même degré. Si ce degré est k on dit que le polynôme est k-equanime.

Exercice 20 *En déduire un algorithme polynomial déterministe pour factoriser un polynôme sans facteurs carrés en produit de polynômes equanimes. Donner un algorithme polynomial déterministe pour tester l'irréducibilité d'un polynôme.*

Pour finir, on doit factoriser un polynôme k-equanime. On ne connaît pas d'algorithme polynomial déterministe qui résolve ce problème. Il existe cependant un algorithme polynomial probabiliste dû à Cantor et Zassenhaus. Soit donc f un polynôme k-equanime de degré $d = rk$ et soient f_1, ..., f_r ses facteurs. L'algèbre $\mathbb{F}_p[X]/f$ est isomorphe au produit des $\mathbb{F}_p[X]/f_i$ et la distribution uniforme sur le groupe des unités $(\mathbb{F}_p[X]/f)^*$ est donc le produit des distributions uniformes sur les $(\mathbb{F}_p[X]/f_i)^*$.

Supposons d'abord que la caractéristique p est impaire. Soit a un élément aléatoire (pour la distribution uniforme) de $\mathbb{F}_p[X]/f$. Si $pgcd(f, a)$ est un facteur non-trivial de f on est déjà bien avancé et on peut reprendre l'algorithme sur chacun des deux facteurs ainsi obtenus. Supposons donc que f et a sont premiers entre eux. Les résidus de a modulo les f_i sont des unités aléatoires indépendantes pour la distribution uniforme. Chacune est un carré avec probabilité $1/2$. Ainsi, le $pgcd(a(X)^{\frac{q-1}{2}} - 1, f(X))$ est un facteur non trivial avec probabilité $1 - 1/2^{r-1}$. On en déduit que la factorisation de polynômes sur $\mathbb{F}_p$ pour p impair est dans **BPP** ou plus exactement dans la version fonctionnelle de cette classe.

Exercice 21 *Dans le cas $p = 2$ tous les éléments de $\mathbb{F}_{2^k}$ sont des carrés et l'algorithme échoue. On observe alors que la moitié des éléments a une trace absolue nulle et l'autre moitié a une trace absolue égale à 1. En déduire un algorithme de factorisation des polynômes à coefficients dans $\mathbb{F}_2$.*

Montrer que le problème de la factorisation des polynômes sur un corps fini (donné comme quotient de $\mathbb{F}_p[X]$ par un polynôme irréductible) est polynomial (en la taille du corps et le degré du polynôme à factoriser) probabiliste.

On remarque enfin que les congruences modulo un polynôme irréductible $f(X)$ à coefficients dans le corps des rationnels permettent de représenter les corps de nombres. Cela nous conduit au problème de la factorisation des polynômes à coefficients rationnels. Une approche classique consiste (après élimination des facteurs carrés) à factoriser le polynôme f modulo un nombre premier puis à relever tous les facteurs (non nécessairement premiers) dans le corps des p-adiques à l'aide du lemme de Hensel. Comme on dispose d'une borne a priori sur la taille des coefficients des facteurs de f ([17]) on sait si un facteur sur les p-adiques provient d'un facteur sur les entiers. Il reste la difficulté combinatoire d'essayer de relever tous les facteurs de f sur les p-adiques. Si f a beaucoup de facteurs premiers modulo p alors l'algorithme est inefficace. Or il existe des polynômes irréductibles sur $\mathbb{Q}$ qui se décomposent en petits facteurs modulo tous les nombres premiers. Par exemple, le polynôme minimal d'une somme de racines carrées de nombre premiers distincts $\sum_p \sqrt{p}$. Ainsi, cet algorithme, bien que très efficace en pratique n'est pas polynomial. Lovasz et Lenstra on montré cependant dans [13] que la factorisation des polynômes à coefficients rationnels est dans **FP**. Ils ramènent ce problème à la recherche de vecteurs courts dans un réseau. Pour cette recherche, l'algorithme de Lovasz est alors utilisé.

2.4 Quelques algorithmes simples

Nous décrivons deux algorithmes très simples pour rechercher des facteurs de nombres entiers. Observons que la vérification de la primalité d'un nombre entier étant chose facile, la difficulté de factoriser réside dans la recherche de facteurs non triviaux pas nécessairement premiers. En effet, si les facteurs trouvés ne sont pas premiers, on peut leur appliquer à nouveau l'algorithme de factorisation.

Une première méthode consiste à calculer la division euclidienne de l'entier N à factoriser par les entiers $r = 2, 3, 5, 7, 9, 11, 13, 15$ etc.

Si on parvient à $r \geq \sqrt{N}$ sans trouver de facteur alors le nombre N est premier et l'algorithme s'arrête.

La complexité de cette méthode est en $O(N^{\frac{1}{2}})$ ce qui n'est pas très bon.

Une méthode élégante et beaucoup plus efficace (mais heuristique) est due à Pollard. On suppose pour simplifier que $N = pq$ avec p et q premiers distincts. On fixe un polynôme f à coefficients entiers (souvent $f(X) = X^2 - 2$) et on considère la suite itérée à valeur dans $\mathbb{Z}/N\mathbb{Z}$ définie par $x = x_0$ quelconque et $x_{k+1} = f(x_k) \pmod N$.

Comme $f(X)$ est un polynôme l'application induite $f_N : \mathbb{Z}/N\mathbb{Z} \to \mathbb{Z}/N\mathbb{Z}$ est un *produit* $f_N = f_p \times f_q$ des deux applications $f_p : \mathbb{Z}/p\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z}$ et $f_q : \mathbb{Z}/q\mathbb{Z} \to \mathbb{Z}/q\mathbb{Z}$.

On suppose que les applications f_p et f_q se comportent comme des applications aléatoires d'un ensemble fini dans lui-même. C'est une assomption un peu folle puisque $f = x^2 - 2$ est déterminée donc choisie aléatoirement dans un ensemble à un élément...

Exercice 22 *Soit E un ensemble fini de cardinal p et f une application de E dans E. L'ensemble $\mathcal{F}$ de ces applications est muni de la mesure uniforme. Soit (f, x_0) un couple formé d'une application et d'un élément de E. L'ensemble de ces couples est muni de la mesure uniforme. À un tel couple on associe la suite itérée $x_{k+1} = f(x_k)$ et deux entiers M et T. L'entier M est le plus petit entier positif tel que $(x_k)_k$ prends plusieurs fois la valeur x_M. Et l'entier T est le plus petit entier positif tel que $x_{k+T} = x_k$ pour tout k assez grand. On appelle T la période et M la prépériode. Montrer que la probabilité que $(x_k)_k$ ait une période et une prépériode données est*

$$P(M, T) = \frac{1}{p} \prod_{1 \leq k \leq M+T} \left(1 - \frac{k}{p}\right).$$

En déduire que la moyenne de T tend vers $\sqrt{\frac{\pi p}{8}}$ quand p tend vers l'infini. Même chose pour la moyenne de M.

On dispose donc d'une suite itérée produit $x_k \in \mathbb{Z}/N\mathbb{Z} = \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$ dont la composante en p (resp. q) a période et prépériode $O(\sqrt{p})$ (resp. $O(\sqrt{q})$).

On note M_p, T_p, M_q, T_q les périodes et prépériodes. Si k est assez grand on aura très vraissemblablement

$$pgcd(x_k - x_{k+T_p}, N) = p$$

ce qui donne un facteur non trivial de N. Bien sûr, cette formule n'est d'aucun secours puisqu'on ne connaît pas T_p. Mais elle implique que pour k assez grand et multiple de T_p

$$pgcd(x_k - x_{2k}, N) = p.$$

L'algorithme de Pollard consiste à calculer itérativement $x_k = f(x_{k-1})$ et $y_k = x_{2k} = f(f(y_{k-1}))$ et le pgcd ci-dessus pour $k = 0, 1, 2, ...$, jusqu'à trouver un facteur de N.

Heuristiquement cette méthode trouve un facteur p en temps $O(\sqrt{p})$ soit $\ll N^{\frac{1}{4}}$.

2.5 Introduction aux cribles

Dans cette section, nous introduisons des algorithmes de factorisation et de calcul de logarithmes discrets plus récents. Ils ont en commun d'utiliser la notion de nombre friable (ou lisse).

2.5.1 Entiers et polynômes lisses

On considère un anneau A qui est soit l'anneau des entiers naturels, soit l'anneau $\mathbb{F}_q[x]$ des polynômes sur un corps fini. On note $A^\times$ le semi-groupe des éléments non nuls de A. On dit qu'un élément de $A^\times$ est normalisé s'il est positif (si $A = \mathbb{Z}$) ou unitaire (si $A = \mathbb{F}_q[x]$).

Le semigroupe $A^\times$ est muni d'une fonction $t : A^\times \to \mathbb{R}$ appelée taille. Dans le cas où A est l'anneau des entiers naturels, $t(n) = \log(|n|)$ est la taille de l'entier n. Si A est l'anneau $\mathbb{F}_q[x]$ des polynômes en une indéterminée sur $\mathbb{F}_q$, la taille est le degré.

Dans ces deux cas, on constate que la taille est additive, c'est-à-dire que la taille d'un produit est la somme des tailles des facteurs.

On constate aussi que le nombre $V(t_0)$ d'éléments de taille inférieure à une taille t_0 donnée est exponentiel en t_0. C'est clair pour les entiers où $V(t_0) = 2\lfloor\exp(t_0)\rfloor$. C'est aussi clair pour les polynômes sur $\mathbb{F}_q$. Il y a en effet $V(t_0) = q^{t_0+1} - 1$ polynômes non nuls de degré inférieur ou égal à t_0. On pose $\chi = e$ si $A = \mathbb{Z}$ et $\chi = q$ si $A = \mathbb{F}_q[x]$. On a $V(t_0) = \chi^{t_0+O(1)}$.

Dans ces conditions, on qualifie d'*y-friable* ou *y-lisse* un élément produit d'éléments de taille inférieure ou égale à la taille de y.

Par exemple, $330 = 2\times3\times5\times11$ est 13-lisse et même 11-lisse dans $\mathbb{Z}$. De même dans $\mathbb{F}_{37}[X]$ le polynôme $X^5+X+1 = (X+27)(X+11)(X+20)(X^2+16X+13)$ est X^2+X-lisse mais on dira plutôt qu'il est 2-lisse en faisant référence au degré.

On étudie alors la proportion d'éléments lisses dans A.

Soient x et y deux éléments non nuls de A et montrons maintenant comment estimer grossièrement la proportion d'éléments y-lisses de taille inférieure à $t(x)$. L'argumentation que nous donnons est à la fois simple et générale mais elle est grossière.

Soit $n = \lfloor t(x)/t(y)\rfloor$ l'entier inférieur le plus proche de $t(x)/t(y)$ et prenons n éléments y_1, y_2,..., y_n irréductibles et normalisés dans A et de taille inférieure ou égale à $t(y)$. Alors le produit $y_1y_2...y_n$ est de taille inférieure ou égale à $t(x)$ et il est y-lisse.

On cherche alors à compter les éléments y-lisses ainsi obtenus. On se souvient que la proportion d'irréductibles parmi les éléments normalisés de taille inférieure ou égale à $t(y)$ est $\gg t(y)^{-1}$ (c'est le théorème des nombres premiers [24] pour les entiers naturels et [23, V.2.10] en caractéristique finie). Ainsi le nombre de n-uplets $(y_1, ..., y_n)$ est $\gg \chi^{O(n)}V(t(y))^n t(y)^{-n} \gg \chi^{n.t(y)-n\log_\chi t(y)+O(n)} \gg \chi^{t(x)-t(y)-n\log_\chi t(y)+O(n)} \gg V(t(x))\chi^{-t(y)+O(n)}t(y)^{-n}$. Mais comme la multiplication est commutative, on a au plus $n!$ permutations d'un même n-uplet qui donnent le même produit. Aussi, en première approximation, on a fabriqué $\gg V(t(x))\chi^{-t(y)+O(n)}t(y)^{-n}(n!)^{-1}$ éléments y-lisses de taille $\leq n.t(y) \leq t(x)$. Comme il y a $V(t(x))$ éléments de taille $t(x)$ on en déduit

Lemme 1 *La probabilité pour qu'un élément de taille t_x soit y-lisse est*

$$\chi^{-t_y+O(n)}t_y^{-n}(n!)^{-1}$$

avec $t_y = t(y)$ et $n = \lfloor t_x/t_y\rfloor$, quand n, t_x et t_y tendent vers l'infini.

Observons que pour les entier naturels, cette probabilité est $n^{-n+o(n)}$ ce qui est proche de notre estimation. Voir [6].

2.5.2 Crible linéaire et logarithmes discrets

On se donne un entier premier positif N et on cherche à calculer des logarithmes discrets modulo N.

On pose $G = (\mathbb{Z}/N\mathbb{Z})^*$. On a $|G| = \phi(N) = N-1$ et si g est un générateur de G on note $\exp_g : \mathbb{Z}/(N-1)\mathbb{Z} \to G$ l'application d'exponentiation $x \mapsto g^x$. On note $\log_g$ l'application réciproque. C'est le logarithme discret de base g.

Pour tout entier non nul x et pour tout nombre premier p on note $e(x,p)$ le plus grand entier k tel que p^k divise x. Tout entier positif x s'écrit

$$x = \prod_p p^{e(x,p)}.$$

On fixe un entier positif B et on cherche des petits nombres entiers $r = 1, 2, ...$ tels que r et $r + N$ soient B-lisses, c'est à dire

$$r = \prod_{p<B} p^{e(r,p)} \text{ et } r + N = \prod_{p<B} p^{e(r+N,p)}.$$

Pour chacun d'entre eux on obtient une congruence

$$\prod_{p<B} p^{e(r,p)-e(r+N,p)} = 1 \pmod{N}.$$

En posant $\delta(r,p) = e(r,p) - e(r+N,p)$ cette congruence peut se traduire en termes de logarithmes discrets

$$\sum_{p<B} \delta(r,p)\log(p) = 0 \pmod{\varphi(N)}.$$

Ici nous ne précisons pas la base du logarithme discret car l'identité est vraie indépendamment de cette base. Lorsque on a collecté suffisamment de relations de ce type on peut exprimer tous les logarithmes $\log(p)$ pour $p < B$ en fonction de l'un d'entre eux. C'est la *première phase* ou *phase homogène* du calcul.

Si l'on souhaite calculer $\log_b(c)$ pour $b, c \in (\mathbb{Z}/N\mathbb{Z})^*$ il suffit de trouver un entier $b + kN$ qui soit B-lisse (on essaye $k = 1, 2, ...$). Cela permet d'exprimer $\log(b)$ en fonction des $\log(p)$ pour $p < B$. On fait de même pour c. On finit en calculant

$$\log_b(c) = \log(c)/\log(b).$$

C'est la *deuxième phase*.

Pour majorer le temps de calcul de cet algorithme, prenons $B = \lceil \exp(\sqrt{\log N}) \rceil$, la probabilité que r soit friable est très supérieure à la probabilité que $r + N$ soit friable. On la négligera donc. On suppose en outre que $r+N$ se comporte comme un résidu aléatoire uniformément distribué (ce qui n'est pas le cas) et on lui applique abusivement le lemme 1. Ainsi, la probabilité P que r et $r + N$ soient friables est $\gg \chi^{(\log N)^{\frac{1}{2}+o(1)}}$. Le nombre d'essais pour trouver une bonne valeur de r est donc l'inverse de cette probabilité. Et le nombre de bonnes valeurs à collecter est de l'ordre de B. Le temps T d'exécution de l'algorithme est donc $\ll B/P$ soit

$$T = \chi^{(\log N)^{\frac{1}{2}+o(1)}}$$

ce qui est bien subexponentiel.

On verra dans la section 2.6 que cet algorithme a pour complexité heuristique

$$T = \exp\left((1 + o(1))\sqrt{2\log(N)\log\log(N)}\right).$$

Il est possible de proposer une variante de cet algorithme, moins efficace en pratique mais de complexité prouvable [20].

2.5.3 Logarithme discret dans $\mathbb{F}_{p^d}^*$

On considère un polynôme irréductible $F(X)$ de degré d à coefficients dans $\mathbb{F}_p$ et on construit le corps $\mathbb{F}_q = \mathbb{F}_p[X]/F(X)$ avec $q = p^d$.

On suppose que p est fixé et que d tend vers l'infini. Pour calculer des logarithmes discrets dans $\mathbb{F}_q^*$ on recherche des polynômes $r(X)$ à coefficients dans $\mathbb{F}_p$ tels que $r(X)$ et $r(X) + F(X)$ soient lisses c'est-à-dire se décomposent en produits de facteurs de degrés inférieurs à un entier b donné. On utilise les estimations grossières de la densité de polynômes lisses données au paragraphe 2.5.1.

Dans le cas opposé où d est fixé et p tend vers l'infini, on se retrouve dans une situation beaucoup plus proche de celle du paragraphe précédent. On remplace simplement $\mathbb{Z}$ par un anneau d'entiers $\mathcal{O}$ (e.g. cyclotomiques) de corps des fractions $\mathbb{K}$, muni d'un idéal $\mathfrak{p}$ de corps résiduel $\mathbb{F}_q$ et on considère des éléments lisses dans cet anneau. Comme cet anneau n'est pas principal, on dit qu'un élément est lisse si sa norme est lisse dans $\mathbb{Q}$. On utilise un système de générateurs du groupe des éléments lisses de $\mathbb{K}^*$. Peu importe qu'il y ait des relations.

Lorsque p et k croissent tous les deux, il faut choisir de travailler avec des polynômes lisses ou des entiers lisses [2] en fonction des tailles respectives de p et d.

2.5.4 Le crible linéaire et le crible quadratique pour factoriser

Factoriser un entier N, c'est le représenter par la forme quadratique de rang 2,

$$Q(x, y) = x.y.$$

Dans une base orthogonale cette forme s'écrit $Q(X, Y) = X^2 - Y^2$. Factoriser c'est donc représenter N ou un multiple de N comme la différence de deux carrés.

Représenter N avec la forme $X^2 - Y^2$ c'est aussi représenter 0 avec la forme de rang trois

$$Q(X, Y, Z) = X^2 - Y^2 - NZ^2.$$

On sait que plus généralement, la recherche de solutions rationnelles à

$$Q(X, Y, Z) = aX^2 + bY^2 + cZ^2$$

avec $abc \neq 0$ est polynomialement équivalente à la recherche de trois entiers r_1, r_2 et r_3 tels que $r_1^2 = -bc \pmod{a}$, $r_2^2 = -ca \pmod{b}$, $r_3^2 = -ab \pmod{c}$,

voir [12]. Or on ne sait résoudre ce dernier problème que par la décomposition de a, b, c en facteurs premiers.

Pour trouver des solutions non triviales à la congruence

$$X^2 - Y^2 \equiv 0 \pmod{N}$$

on cherche des congruences entre nombres lisses à un carré près, c'est-à-dire des congruences de la forme

$$\prod_i p_i = x^2 \pmod{N}$$

où x est un entier non nul et les p_i sont des nombres premiers plus petits qu'une borne B donnée. Une fois collectées de nombreuses relations telles que celle ci-dessus, on peut, par élimination linéaire sur le corps à deux éléments, obtenir une congruence entre deux carrés.

Le plus simple de ces algorithmes est appelé crible linéaire :

Algorithme 1 *Pour factoriser N, choisir un entier B puis chercher des entiers r tels que r et $r+N$ soient B-lisses. Par exemple on prend $r = 0, 1, 2, ..., C$ et on garde les valeurs de r telles que $r(r+N)$ soit B-lisse. On a alors des congruences modulo N et lorsqu'on a rassemblé plus de $\pi(B)$ telles congruences, on peut obtenir par élimination une congruence modulo N entre deux carrés et donc une chance sur deux de factoriser N.*

On peut se convaincre que cet algorithme est subexponentiel, soit par les heuristiques élémentaires du paragraphe 2.5.2, soit par un calcul plus précis comme nous verrons dans la section 2.6. Cet algorithme n'est pourtant pas très efficace, surtout parce que les entiers $N+r$ ont peu de chances d'être lisses pour N grand. Il présente l'avantage d'admettre une variante prouvable.

Nous décrivons maintenant le crible quadratique. Comme le précédent, cet algorithme est dû à Carl Pomerance. Nous nous contentons de l'illustrer sur un exemple.

Soit

$$N = 21311 = 101.211$$

le nombre à factoriser. On choisit un entier m proche de la racine carrée de N

$$m = \lfloor N^{1/2} \rceil = 146.$$

On forme des congruences modulo N en observant que pour tout entier a,

$$(m+a)^2 \equiv (m^2 - N) + a^2 + 2am \pmod{N} = 5 + a^2 + 292a \pmod{21311},$$

où l'on note que $m^2 - N$ est de l'ordre de $\sqrt{N}$.

On se donne une borne $B = 13$ et l'on cherche de petits entiers a tels que $5 + a^2 + 292a$ soit B-lisse. Par exemple pour a compris entre -60 et 60 on trouve

a	$5 + 292a + a^2$
-27	$-2.5^2.11.13$
-5	$-2.5.11.13$
-1	$-2.11.13$
0	5
60	$5^3.13^2$

On porte dans une matrice la parité des valuations :

	-1	2	5	11	13
-27	1	1	0	1	1
-5	1	1	1	1	1
-1	1	1	0	1	1
0	0	0	1	0	0
60	0	0	1	0	0

On forme des carrés à partir des vecteurs du noyau de la transposée de cette matrice. Ce noyau est formé des trois lignes de la matrice suivante

-27	-5	-1	0	60
1	0	1	0	0
1	1	0	1	0
1	1	0	0	1

La première ligne du tableau donne la congruence

$$(2.5.11.13)^2 \equiv (146-27)^2.(146-1)^2 \pmod{21311}.$$

On calcule alors le plus grand diviseur commun de $2.5.11.13 - (146-27).(146-1) = -15825$ et de 21311. On trouve le facteur non trivial $p = 211$ de $N = 21311$ et son cofacteur 101. Un test de primalité prouve aisément que p et q sont premiers.

2.5.5 Le crible quadratique pour le logarithme discret

Il est naturel d'adapter le crible quadratique au calcul de logarithmes discrets. Cela se fait de plusieurs façons différentes. Nous présentons la plus simple.

Soit donc N le nombre premier et $G = (\mathbb{Z}/N\mathbb{Z})^*$ le groupe concernés. Soit $m = \lceil N^{1/2} \rceil$ et $\ell = N - m^2$ et $F(X) = X^2 + \ell$ un polynôme de degré 2 et

de hauteur $\sqrt{N}$ tel que $F(m) = N$. Soit $\mathcal{O}$ l'anneau $\mathbb{Z}[X]/F$ et $\mathfrak{p}$ l'idéal de $\mathcal{O}$ engendré par $X - m$. C'est un idéal premier de degré 1 au dessus de N. On cherche des entiers algébriques $R \in \mathcal{O}$ de la forme $X + r$ avec $r = 1, 2, 3, ...$ tels que $r + m$ et $r^2 + \ell$ soient lisses. En effet, si tel est le cas, la congruence

$$X + r = R = R - (X - m) = r + m \pmod{\mathfrak{p}}$$

entre deux nombres lisses (l'un $X + r$ dans $\mathcal{O}$ et l'autre $r + m$ dans $\mathbb{Z}$) se traite comme dans le paragraphe 2.5.3.

Dans le cas où ℓ est petit (N proche d'un carré), cet algorithme est plus efficace que le crible linéaire car les nombres impliqués sont de taille $N^{1/2}$. Cependant, on ne sait pas davantage le prouver. L'amélioration est donc seulement heuristique.

Une condition moins restrictive pour N que d'être proche d'un carré est d'être la somme $N = m^2 + \ell^2$ de deux carrés. On considère alors l'idéal premier de $\mathbb{Z}[i]$ engendré par $m + i\ell$. C'est un idéal premier de norme N. On cherche alors de couple d'entiers (a, b) tels que $a + ib$ et $an - mb$ soient lisses. Cela revient à demander que $(a^2 + b^2)(an - bm)$ soit lisse. Ce dernier nombre est de l'ordre de $\sqrt{N}$.

Cette méthode souffre de nombreux aménagements. Mais plutôt que de les égrener nous en venons au crible algébrique qui les résume.

2.5.6 Le crible algébrique, présentation générale

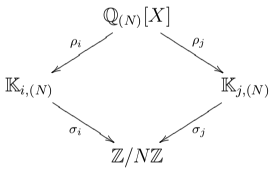
Le crible algébrique est un algorithme inventé par Pollard et Lenstra. Cette section vise à rassembler quelques notions de base relatives au crible algébrique et à les illustrer sur un exemple. Les frontières de cet algorithme ne sont pas nettes. De nombreuses variantes existent pour chacune des étapes qui le constituent. Nous donnons une présentation aussi générale que possible du centre de cet algorithme. Nous enrichirons cette description dans le paragraphe suivant.

On appelle toujours N l'entier à factoriser et m un entier auxiliaire qu'il conviendra de préciser plus tard.

Pour tout corps de nombres $\mathbb{K}$ on note $\mathcal{O}_{\mathbb{K}}$ son anneau des entiers et $\mathbb{K}_{(N)}$ l'anneau local en N c'est-à-dire l'ensemble des éléments de $\mathbb{K}$ dont le dénominateur est premier à N.

Soient $f_i(X)$ pour $1 \leq i \leq I$ des polynômes irréductibles unitaires à coefficients entiers tels que $f_i(m) = N$. On suppose que le degré de $f_i(X)$ est inférieur à tous les facteurs premiers de N. La condition $f_i(m) = N$ se traduit alors algébriquement par l'égalité de tous les idéaux $(X - m, f_i(X))$ et de $(X - m, N)$ dans $\mathbb{Q}_{(N)}[X]$.

Appelons $\mathbb{K}_i = \mathbb{Q}[X]/f_i(X)$ le corps de nombres associé à f_i. L'égalité entre idéaux ci-dessus implique la commutativité du diagramme suivant pour tout couple (i, j) d'entiers compris entre 1 et I.



Les flèches ρ_i sont des quotients par les polynômes f_i et les σ_i sont des substitutions de X par m.

Voici comment le crible algébrique utilise le diagramme ci-dessus. Observons d'abord que les trois étages du diagramme sont de natures arithmétiques très différentes.

Dans $\mathbb{Q}[X]$, la plupart des éléments sont irréductibles et très peu sont inversibles. Dans $\mathbb{Z}/n\mathbb{Z}$ c'est l'inverse. Il y a une majorité d'éléments inversibles et très peu d'éléments irréductibles puisqu'ils correspondent aux facteurs de n. Les corps $\mathbb{K}_i$ sont les seuls êtres intermédiaires que l'on puisse intercaler entre les deux précédents.

Prenons maintenant un élément $A(X)$ de $\mathbb{Q}[X]$. Pour tout i, on dit que $\rho_i(A) = A(X) \bmod f_i(X)$ est lisse s'il se factorise dans le groupe multiplicatif $\mathbb{K}_i^*$ en

$$A(X) \bmod f_i(X) = \prod_k \mathfrak{p}_{i,k}^{e_{i,k}}$$

où les $\mathfrak{p}_{i,k}$ sont une base de lissité, c'est-à-dire un système de générateurs d'un sous-groupe de $\mathbb{K}_i^*$ que l'on note $\mathbb{K}_{i,s}$. On choisit en général le groupe formé des éléments dont la norme est un rationnel B-lisse, c'est-à-dire divisible uniquement par des nombres premiers plus petits que B. Si $\mathcal{O}_{\mathbb{K}_i}$ est principal, les $\mathfrak{p}_{i,k}$ sont des générateurs de petits idéaux premiers et des unités fondamentales. On a alors

$$\sigma_i(\rho_i(A(X))) = A(m) \pmod{N} = \prod_k \sigma_i(\mathfrak{p}_{i,k})^{e_{i,k}}$$

Si $\rho_i(A)$ et $\rho_j(A)$ sont lisses on obtient une congruence modulo N

$$A(m) \pmod{N} = \prod_k \sigma_i(\mathfrak{p}_{i,k})^{e_{i,k}} = \prod_k \sigma_j(\mathfrak{p}_{j,k})^{e_{j,k}},$$

soit encore

$$\prod_k \sigma_i(\mathfrak{p}_{i,k})^{e_{i,k}} \prod_k \sigma_j(\mathfrak{p}_{j,k})^{-e_{j,k}} = 1 \pmod{N}.$$

On se fixe une base de lissité dans chacun des corps $\mathbb{K}_i$. Si pour un polynôme A donné et deux entiers $i \neq j$, $\rho_i(A(X))$ et $\rho_j(A(X))$ sont lisses on obtient une congruence utile comme ci-dessus. Lorsque le nombre de ces congruences excède

le cardinal de la réunion de toutes les bases de lissité, on obtient une congruence entre 2 carrés modulo n et donc une chance de factoriser n pour chaque relation excédentaire.

En général, les $\mathcal{O}_{\mathbb{K}_i}$ ne sont pas factoriels, aussi on adopte un point de vue dual. On sait que $\mathbb{K}_{i,s}$ est un groupe de type fini engendré par à peu près $\pi(B)$ générateurs. Ces générateurs ne sont pas canoniques et on ne sait pas les calculer. En revanche on sait que le groupe des éléments lisses modulo les carrés $\mathbb{K}_{i,s}/{\mathbb{K}_{i,s}}^2$ est un groupe commutatif d'exposant deux et de type fini, donc un espace vectoriel sur $\mathbb{Z}/2\mathbb{Z}$ de dimension voisine de $\pi(B)$. On remplace chaque idéal par la valuation associée ou plutôt le résidu modulo deux de cette valuation. On obtient ainsi un certain nombre de formes linéaires de $\mathbb{K}_{i,s}/{\mathbb{K}_{i,s}}^2$ dans $\mathbb{Z}/2\mathbb{Z}$. On ajoute à ces formes quelques caractères construits à partir de résidus quadratiques. Ces derniers, en nombre suffisant, servent à tuer l'obstruction provenant du groupe de classe et du groupe des unités. On suppose que la collection des valuations et des caractères engendre le dual de $\mathbb{K}_{i,s}/{\mathbb{K}_{i,s}}^2$. Autrement dit, si toutes ces valuations et caractères s'annulent en un nombre $a \in \mathbb{K}_{i,s}$, alors ce nombre est un carré.

Une relation élémentaire est alors définie par un polynôme $A(X)$ et deux entiers i et j tels que $\rho_i(X)$ et $\rho_j(X)$ soient lisses chacun dans son corps. Une combinaison de relations ayant le même i et le même j donne un produit $\Pi = \Pi_k A_k(X)$ tel que $C_i = \rho_i(\Pi)$ et $C_j = \rho_j(\Pi)$ soient des carrés dans $\mathbb{K}_i$ et $\mathbb{K}_j$ respectivement. Il reste à calculer les racines carrées R_i et R_j telles que $R_i^2 = C_i$ et $R_j^2 = C_j$. C'est un problème en soi que nous n'aborderons pas dans ce rapport. La relation s'écrit alors

$$\sigma_i(R_i)^2 = \sigma_j(R_j)^2 \pmod N.$$

Remarque : En général, on choisit une unique polynôme f tel que $f(m) = N$ et on définit le corps de nombre $\mathbb{K} = \mathbb{Q}[X]/f$ et les quatre morphismes $\sigma_{alg} : \mathbb{K}_{(N)} \to \mathbb{Z}/N\mathbb{Z}$, $\sigma_{rat} : \mathbb{Q}_{(N)} :\to \mathbb{Z}/N\mathbb{Z}$, $\rho_{rat} : \mathbb{Q}_{(N)}[X] \to \mathbb{Q}_{(N)}$, et $\rho_{alg} : \mathbb{Q}_{(N)}[X] \to \mathbb{K}_{(N)}$ par $\sigma_{rat}(N) = 0$, $\sigma_{alg}(X \bmod f) = m$, $\rho_{rat}(X) = m$, $\rho_{alg}(X) = X \bmod f$. Ces applications forment un diagramme commutatif,

$$\sigma_{alg}\rho_{alg} = \sigma_{rat}\rho_{rat}.$$

2.5.7 Un exemple de crible algébrique général

Soit toujours

$$N = 21311 = 101.211$$

le nombre à factoriser. Posons

$$m = \lfloor N^{1/2} \rceil = 146$$

et écrivons n en base m

$$N = 146^2 - 5 = f(N)$$

avec

$$f(X) = X^2 - 5.$$

Le polynôme $f(X)$ étant irréductible, on définit le corps de nombres

$$\mathbb{K} = \mathbb{Q}[X]/f(X)$$

qui est un corps quadratique réel. Donc $d = 2$. On se donne une borne de lissité $B = 20$ et on dit un nombre premier petit s'il est inférieur à B. On note $\mathcal{O}_{\mathbb{K}}$ l'anneau des entiers de $\mathbb{K}$ et on procède alors à l'évaluation de trois obstructions fondamentales que nous noterons o_u, o_c et o_e et que nous allons définir sur cet exemple.

L'obstruction o_u est le rang de $\mathcal{U}/2\mathcal{U}$ où $\mathcal{U}$ est le groupe des unités de $\mathcal{O}_{\mathbb{K}}$. Si on note r_1 le nombre de racines réelles de $f(X)$, et $2r_2$ le nombre des racines imaginaires, et r l'ordre du groupe des racines de l'unité, c'est-à-dire le plus grand entier r tel que $X^r - 1$ ait une racine dans $\mathbb{K}$, alors on a

$$o_u = r_1 + r_2 - 1 + \pi(r)$$

où π est la fonction parité qui vaut 0 si r est impair et 1 sinon.

L'obstruction o_c est le 2-rang du groupe de classe $\mathcal{H}_{\mathbb{K}}$. On sait d'après [6] que

$$|\mathcal{H}_{\mathbb{K}}| \leq M.\frac{(d - 1 + \log M)^{d-1}}{(d-1)!}$$

où

$$M = (d!/d^d)(4/\pi)^{r_2}\sqrt{|\,disc(\mathbb{K})|}.$$

Donc o_c est majoré par le logarithme en base deux de cette expression.

L'obstruction o_e est le nombre de petits idéaux de $\mathcal{O}_{\mathbb{K}}$ pour lesquels on ne sait pas facilement construire la valuation associée. En général ce sont ceux qui divisent l'index θ de X dans $\mathbb{K}$. On rappelle que

$$disc(f) = \theta^2\, disc(\mathbb{K})$$

si bien que o_e est inférieur à d fois le nombre de petits nombres premiers p tels que p^2 divise $disc(f)$.

Dans notre cas le groupe des unités est de la forme $\{-1, 1\} \times \langle u \rangle$ avec u l'unité fondamentale du corps quadratique. On a

$$o_u = 2 + 0 - 1 + 1 = 2.$$

Par ailleurs on sait que $\mathcal{O}_{\mathbb{K}} = \mathbb{Z}[\frac{1+X}{2}]$ est principal. On a donc $o_c = 0$, $\theta = 2$ et on sait que 2 est inerte dans $\mathcal{O}_{\mathbb{K}}$ si bien que la valuation associée se construit aisément : c'est la valuation de 2 dans la norme, divisée par deux. On a donc $o_e = 0$. Au total

$$o = o_u + o_c + o_e = 2.$$

On doit donc, d'après Adleman [1], chercher $o = 2$ caractères quadratiques indépendants. On note $\mathbb{K}_s$ le groupe des éléments inversibles et lisses de $\mathbb{K}$. On rappelle qu'un caractère est une forme linéaire de $\mathbb{K}_s$ dans $\mathbb{Z}/2\mathbb{Z}$. Ces formes se construisent de deux manières. Soit en regardant le signe des éléments du corps dans les r_1 plongements réels, soit en regardant si les résidus de ces éléments modulo des nombres premiers non lisses sont des carrés. Dans notre cas nous choisirons les deux caractères à l'infini c'est-à-dire les deux fonctions χ_1 et χ_2 définies par

$$\chi_1(a + bX) = signe(a + b\sqrt{5}),$$

et

$$\chi_2(a + bX) = signe(a - b\sqrt{5}).$$

Venons en maintenant à la détermination de la base de lissité. Elle se compose d'une base algébrique et d'une base rationnelle. La base rationnelle comprend -1, c'est-à-dire le signe et les nombres premiers inférieurs à $B = 20$ soit

$$\mathcal{B}_{rat} = \{-1, 2, 3, 5, 7, 11, 13, 17, 19\}.$$

Pour construire la base algébrique on décompose les petits nombres premiers impairs p tels que p^2 ne divise pas $disc(f)$. Pour cela il suffit de chercher les racines de $f(X)$ modulo p. Pour chaque racine c on définit un idéal (p, c). On ajoute les caractères à la base ainsi obtenue. On ajoute aussi l'idéal 2.

Ici on trouve

$$\mathcal{B}_{alg} = \{\chi_1, \chi_2, 2, (5, 0), (11, 7), (11, 4), (19, 9), (19, 10)\}.$$

À tout élément $a + bX$ de $\mathcal{O}_{\mathbb{K}}$ on associe sa norme

$$\mathcal{N}(a + bX) = \mathcal{N}(a, b) = f(-a/b)b^d.$$

Noter que la norme d'un nombre $A(X)$ mod $f(X)$ se calcule comme le résultant de A et de f.

Pour a et b premiers entre eux, on sait que $a + bX$ a un facteur commun avec p si et seulement si $\mathcal{N}(a, b)$ est divisible par p. Si p ne divise pas θ, ce qui est assuré si p^2 ne divise pas $disc(f)$, alors il existe un unique idéal premier (p, c) tel

que $a + bc \equiv 0 \pmod{p}$ et la multiplicité de (p, c) dans $a + bX$ est la multiplicité de p dans $\mathcal{N}(a, b)$.

Il reste maintenant à fixer une borne C_a et une borne C_b et à chercher des paires (a, b) telles que $-C_a \leq a \leq C_a$, $1 \leq b \leq C_b$, a et b sans facteur commun, et $(a + bm).\mathcal{N}(a, b)$ lisse. Pour $b = 0$ et a premier lisse, on obtient des relations dites gratuites : Lorsqu'un petit nombre premier p se décompose dans $\mathbb{K}$, on écrit $(p) = \mathfrak{p}_1\mathfrak{p}_2$ et on obtient une ligne dans la matrice en plaçant 1 dans la colonne de p, $\mathfrak{p}_1$ et $\mathfrak{p}_2$. Il faut aussi calculer les valeurs des caractères en p.

Ici nous prendrons

$$C_a = 40 \text{ et } C_b = 20.$$

Grâce à un crible on trouve 14 paires (a, b) plus 4 relations gratuites soit 18 relations pour 17 inconnues dans la base. Comme dans le cas du crible quadratique, nous plaçons dans une matrice les résidus modulo 2 des valuations et les logarithmes à base -1 des caractères (avec $\log_{-1}(-1) = 1$ et $\log_{-1}(1) = 0$). Les lignes de la matrice correspondent aux relations et ses colonnes aux inconnues.

-1	2	3	5	7	11	13	2_1	5_0	11_7	11_4	19_9	19_{10}	χ_1	χ_2	
0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	$(-3, 1)$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	$(-2, 1)$
0	1	0	0	1	0	1	0	0	0	0	0	0	0	1	$(-2, 5)$
0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	$(1, 1)$
0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	$(4, 1)$
0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	$(5, 2)$
0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	$(10, 3)$
0	1	0	1	1	0	0	0	0	0	0	0	0	0	1	$(38, 17)$
0	0	0	0	0	0	1	0	0	0	1	0	1	0	1	$(-14, 9)$
0	0	1	0	0	0	1	0	1	0	0	1	0	0	0	$(10, 1)$
0	0	0	0	1	0	0	0	1	0	0	0	1	0	1	$(15, 8)$
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	$(21, 4)$
0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	$(22, 13)$
0	0	0	0	1	0	0	1	0	0	1	1	0	0	0	$(29, 1)$
0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	$(2, 0)$
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	$(5, 0)$
0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	$(11, 0)$

Noter que les nombres premiers 17 et 19 n'ont pas été utiles dans la base rationnelle puisque on trouve assez de relations 16-lisses à gauche. En revanche, on a besoin de 19 à droite car les nombres premiers 13 et 17 sont inertes dans le corps

choisi. Ceci montre que notre corps de nombres n'est pas très bon. En général, il faut préférer un polynôme $f(X)$ avec beaucoup de racines modulo les petits nombres premiers.

Nous appelons $\mathcal{S}$ la matrice ci-dessus et nous calculons maintenant le noyau de la transposée de $\mathcal{S}$. Ici la méthode de Gauss est recommandée. Pour de plus grandes matrices il vaut mieux exploiter le caractère lacunaire de $\mathcal{S}$.

Nous choisissons un vecteur v_1, dans ce noyaux de dimension 5.

$$v_1 = [0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0].$$

Ce vecteur correspond au polynôme

$$P_1 = (-2 + 5X)(10 + 3X)(38 + 17X)(-14 + 9X)(15 + 8X)(21 + 4X).$$

On lui associe l'entier rationnel

$$\rho_{rat}(P_1) = (-2 + 5m)(10 + 3m)(38 + 17m)(-14 + 9m)(15 + 8m)(21 + 4m),$$

et l'entier algébrique

$$\rho_{alg}(P_1) = (-2+5X)(10+3X)(38+17X)(-14+9X)(15+8X)(21+4X) \bmod f(X).$$

Chacun de ces deux entiers est lisse et c'est même un carré. Il reste à calculer les racines carrées. Du coté rationnel, il suffit de décomposer chacun des termes du produit. Ce travail a déjà été fait au moment de construire $\mathcal{S}$. On obtient

$$\rho_{rat}(P_1) = 2^{14}.3^2.5^4.7^4.11^2.13^4 = (2^7.3.5^2.7^2.11.13^2)^2.$$

Du coté algébrique, on ne connaît pas la décomposition des facteurs et d'ailleurs, elle n'est pas forcément unique puisque l'anneau $\mathcal{O}_{\mathbb{K}}$ peut ne pas être factoriel. On doit donc calculer une racine de $\rho_{alg}(P_1)$ par une méthode *ad hoc*. Nous n'abordons pas ce problème ici. On trouve

$$\rho_{alg}(P_1) = (4180 + 1881X)^2 \pmod{f(X)}.$$

On a donc

$$\sigma_{rat}(\rho_{rat}(P_1)) = (2^7.3.5^2.7^2.11.13^2)^2 \bmod n = 19337^2 \bmod N,$$

et

$$\sigma_{alg}(\rho_{alg}(P_1)) = (4180 + 1881m)^2 \bmod n = 1763^2 \bmod N.$$

Or on sait que

$$\sigma_{rat}(\rho_{rat}(P_1)) = \sigma_{alg}(\rho_{alg}(P_1)),$$

donc

$$19337^2 = 1763^2 \bmod 21311.$$

On calcule alors

$$gcd(19337 - 1763, 21311) = 101,$$

ce qui donne un facteur non trivial de N.

2.6 Éléments pour l'analyse des cribles

Dans cette section nous présentons les outils fondamentaux de l'analyse des algorithmes subexponentiels.

On définit tout d'abord la quantité suivante pour $x, \nu, \lambda \in \mathbb{R}$ avec $x > e$, $\lambda \neq 0$, $0 \leq \nu \leq 1$

$$L_x[\nu, \lambda] = exp(\lambda(\log x)^{\nu}(\log\log x)^{1-\nu}).$$

Comme on utilisera ces fonctions pour des estimations asymptotiques, x tendant vers l'infini, on convient d'écrire $L_x[\nu, \lambda]$ pour $L_x[\nu, \lambda + o(1)]$ et $L_x[\nu]$ pour $L_x[\nu, \lambda]$ et $\lambda \neq 0$. Noter que $L_x[\nu]$ est défini à une exponentiation près (positive ou négative). L'échelle des $L_x[\nu, \lambda]$ est assez grossière puisque on identifie deux quantités dont les logarithmes sont équivalents. L'échelle des $L_x[\nu]$ est encore plus grossière. On note aussi que

$$L_x[\nu_1]L_x[\nu_2] = L_x[max(\nu_1, \nu_2)],$$

si $\nu_1 \neq \nu_2$ et

$$L_x[\nu, \lambda_1]L_x[\nu, \lambda_2] = L_x[\nu, \lambda_1 + \lambda_2]$$

si $\lambda_1 + \lambda_2 \neq 0$. Ces fonctions forment une progression entre le polynomial et l'exponentiel. On sait par ailleurs que

Théorème 2 *Soient ν, w, λ et μ des constantes et x un réel tendant vers l'infini. La probabilité qu'un nombre de taille $L_x[\nu, \lambda]$ soit $L_x[w, \mu]$-lisse est*

$$L_x[\nu - w, -\lambda(\nu - w)/\mu + o(1)].$$

Pour une preuve voir [14]. Supposons maintenant que nous ayons à analyser l'algorithme suivant.

Algorithme 2 *Soit N un entier. On se donne une borne de lissité $B = L_N[w]$ et l'on tire au hasard des entiers de taille $C = L_N[1]$, c'est-à-dire polynomiale en n. On continue jusqu'à avoir trouvé B tels entiers B-lisses.*

Cet algorithme est le cœur de tous les algorithmes de factorisation. Analysons le brièvement. La probabilité pour qu'un entier de taille $L_N[1]$ soit B-lisse est

$$P = L_N[1-w]$$

et le temps de calcul est

$$T = P^{-1}B = L_N[1-w]L_N[w] = L_N[max(1-w,w)].$$

On voit que T est minimum pour $1-w = w = 1/2$ ce qui indique une tendance des algorithmes subexponentiels vers une complexité

$$T_0 = L_N[\frac{1}{2}] = exp((\log N)^{1/2+o(1)}).$$

Pour cette raison, on a cru très longtemps que la complexité intrinsèque de la factorisation était $L_N[\frac{1}{2}]$, c'est-à-dire la complexité du crible quadratique. C'est pourquoi presque toutes les grandeurs que nous aurons à considérer dans les sections suivantes seront de la forme

$$A = L_N[\frac{1}{2}, \alpha],$$

où N est le nombre à factoriser et α un réel.

Dans la section 2.10 nous montrons comment le crible algébrique (number field sieve) factorise heuristiquement en temps

$$T_1 = L_N[1/3] = exp((\log N)^{1/3+o(1)}).$$

2.7 Analyse d'un algorithme naïf

On étudie l'algorithme de factorisation 1 (crible linéaire).

On suppose que B est de l'ordre de $L_N[1/2, b]$ et C est de l'ordre de $L_N[1/2]$, donc r est B-lisse avec probabilité

$$P_1 = L_N[0].$$

Quant à $r+N$, il est de taille

$$L_N[1,1],$$

et il est B-lisse avec probabilité

$$P_2 = L_N[\frac{1}{2}, -\frac{1}{2}.\frac{1}{b}].$$

On voit donc que P_1 est négligeable devant P_2. Nous considérons donc comme acquise la condition que r est B-lisse. Par ailleurs, B et $\pi(B)$ ont des logarithmes équivalents, on peut donc les identifier.

Le temps total de la recherche de couples $(r, r+N)$ B-lisses est alors

$$T_1 = \frac{B}{P_2}$$

puisqu'on veut $\pi(B)$ couples $(r, r+n)$ qui soient B-lisses et que chaque couple a une probabilité P_2 d'être B-lisse.

L'inversion d'une matrice creuse de taille $\pi(B)$ se fait en temps $\pi(B)^2$ par des méthodes *ad hoc*. Ici nous avons donc

$$T_2 = B^2.$$

De plus, on doit s'assurer qu'il y a assez de couples B-lisses, c'est-à-dire que le nombre de couples B-lisses pour r entre 0 et C est plus grand que $\pi(B)$. Cela s'exprime par la contrainte

$$CP_1P_2 > B.$$

Nous devons choisir les paramètres $B = L_N[1/2, b]$ et $C = L_N[1/2, c]$ afin de minimiser $T_1 + T_2$ sous la contrainte ci-dessus. Toutes les analyses que nous aurons à faire prendront la même forme, à savoir une optimisation sous contrainte.

Ici nous avons

$$T_1 + T_2 = L_N[\frac{1}{2}, max(b + \frac{1}{2b}, 2b)]$$

et la contrainte

$$c \geq b + \frac{1}{2b}.$$

On voit qu'ici la contrainte est innocente. On minimise pour $b > 0$ la fonction affine par morceau

$$b \mapsto max(b + \frac{1}{2b}, 2b)$$

avec $b = 1/\sqrt{2}$. On a alors $c = \sqrt{2}$ et donc

Théorème 3 *L'algorithme ci-dessus est optimal pour*

$$B = L_N[\frac{1}{2}, \frac{1}{\sqrt{2}}],$$

et il s'exécute en temps

$$T = L_N[\frac{1}{2}, \sqrt{2}].$$

2.8 Analyse du crible quadratique

Dans cette section nous analysons le crible quadratique, dont un exemple a été donné aux sections 2.5.4 et 2.5.5. Nous donnons d'abord une version sommaire de cet algorithme.

Algorithme 3 *Pour factoriser un entier N, poser $m = \lfloor N^{1/2} \rceil$ et soit $k = m^2 - N$. Choisir un entier B puis chercher des entiers a tels que $k + a^2 + 2am$ soit B-lisse. On peut prendre $a = 0, 1, 2, ..., C$. Lorsqu'on a trouvé $\pi(B)$ tels entiers, on produit des congruences entre deux carrés par l'inversion d'une matrice de taille $\pi(B)$.*

On suppose toujours que B est de l'ordre de $L_N[1/2, b]$ et C de l'ordre de $L_N[1/2, c]$. On voit que cette fois, les entiers $(m^2 - N) + a^2 + 2am$ sont de taille

$$L_N[1, 1/2].$$

Ils sont donc B-lisses avec probabilité

$$P = L_N[\frac{1}{2}, -\frac{1}{2}.\frac{1}{2b}].$$

Le temps de recherche des entiers a convenables est

$$T_1 = \frac{B}{P} = L_N[\frac{1}{2}, b + \frac{1}{4b}].$$

Le temps d'inversion de la matrice est

$$T_2 = \pi(B)^2 = L_N[\frac{1}{2}, 2b].$$

La contrainte se traduit par l'inégalité

$$c \geq b + \frac{1}{4b}.$$

On minimise $T_1 + T_2$ en minimisant $max(b + \frac{1}{4b}, 2b)$ pour $b > 0$ ce qui donne $b = 1/2$. On a donc le

Théorème 4 *Le crible quadratique est optimal pour*

$$B = L_N[\frac{1}{2}, \frac{1}{2}],$$

et il s'exécute en temps

$$T = L_N[\frac{1}{2}, 1].$$

On note un gain *exponentiel* de $\sqrt{2}$ par rapport à l'algorithme naïf. Ce gain est dû à la plus petite taille des entiers supposés B-lisses. Au lieu d'avoir des entiers de taille N, on a des entiers de taille $N^{1/2}$.

2.9 Analyse du crible algébrique à dimension fixée

Dans cette section nous analysons le crible algébrique avec un corps de nombres tel qu'il est présenté dans le paragraphe 2.5.7. Nous rappelons brièvement cet algorithme dans une version très primitive. Soit d un entier plus grand que 1.

Algorithme 4 *Pour factoriser l'entier N, soit $m = \lfloor N^{1/d} \rceil$. Choisir un entier B. Fabriquer un polynôme f unitaire de degré d et de hauteur $N^{1/d}$ tel que $f(m) = N$. Chercher des entiers a et b premiers entre eux tels que $(a + bm)f(-a/b)b^d$ soit B-lisse. On essaie par exemple $a = 1, -1, 2, -2, 3, -3...$ et $b = 1, 2, 3, 4, 5, ...$ Les $|a|$ et b sont bornés par un entier C. Lorsqu'on a collecté $2\pi(B)$ tels couples (a, b), on construit une matrice de taille $2\pi(B)$ dont l'inversion produit des congruences entre carrés modulo N.*

Avant d'analyser cet algorithme on remarque que la probabilité pour que le produit de deux nombres de taille donnée soit lisse est plus grande que la probabilité pour qu'un nombre général de taille double soit lisse. Cependant, dans le cadre primitif de nos analyses, on ne voit pas la différence.

On prend B de taille $L_N[\frac{1}{2}, b]$ et C de taille $L_N[\frac{1}{2}, c]$. Alors m et f sont de taille $L_N[1, \frac{1}{d}]$ et donc $(a + bm)f(-a/b)b^d$ est de taille

$$L_N[1, \frac{2}{d}]$$

aussi car a et b sont $L_N[\frac{1}{2}]$.

La probabilité pour que $(a + bm)f(-a/b)b^d$ soit lisse est alors

$$P = L_N[\frac{1}{2}, -\frac{1}{db}].$$

Le temps est donc

$$T = L_N[\frac{1}{2}, b + \frac{1}{db}],$$

et la contrainte est que le nombre de paires (a, b) de taille C multiplié par la probabilité d'être lisse soit plus grand que B

$$PC^2 \geq B,$$

soit

$$2c \geq b + \frac{1}{db}.$$

On minimise le temps en prenant

$$b = \frac{1}{\sqrt{d}}.$$

Théorème 5 *Pour tout entier $d \geq 2$, le crible algébrique de dimension d factorise en temps*

$$T_d = L_N[\frac{1}{2}, \sqrt{\frac{4}{d}}].$$

Il est important de remarquer que si l'on prend $d = 2$ alors on retrouve la complexité de crible naïf et non celle du crible quadratique. En particulier ce crible général devient meilleur que le crible quadratique si

$$\sqrt{\frac{4}{d}} < 1,$$

soit pour $d \geq 5$.

2.10 Variation de la dimension

On veut étudier plus en détail l'effet de la variation de dimension d sur le crible algébrique. À cause de la forme des fonctions L, il est naturel de poser

$$d = (\log N)^{\epsilon}(\log\log N)^{-\epsilon}$$

avec $0 < \epsilon < 1$.

Ainsi la dimension croît doucement avec N. Alors la taille de $N^{1/d}$ n'est plus $L_N[1, 1/d]$ mais $L_N[1-\epsilon]$. De même, si la taille de C est $L_N[c]$, et si la taille de B est $L_N[b]$, alors la taille de c^d est $L_N[c+\epsilon]$. Ainsi la taille de $(a+bm)f(-a/b)b^d$ est $L_N[max(1-\epsilon, c+\epsilon)]$, et la probabilité d'être lisse est

$$P = L_N[max(1-\epsilon, c+\epsilon) - b].$$

On a alors

$$T = \frac{B}{P} = L_N[max(1-\epsilon-b, c+\epsilon-b, b)],$$

et la contrainte

$$\frac{C}{P} \geq B$$

donne

$$c \geq max(1-\epsilon-b, c+\epsilon-b, b).$$

La minimisation sous contrainte de T pour $\epsilon, b, c \in [0,1]$ donne $\epsilon = b = c = 1/3$ et donc on a le

Théorème 6 *Le crible algébrique avec un corps de dimension $d = (\log N)^{\frac{1}{3}}(\log\log N)^{-\frac{1}{3}}$ factorise en temps*

$$L_N[\frac{1}{3}].$$

Il nous reste à étudier plus finement cette complexité. On pose donc

$$B = L_N[\frac{1}{3}, b] \ , \ C = L_N[\frac{1}{3}, c] \ , \text{ et } d = \epsilon(\log N)^{\frac{1}{3}}(\log\log N)^{-\frac{1}{3}}.$$

Alors m et les coefficients de f sont de taille $L_N[\frac{2}{3}, \frac{1}{\epsilon}]$, et a^d est de taille $L_N[\frac{2}{3}, \epsilon c]$. Ainsi $(a+bm)f(-a/b)b^d$ est de la taille de m^2a^d c'est-à-dire $L_N[\frac{2}{3}, \frac{2}{\epsilon}+c\epsilon]$. La probabilité que $(a+bm)f(-a/b)b^d$ soit lisse est alors

$$P = L_N[\frac{1}{3}, -\frac{1}{3b}.(\frac{2}{\epsilon} + c\epsilon)],$$

et le temps

$$T = L_N[\frac{1}{3}, b + \frac{1}{3b}.(\frac{2}{\epsilon} + c\epsilon)].$$

Quant à la contrainte, c'est

$$2c \geq b + \frac{1}{3b}(\frac{2}{\epsilon} + c\epsilon).$$

Ici, le problème d'optimisation est plus subtil car il implique trois variables. Nous donnons quelques astuces pour deviner la solution optimale.

D'abord, on remarque que la contrainte est atteinte en l'optimum. Ensuite on note qu'une expression de la forme $x + \frac{K}{x}$ est optimale pour $x = \frac{K}{x}$.

En faisant cette observation pour la variable ϵ on trouve

$$\epsilon c = \frac{2}{\epsilon}.$$

Ensuite en regardant la variable b on trouve

$$b = \frac{4}{3b\epsilon}.$$

Enfin, puisque la contrainte est atteinte on a $2c = 2b$ et donc $b = c$. On trouve alors

$$\epsilon = 3^{\frac{1}{3}}, \ , b = c = (\frac{8}{9})^{\frac{1}{3}}.$$

On en déduit le

Théorème 7 *Le crible algébrique avec un corps de dimension*

$$d = 3^{\frac{1}{3}}.(\log N)^{\frac{1}{3}}(\log\log N)^{-\frac{1}{3}}$$

factorise en temps

$$L_N[\frac{1}{3}, (\frac{64}{9})^{\frac{1}{3}}].$$

Bibliographie

[1] L.M. Adleman. Factoring numbers using singular integers. *STOC*, pages 64–71, 1991.

[2] L.M. Adleman, J. DeMarrais, and M.-D. Huang. A subexponential algorithm for discrete logarithms over all prime fields. *Math. Comp.*, 61 :1–155, 1993.

[3] L.M. Adleman and M.-D. A. Huang. *Primality testing and abelian varieties over finite fields*. Springer, 1992.

[4] A.O.L. Atkin and F. Morain. Elliptic curves and primality proving. *Math. Comp.*, 61 :29–68, 1993.

[5] Gilbert Baumslag. *Topics in Combinatorial Group Theory*. Birkhauser, 1993.

[6] J.P. Buhler, H.W. Lenstra, and C. Pomerance. *Factoring integers with the number field sieve*, volume 1554 of *Lect. Notes in Math.*, pages 48–89. Springer Verlag, 1993.

[7] A.M. Cohen, H. Cuypers, and H. Sterk. *Some Tapas of Computer Algebra*. Springer, 1993.

[8] M.R. Garey and D.S. Johnson. *Computers and intractability*. Freeman, 1979.

[9] D.L. Johnson. *Presentation of Groups*. London Mathematical Society, 1990.

[10] A. Karatsuba. *Soviet Physics-Doklady*, 7 :595–596, 1963.

[11] D. Knuth. *The Art of Computer Programming, Volume 2, third edition*. Lecture Notes in Mathematics. Addison -Wesley, 1998.

[12] Henri Lebesgue. *Leçons sur les constructions géométriques*. Gauthier-Villars, 1950.

[13] A.K. Lenstra, H.W. Lenstra Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261 :515–534, 1982.

[14] A.K. Lenstra, H.W. Lenstra, M.S. Manasse, and J.M. Pollard. *The number field sieve*, volume 1554 of *Lect. Notes in Math.*, pages 11–40. Springer Verlag, 1993.

[15] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proofs. *Proc. 31st IEEE Symp. on the Foundations of Computer Science*, pages 1–10, 1990.

[16] Y. Matiyasevich. Enumerable sets are diophantine. *Soviet Math. Dokl.*, 11 :354–357, 1970.

[17] Maurice Mignotte. An inequality about irreducible factors of integer polynomials. *Journal of Number Theory*, 30 :156–166, 1988.

[18] Louis Monier. Evaluation and comparison of two efficient probabilistic primality testing algorithms. *Theoret. Comput. Sci.*, 12 :97–108, 1980.

[19] C.H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.

[20] C. Pomerance. Fast, rigorous factorization and discrete logarithm algorithms. In *Discrete algorithms and complexity*, pages 119–143. Academic Press, 1987.

[21] Michael O. Rabin. Probabilistic algorithm for testing primality. *J. Number Theory*, 12 :128–138, 1980.

[22] A. Shamir. **IP=PSPACE**. *Proc. 31st IEEE Symp. on the Foundations of Computer Science*, pages 11–15, 1990.

[23] H. Stichtenoth. *Algebraic function fields and codes*. Springer, 1993.

[24] E.C. Titchmarsh and D.R. Heath-Brown. *The theory of the Riemann Zeta-function*. Springer, 1986.

Architectures tolérant les intrusions : concepts et conception

Paulo Veríssimo

Professeur à l'Université de Lisbonne
Portugal

Intrusion Tolerance: Concepts and Design Principles

a Tutorial

Paulo Veríssimo
Univ. of Lisboa Faculty of Sciences
Lisboa – Portugal

pjv@di.fc.ul.pt
http://www.navigators.di.fc.ul.pt

Pointers to tutorial material

- With the aim of **disseminating intrusion tolerance concepts and techniques to a wide audience, the tutorial, and a companion text, are available from the University of Lisboa Technical Reports web site.**
- **http://www.navigators.di.fc.ul.pt/it/index.htm**
- **Tutorial**
- **Veríssimo, Paulo: Intrusion Tolerance: Concepts and Design Principles. A Tutorial. Technical Report DI/FCUL TR02-6, Dept. of Informatics, University of Lisboa (2002). abstract - pdf**
- **Text**
- **An *extended version* of the paper: Veríssimo, P. E., and Neves, N. F., and Correia, M. P.: Intrusion-Tolerant Architectures: Concepts and Design. In: Architecting Dependable Systems. Springer-Verlag LNCS 2677 (2003). Technical Report DI/FCUL TR03-5, Dept. of Informatics, University of Lisboa (2003). abstract - pdf**

- *A lot of the material presented here derives from past experience with fault tolerant and secure systems, and from new work and challenging discussions, during the past three years, within the European IST MAFTIA project. I wish to warmly acknowledge the contributions of all members of the team, several of whom contributed results presented here, and collectively have represented a fenomenal thinking tank.*
- *Further reading on the concepts and design principles presented here can thus be found in:*
  - Paulo Veríssimo and Luís Rodrigues, *Distributed Systems for System Architects*, Kluwer Academic Publishers, 2001. http://www.navigators.di.fc.ul.pt/dssa/
  - A. Avizienis, J.-C. Laprie and B. Randell, *Fundamental Concepts of Dependability*, Research Report N°01145, LAAS-CNRS, April 2001. http://www.laas.fr/
  - P. Veríssimo and N. F. Neves, eds., *Service and Protocol Architecture for the MAFTIA Middleware*. Deliv. D23, Project MAFTIA IST-1999-11583, Tech.Rep. DI/FCUL TR-01-1, Univ. Lisboa Jan. 2001. http://www.di.fc.ul.pt/tech-reports/abstract01-1.html
  - D. Powell and R. Stroud, eds., *MAFTIA Conceptual Model and Architecture*, Deliv. D2, Project MAFTIA IST-1999-11583, Tech.Rep. DI/FCUL TR-01-10, Univ. Lisboa Nov. 2001. http://www.di.fc.ul.pt/tech-reports/abstract01-10.html

The case for Intrusion Tolerance

- **Distribution and fault tolerance go hand in hand:**
  - **You distribute to achieve resilience to common mode faults**
  - **You embed FT in a distributed system to resist higher fault probabil.**
- **Security and distribution go hand in hand:**
  - **You break, split, and separate your information**
  - **You make life harder to an attacker**
- **So it looks like we should be talking about (distributed) malicious fault tolerance, a.k.a.**

  (Distributed) Intrusion Tolerance
- **If this is so obvious, why hasn't it happened earlier?**
  - **Distributed systems present fundamental issues and limitations that took a long time to learn**
  - **Classical fault tolerance tolerance follows a framework that is not completely fit to the universe of intentional and/or malicious faults**

Dependability as a common framework

- Dependability is classically defined as "that property of a computer system such that reliance can justifiably be placed on the service it delivers"
- The service delivered by a system is its behaviour as it is perceptible by its user(s); a user is another system (human or physical) which interacts with the former
- Dependability grew under the mental framework of *accidental faults*
- But all that is defined w.r.t. dependability can be applied to *malicious faults*

Intrusion Tolerance

- Traditionally, security has involved either:
  - Trusting that certain attacks will not occur
  - Removing vulnerabilities from initially fragile software
  - Preventing attacks from leading to intrusions
- In contrast, the tolerance paradigm in security:
  - Assumes that systems remain to a certain extent vulnerable
  - Assumes that attacks on components or sub-systems can happen and some will be successful
  - Ensures that the overall system nevertheless remains secure and operational
- Obviously, a complete approach combines tolerance with prevention, removal, forecasting, after all, the classic dependability fields of action!

1

Introduction to Security & Dependability

Introduction to security

Security Properties

- **Confidentiality**
  - the measure in which a service or piece of information is protected from unauthorized disclosure
- **Integrity**
  - the measure in which a service or piece of information is protected from illegitimate and/or undetected modification
- **Authenticity**
  - the measure in which a service or piece of information is genuine, and thus protected from personification or forgery
- **Availability**
  - the measure in which a service or piece of information is protected from denial of authorized provision or access

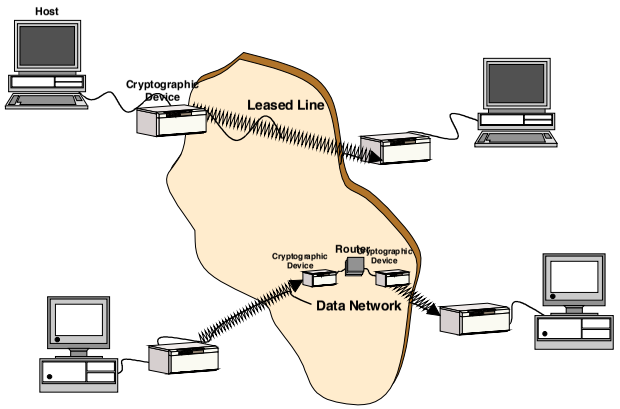
Security Frameworks

- **Secure Channels and Envelopes**
  - Communicating in a secure way
  - Dispatching information in a secure way
- **Authentication**
  - Ensuring what we deal with is genuine: end-users, data, servers, etc.
- **Protection and Authorization**
  - Protecting resources from unauthorized access
  - Ensuring the users are authorized to do just what they should
- **Auditing and Intrusion Detection**
  - Following the system execution for a posteriori analysis
  - Detecting anomalous usage in runtime

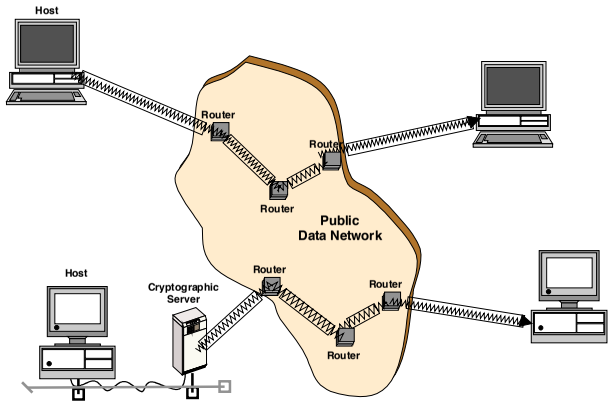
Example Secure Networks and Architectures

In Paulo Veríssimo and Luís Rodrigues, ***Distributed Systems for System Architects***, Kluwer Academic Publishers, 2001.
http://www.navigators.di.fc.ul.pt/dssa/

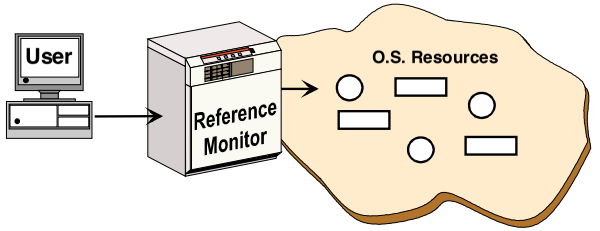
Secure Physical Circuits



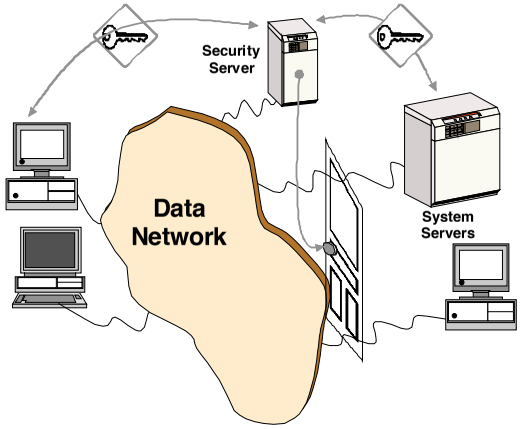
Secure Virtual Circuits



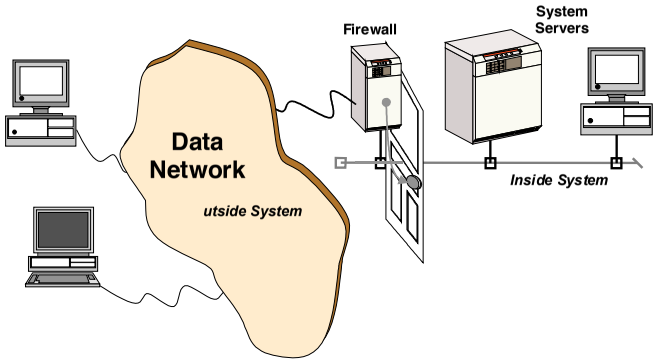
Reference Monitor



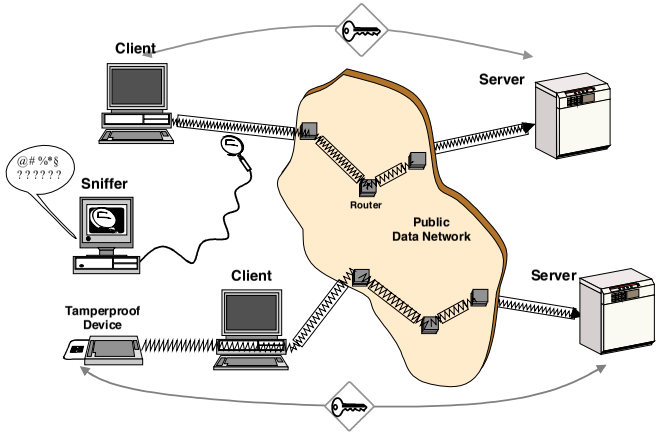
Security Server



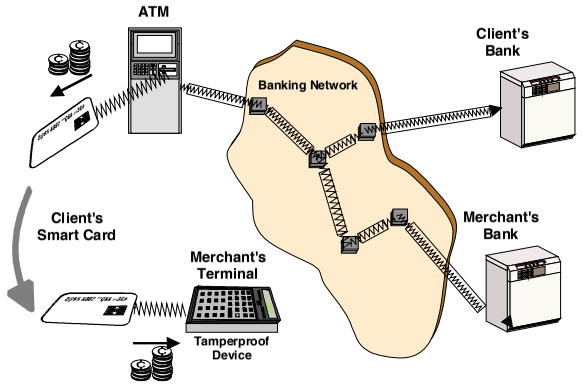
Firewall



Secure Remote Operations



Electronic Payment



Introduction to fault tolerance and dependability

The failure of computers

- *hy do computers fail and what can we do about it*

  [J. Gray]

- **Because:**
  - All that works, fails
  - We tend to overestimate our HW e SW--- that's called faith☺
- **So:**
  - We had better prevent (failures) than remedy
- **Dependability is ...**
  - that property of a computer system such that reliance can justifiably be placed on the service it delivers
- **Why?**
  - Because (faith notwithstanding) it is the scientific way to quantify, predict, prevent, tolerate, the effect of disturbances that affect the operation of the system

Does not get better with distribution

- *A distributed system is the one that prevents you from working because of the failure of a machine that you had never heard of.*

  [L. Lamport]

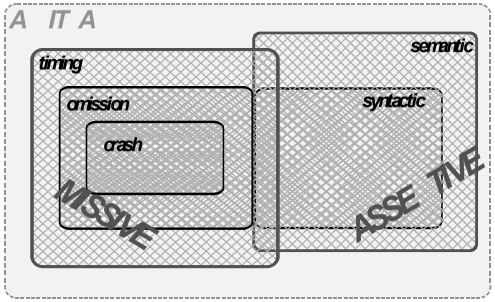
- Since:
  - Machines fail independently, for a start
  - But they may influence each other,
  - They communicate through unreliable networks, with unpredictable delays
- ...gathering machines renders the situation worse:
  - The reliability (<1) of a system is the product of the individual component reliabilities, for independent component failures
  - R(10 @ 0.99)= 0.9910= 0.90; R(10 @ 0.90)= 0.9010= 0.35

Can get much worse with malicious failures

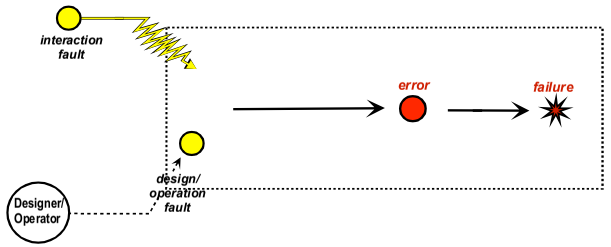
- Failures are no longer independent
  - Human attackers are the "common-mode" link
  - Components may perform collusion through distributed protocols
- Failures become harder
  - Components producing malicious, e.g. inconsistent output, at wrong times, forged, etc.
- "Fault models"?
  - How do you model the mind of an attacker?
  - Maliciously induced failures defy stochastic models for fault distribution

Interaction Fault classification

- Omissive
  - Crash
    - » host that goes down
  - Omission
    - » message that gets lost
  - Timing
    - » computation gets delayed
- Assertive
  - Syntactic
    - » sensor says air temperature is 100°
  - Semantic
    - » sensor says air temperature is 26° when it is 30°



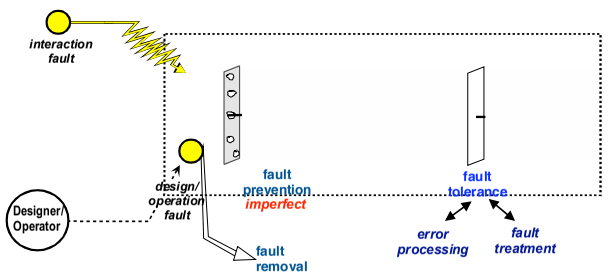
sequence fault→ error→ failure



Achieving dependability

- Fault prevention
  - how to prevent the occurrence or introduction of faults
- Fault tolerance
  - how to ensure continued correct service provision despite faults
- Fault removal
  - how to reduce the presence (number, severity) of faults
- Fault forecasting
  - how to estimate the presence, creation and consequences of faults

Dependability measures



Dependability properties

- Reliability
  - the measure of the continuous delivery of correct service (ex. MTTF)
- Maintainability
  - the measure of the time to restoration of correct service (ex. MTTR)
- Availability
  - measure of delivery of correct service with respect to alternation between correct and incorrect service (ex. MTBF/(MTBF+MTTR))
- Safety
  - the degree to which a system, upon failing, does so in a non-catastrophic manner
- Integrity ?
- Confidentiality ?
- Authenticity ?

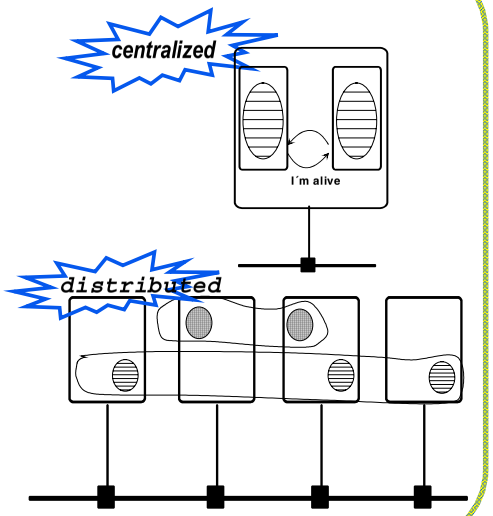
Some philosophy for a start

- What characterizes a dependable system?
  - A set of safety and liveness properties
- What characterizes a secure system?
  - A set of safety and liveness properties
- What may impair a dependable system?
  - A set of faults -> failure
- What may impair a secure system?
  - A set of faults (attacks, vulnerabilities, intrusions) -> failure
- How do I make a system dependable (normally)?
  - Using fault avoidance (prevention, removal) and fault tolerance (error detection, recovery, masking)
- How do I make a system secure (normally)?
  - Using fault avoidance (attack prevention, vulnerability removal)
  - and some bits of fault tolerance (intrusion detection)
  - Nowadays, increasingly fault tolerance (intrusion detection, recovery, masking)

Foundations of (Modular and Distributed) Fault-Tolerant Computing

Foundations of modular and distributed fault tolerance

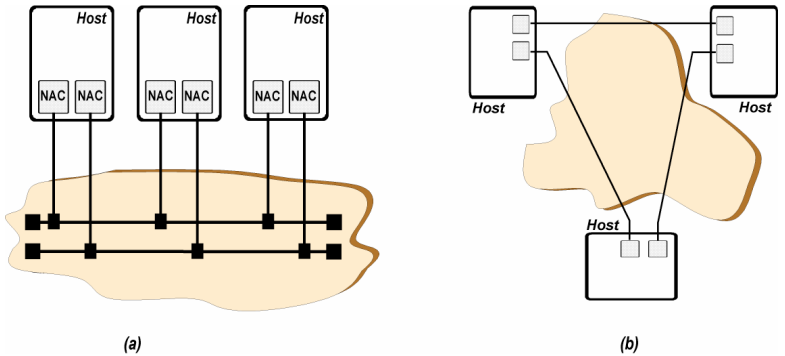
- **Topological separation**
  - failure independence
  - graceful degradation
- **Replication**
  - software vs. hardware
  - fine granularity
  - Resource optimization
- **incremental T/F by:**
  - class (omissive, semantic)
  - number of faults
  - number of replicas
    - » pairs, triples, etc.
  - Type of replica control
    - » active, passive
    - » round robin, voting



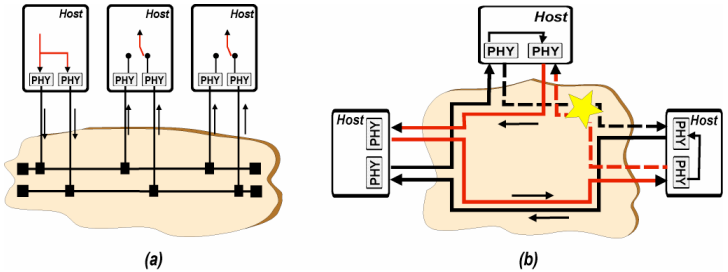
Example Fault Tolerant Networks and Architectures

In Paulo Veríssimo and Luís Rodrigues, *Distributed Systems for System Architects*, Kluwer Academic Publishers, 2001.
http://www.navigators.di.fc.ul.pt/dssa/

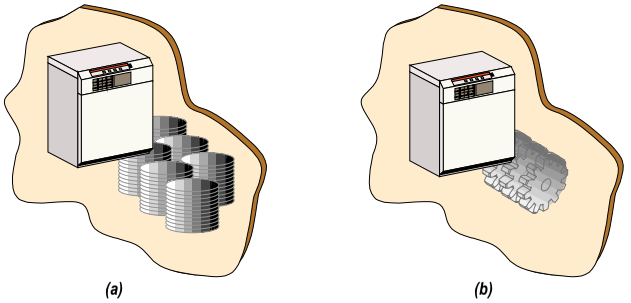
Redundant Networks



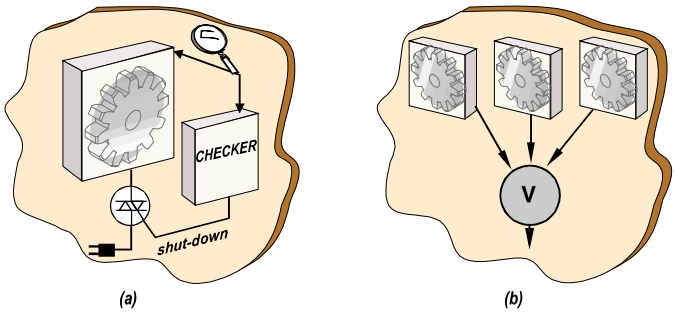
Redundant Media Networks



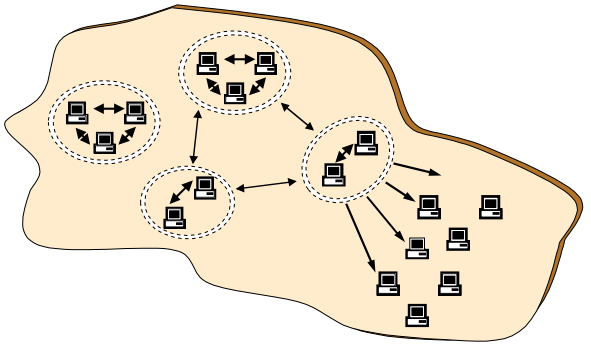
Redundant Storage and Processing



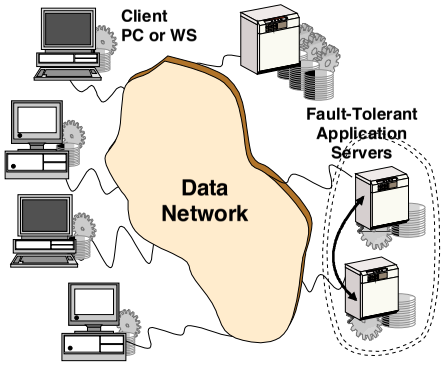
Error Detection and Masking



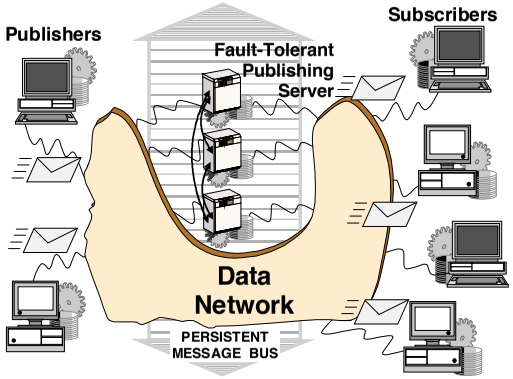
Modular Distributed FT with Replica Sets



Client-Server with FT Servers



FT Publisher-Subscriber



2

Introduction to Intrusion Tolerance

Some preliminary observations...

What measures the risk of intrusion?

- RISK is a combined measure of the level of threat to which a computing or communication system is exposed, and the degree of vulnerability it possesses:

  RISK = VULNERABILITY X THREAT

- ***The correct measure of how potentially insecure a system can be (in other words, of how hard it will be to make it secure) depends:***
  - ***on the number and severity of the flaws of the system (vulnerabilities)***
  - ***on the potential of the attacks it may be subjected to (threats)***

An example

- ***Consider the following two example systems:***
  - ***System Vault has a degree of vulnerability v_{vault}=0.1, and since it has such high resilience, its designers have put it to serve anonymous requests in the Internet, with no control whatsoever to whoever tries to access it, that is, subject to a high level of threat, t_{vault}=100.***
  - ***System Sieve, on the other hand, is a vulnerable system, v_{sieve}=10, and in consequence, its designers have safeguarded it, installing it behind a firewall, and controlling the accesses made to it, in what can be translated to a level of threat of t_{sieve}=1.***
- ***Which of them offers a lower operational risk?***
- ***Consider the product threat x vulnerability:***
  - ***with the imaginary values we attributed to each system, it equates to the same value in both systems (10), although system Vault is a hundred times less vulnerable than system Sieve.***

Should we bring the risk to zero? And... can we?

- This is classical prevention/removal
  - ***of the number and severity of the flaws of the system (vulnerabilities)***
  - ***of the potential of the attacks it may be subjected to (threats)***
- We cannot make either arbitrarily low
  - too costly and infeasible
  - certain attacks come from the kind of service being deployed
  - certain vulnerabilities are attached to the design of the system proper
- ...and the question is: should we?
- can't we talk about acceptable risk?
- doesn't the hacker also incur in a cost of intruding??!!

Another example

- ***Is SSL secure?***
  - ***Secure Sockets Layer (SSL) reportedly ensures secure client-server interactions between browsers and WWW servers. Users have tended to accept that the interactions are secure (assumed low degree of vulnerability), without quantifying how secure.***
  - ***Several companies have built their commerce servers around SSL, some of them to perform financially significant transactions on the Internet (high level of threat).***
  - ***Netscape's SSL implementation broken because of a bug that allowed to replicate session keys and thus decrypt any communication. The corrected version was then broken at least twice through brute-force attacks on the 40-bit keys version.***
  - ***This initial situation led to a high risk***

Another example (cont.)

- ***Is SSL secure enough?***
  - ***Netscape reported that it would cost at least USD10,000 to break an Internet session on the second version of SSL, in computing time.***
  - ***The cost of intruding a system versus the value of the service being provided allows the architect to make a risk assessment. Someone who spends 10 000 EURO to break into a system and get 100 EURO worth of bounty, is doing a bad business.***
  - ***This defined the acceptable risk. Unfortunately, these estimates may fail: shortly after Netscape's announcement, a student using a single but powerful desktop graphics workstation, broke it for just USD600.***
  - ***However, what went wrong here was not the principle and the attitude of Netscape, just the risk assessment they made, which was too optimistic.***

Dependability and security, two faces of a same coin

- What characterizes a dependable system?
  - A set of safety and liveness properties
- What characterizes a secure system?
  - A set of safety and liveness properties
- What may impair a dependable system?
  - A set of faults -> failure
- What may impair a secure system?
  - A set of faults (attacks, vulnerabilities, intrusions) -> failure
- How do I make a system dependable (normally)?
  - Using fault avoidance (prevention, removal) and fault tolerance (error detection, recovery, masking)
- How do I make a system secure (normally)?
  - Using fault avoidance (attack prevention, vulnerability removal)
  - and some bits of fault tolerance (intrusion detection)
  - Nowadays, increasingly fault tolerance (intrusion detection, recovery, masking)

What is Intrusion Tolerance?

- The tolerance paradigm in security:
  - Assumes that systems remain to a certain extent vulnerable
  - Assumes that attacks on components or sub-systems can happen and some will be successful
  - Ensures that the overall system nevertheless remains secure and operational, with a measurable probability
- In other words:
  - Faults--- malicious and other--- occur
  - They generate errors, i.e. component-level security compromises
  - Error processing mechanisms make sure that security failure is prevented
- Obviously, a complete approach combines tolerance with prevention, removal, forecasting, after all, the classic dependability fields of action!

Intrusion Tolerance

terminology and concepts

Fault Models
Classical methodologies
Error processing
Fault treatment

Attacks, Vulnerabilities, Intrusions

- **Intrusion**
  - an externally induced, intentionally malicious, operational fault, causing an erroneous state in the system

an intrusion has two underlying causes:

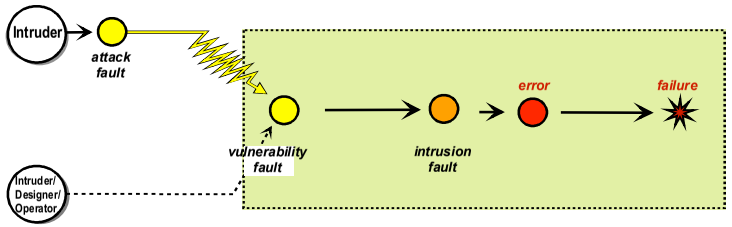
- **Vulnerability**
  - malicious or non-malicious weakness in a computing or communication system that can be exploited with malicious intention
- **Attack**
  - malicious intentional fault introduced in a computing or comm's system, with the intent of exploiting a vulnerability in that system

interesting corolaries:

- without attacks, vulnerabilities are harmless
- without vulnerabilities, there cannot be successful attacks

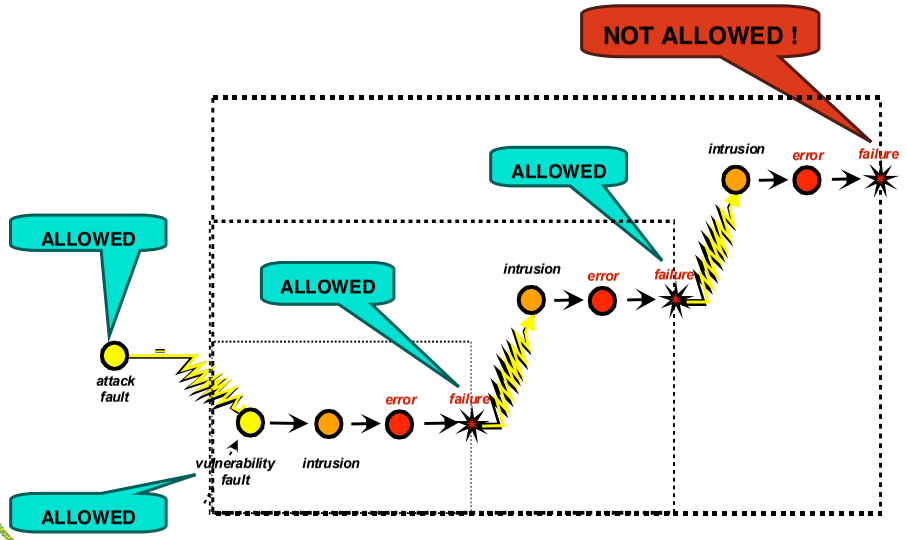
Attack-Vulnerability-Intrusion composite fault model

Hence: attack + vulnerability → intrusion → error → failure
A specialization of the generic *fault,error,failure* sequence



AVI sequence : *attack + vulnerability→ intrusion → error → failure*

Faults in Cascade

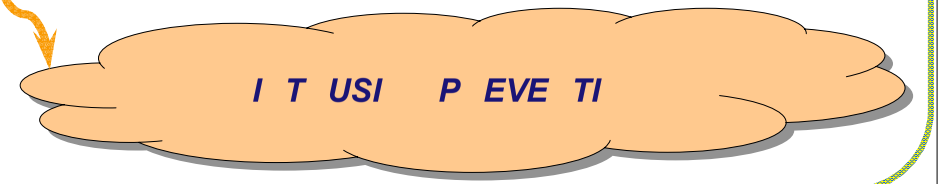


Intrusion Tolerance

Fault Models
Classical methodologies
Error processing
Fault treatment

Achieving trustworthiness w.r.t. malicious faults (the classical ways...)

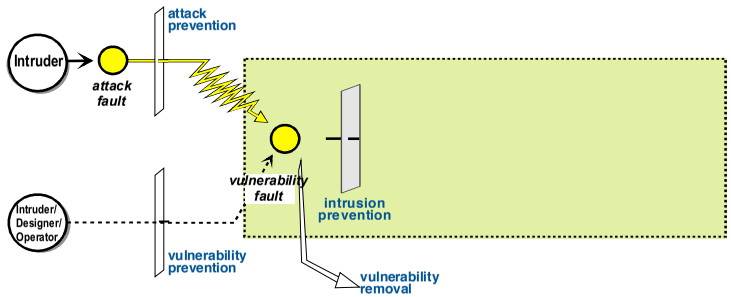
- **Attack prevention**
  - Ensuring attacks do not take place against certain components
- **Attack removal**
  - Taking measures to discontinue attacks that took place
- **Vulnerability prevention**
  - Ensuring vulnerabilities do not develop in certain components
- **Vulnerability removal**
  - Eliminating vulnerabilities in certain components (e.g. bugs)



Examples

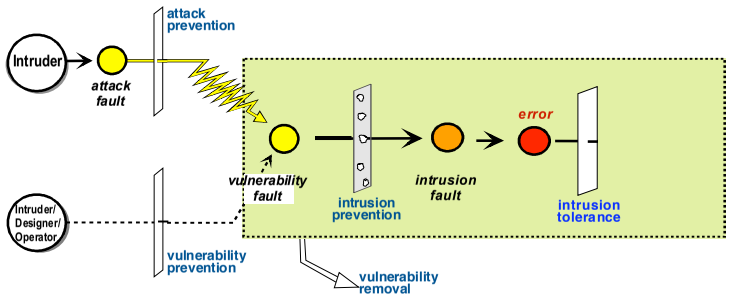
- **Attack prevention**
  - selectively filtering access to internal parts of the system (e.g., if a component is behind a firewall and cannot be accessed from the Internet, attack from there is prevented)
  - disabling JavaScript and/or Java prevents attacks by malicious scripts or applets
- **Attack removal**
  - identifying source of an external attack and taking measures to terminate it
- **Vulnerability prevention**
  - best practice in software development
  - measures preventing configuration and operation faults
- **Vulnerability removal**
  - of: coding faults allowing program stack overflow, files with root setuid in UNIX, naive passwords, unprotected TCP/IP ports

AVI Composite fault model



➢sequence : *attack + vulnerability→ intrusion→ failure*

AVI Composite fault model



➢sequence : *attack + vulnerability→ intrusion→ failure*

Intrusion Tolerance

Fault Models
Classical methodologies
Error processing
Fault treatment

Processing the errors deriving from intrusions

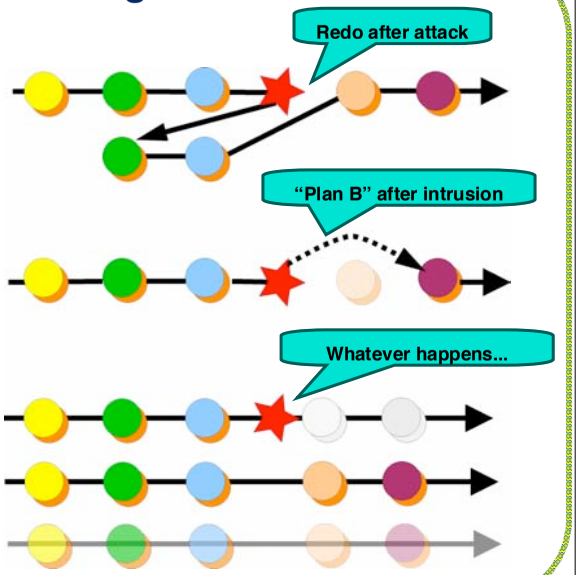
- error detection
  - detecting the error after it occurs aims at: confining it to avoid propagation; triggering error recovery mechanisms; triggering fault treatment mechanisms
  - modified files or messages; phony OS account; sniffer in operation; host flaky or crashing on logic bomb
- error recovery
  - recovering from the error aims at: providing correct service despite the error
  - recovering from effects of intrusions

Processing the errors deriving from intrusions

- backward recovery:
  - the system goes back to a previous state known as correct and resumes
  - system suffers DOS (denial of service) attack, and re-executes the corrupted operation
  - system detects corrupted files, pauses, reinstalls them, goes back
- forward recovery:
  - or proceeds forward to a state that ensures correct provision of service
  - system detects intrusion, considers corrupted operations lost and increases level of security (threshold/quorums increase, key renewal)
  - system detects intrusion, moves to degraded but safer op mode
- error masking
  - redundancy allows providing correct service without any noticeable glitch
  - voting, Byzantine agreement; fragmentation-redundancy-scattering
  - sensor correlation (agreement on imprecise values)

Error processing at work

- backward recovery
- forward recovery
- error masking



Intrusion Tolerance

Fault Models
Classical methodologies
Error processing
Fault treatment

Fault Treatment facets

- Diagnosis
  - determine cause of error, i.e., the fault(s): location and nature
  - non-malicious or malicious syndrome (intrusion)?
  - attack? --- to allow removal/retaliation
  - vulnerability? --- to allow removal
- Isolation
  - prevent new activation
  - intrusion: prevent further penetration
  - attack: disable further attacks of this kind (block the origin)
  - vulnerability: passivate the cause of successful attack (e.g. patch)
- Reconfiguration
  - so that fault-free components provide adequate/degraded service
  - contingency plans to degrade/restore service

Intrusion Tolerance

mechanisms and strategies

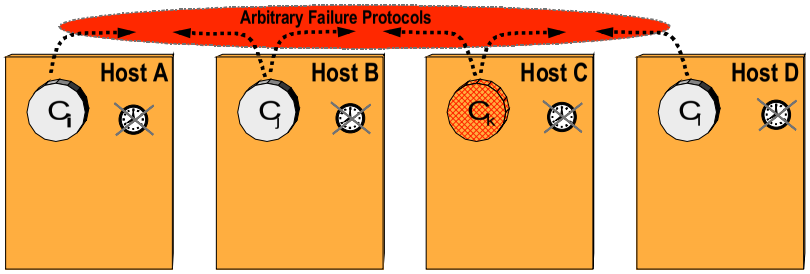
In extended version of the paper: Veríssimo, P. E., and Neves, N. F., and Correia, M. P.: Intrusion-Tolerant Architectures: Concepts and Design. In: Architecting Dependable Systems. Springer-Verlag LNCS 2677 (2003). Technical Report DI/FCUL TR03-5, Dept. of Informatics, University of Lisboa (2003). abstract - pdf

Modelling malicious failures

- Basic types of failure assumptions:
  - Controlled failures : assume qualitative and quantitative restrictions on compon. failures, hard to specify for malicious faults
  - Arbitrary failures : unrestricted failures, limited only to the "possible" failures a component might exhibit, and the underlying model (e.g. synchronism)
- What are malicious failures?
  - how do we model the mind and power of the attacker?
  - hard, but definitely the hacker has both a quantitative and a qualitative effect on failure modes
- Fail-controlled vs. fail-arbitrary models in face of intrusions
  - FC have a coverage problem, but are simple and efficient
  - FA are normally inefficient, but safe

Fail-uncontrolled IT protocols

- Time-free, R/T operation impossible
- Arbitrary failure environment
- Arbitrary failure protocols
- Used in: probabilistic Byzantine-agreement based set of protocols



Arbitrary failure assumptions

- probl. of coverage of controlled failure assumptions:
  - all lies on the coverage of the fail-controlled subsystem assumptions
- operations of very high value and/or criticality:
  - financial transactions of very high value
  - contract signing; provision of long term credencials
  - critical control operations
  - risk of failure due to violation of assumptions cannot be incurred
- arbitrary-failure resilient building blocks (e.g. Byzantine agreement protocols):
  - no assumptions on existence of security kernels or other fail-controlled components
  - time-free approach, i.e. no assumptions about timeliness
  - then no R/T possible!

Modeling malicious failures

- Intrusion-aware composite fault models
  - the competitive edge over the hacker
  - AVI: attack-vulnerability-intrusion fault model
- Combined use of prevention and tolerance
  - malicious failure universe reduction
  - attack prevention, vulnerability prevention, vulnerability removal, in system architecture subsets and/or functional domains subsets
- Hybrid failure assumptions
  - different failure modes for distinct components
  - reduce complexity and increase performance, maintaining coverage
- Quantifiable assumption coverage
  - fault forecasting (on AVI)

Did you say trusted?

- Sometimes components are tamper-proof, others tamper-resistant...
  - Watch-maker syndrome:
    - » *Is this watch waterproof*
    - » *o, it s water resistant*
    - » *Anyway, I assume that I can swim with it*
    - » *ell yes, you can but i wouldn t trust that very much*
- How can something trusted be not trustworthy?
  - *Unjustified reliance syndrome:*
    - » *I trust Alice*
    - » *ell ob, you shouldn t, she s not trustworthy*
- What is the difference? If we separate specification from implementation, and provide notions of *justification* and of *coverage*, all becomes clearer

Trusted, Trustworthy

- **Trusted**
  - Believed to be dependable. Relation of reliance, dependence.
- **Trustworthy**
  - Dependable. Property of a (sub)system that makes us justifiably rely on it

On Trust and Trustworthiness

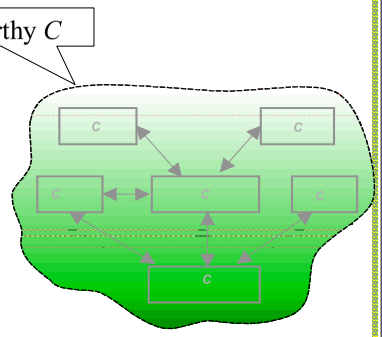
- *Thou shalt not trust non trustworthy components*
  - A trusted component has a set of properties on which another component(s) depend…
- Trust should be placed to the extent of that component's trustworthiness, the measure in which it meets those properties
  - trust may have several degrees, quantitatively or qualitatively
  - related not only with security-relat. properties (e.g., timeliness)
  - trust and trustworthiness lead to complementary aspects of the design and verification process
- when A trusts B, A assumes something about B: Trustworthiness of B measures the coverage of the assumption
- ... and trustworthiness is never 1 in real systems...

Tamperproofness and its coverage or "tamper-resistance" not needed

- **Tamperproof**
  - Property of a system/component of being shielded, i.e. whose attack model is that attacks can only be made at the regular interface
  - Coverage of the "tamperproof" assumption may not be perfect, and there can be several degrees of such tamperproofness
- **Example:**
  - *Implementation of a security service using ava Cards to store private keys. e assume .Cards are tamperproof, and so we argue that they are trustworthy they will not reveal these keys to an unauthorised party . ence we can justifiably argue that the service is trusted, with the coverage given by our assumptions, namely, the tamperproofness of Cards*

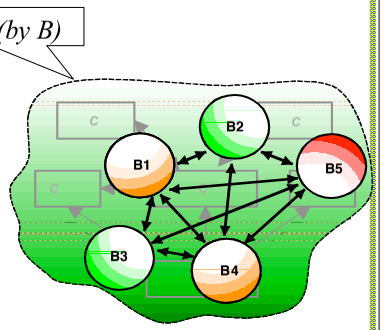
A robust design approach

- Architectural hybridization:
  - failure assumptions enforced by architecture and construction, thus substantiated
  - combined/recursive use of attack/vulnerability/intrusion prevention/removal/tolerance
- Trusted (trustworthy) components:
  - components or subsystems with justified coverage, used in the construction of fault-tolerant protocols under architectural hybrid failure assumptions



A robust design approach

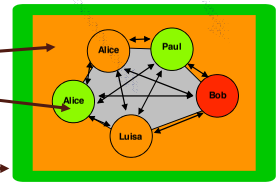
- Architectural hybridization:
  - failure assumptions enforced by architecture and construction, thus substantiated
  - combined/recursive use of attack/vulnerability/intrusion prevention/removal/tolerance
- Trusted (trustworthy) components:
  - components or subsystems with justified coverage, used in the construction of fault-tolerant protocols under architectural hybrid failure assumptions



On coverage and separation of concerns

- predicate P holds with a coverage Pr
  - we say that we are confident that P has a probability Pr of holding
- environmental assumption coverage (Pr_e)
  - set of assumptions (H) about the environment where system will run
  - *Pre Pr f f any fault*
- operational assumption coverage (Pr_o)
  - the assumptions about how the system/algorithm/mechanism proper (A) will run, under a given set of environmental assumptions
  - *Pro Pr A*

Pr A Pro x Pre Pr A x Pr f



Hybrid failure assumptions considered useful

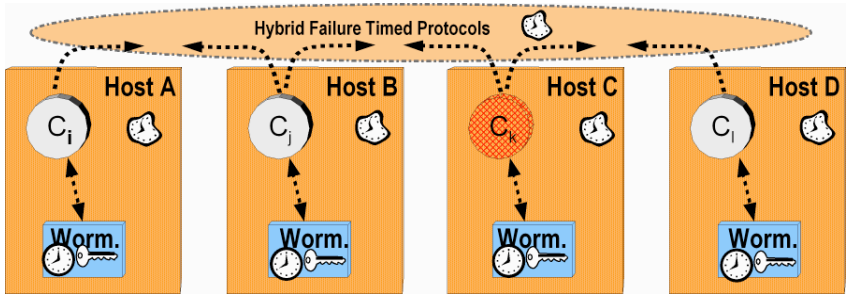
- Classic hybrid fault models
  - flat, use stochastic foundation to explain different behavior from same type of components (i.e. *k* crash and *w* byzantine in vector of values)
- The problem of substance
  - an intentional player defrauds these assumptions
- Architectural hybridisation
  - different assumptions for distinct component subsets
  - behavior enforced by construction: trustworthiness

Intrusion tolerance with hybrid failure assumptions

- Using trusted components or subsystems:
  - black boxes with benign behavior, of omissive or weak fail-silent class
  - can have different capabilities (e.g. synchronous or not; local or distributed), can exist at different levels of abstraction
- Fault-tolerant protocols:
  - more efficient than truly arbitrary assumptions protocols
  - more robust than non-enforced controlled failure protocols
- Tolerance attitude in design:
  - unlike classical prevention-based approaches, trusted components do not mediate all accesses to resources and operations
  - assist only crucial steps of the execution of services and applications
  - protocols run in untrusted environment, local participants only trust trusted components, single components can be corrupted
  - correct service built on distributed fault tolerance mechanisms, e.g., agreement and replication amongst participants in several hosts

Fail-controlled I/T protocols with Local Trusted Subsystems

- Trustworthy subsystem (also called Wormhole) - e.g. smart or Java card; appliance board
- Secure, and time-free or timed (as in figure)
- Arbitrary failure environment + Local Wormhole
- Hybrid failure protocols
- Example usage: FT distributed data dissemination with authentication and authorisation protocols

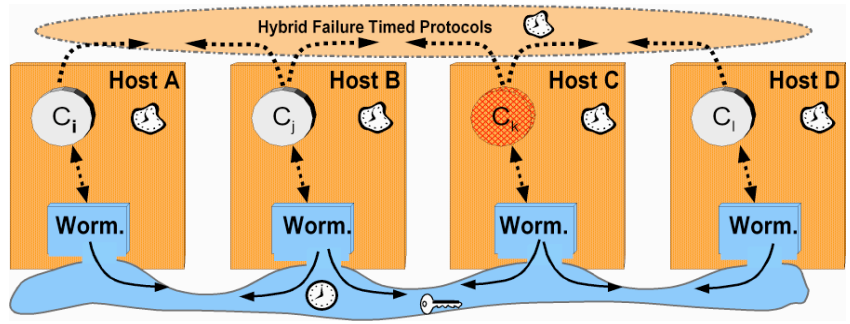


Intrusion tolerance with hybrid failure assumptions

- distributed trusted components or subsystems:
  - amplifying the notion of local trusted component, implementing distributed trust for low-level operations
  - based on appliance boards with private control channels
  - can supply basic distributed security and HRT timing functions
- how they assist protocols security-wise:
  - protocol participants exchange messages in a world full of threats, some of them may even be malicious and cheat
  - there is an oracle that correct participants trust, and a channel that they can use to get in touch with each other, even for rare moments
  - acts as a checkpoint that malicious participants have to synchronise with, and this limits their potential for Byzantine actions

Fail-controlled IT protocols with a Distributed Trusted Subsystem

- Distributed Trustworthy subsystem (distr. Wormhole) - e.g. appliance boards interconnected by dedicated network
- Secure, and time-free or timed (as in figure)
- Arbitrary failure environment + Distributed Wormhole
- Hybrid failure protocols
- Example: FT transac. prots requiring timing constraints (e.g. SCADA, DCS)

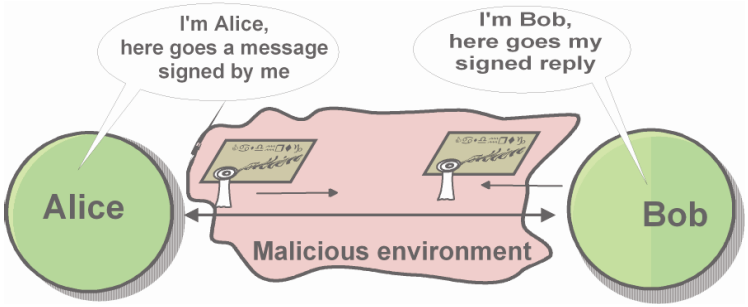


Design of intrusion tolerant systems

Some paradigms under an Intrusion Tolerance look

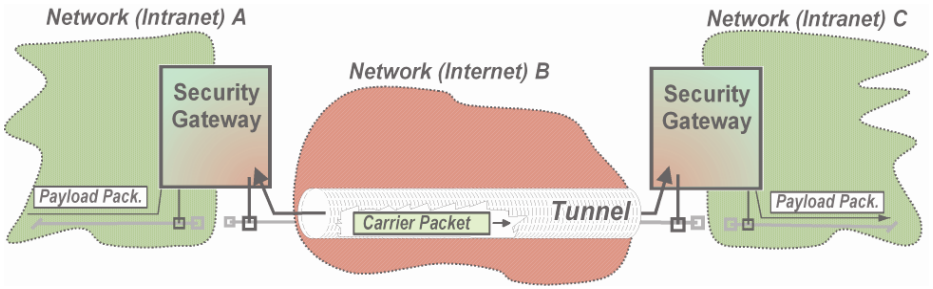
In extended version of the paper: Veríssimo, P. E., and Neves, N. F., and Correia, M. P.: Intrusion-Tolerant Architectures: Concepts and Design. In: Architecting Dependable Systems. Springer-Verlag LNCS 2677 (2003). Technical Report DI/FCUL TR03-5, Dept. of Informatics, University of Lisboa (2003). abstract - pdf

Authentication, signatures, MACs



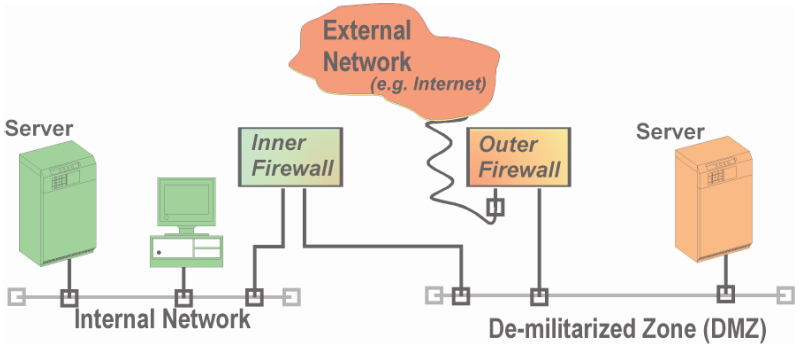
- Intrusion prevention device: enforces authenticity, integrity
- Coverage: signature/authentication method
- End-to-end problem: who am I authenticating? me or my PC?

Tunnelling, secure channels



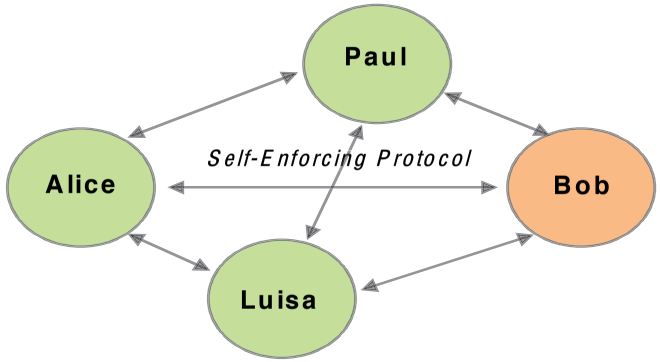
- Intrusion prevention device: enforces confidentiality, integrity (authenticity)
- Coverage: tunnelling method, resilience of gateway
- End-to-end problem: are all intranet guys good?

Firewalling



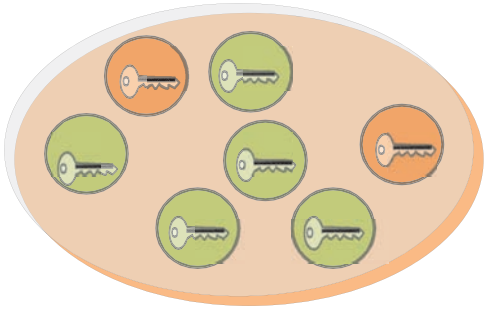
- Intrusion prevention device: prevents attacks on inside machines
- Coverage: semantics of firewall functions, resilience of bastions
- End-to-end problem: are all internal network guys good?

Communication and agreement protocols



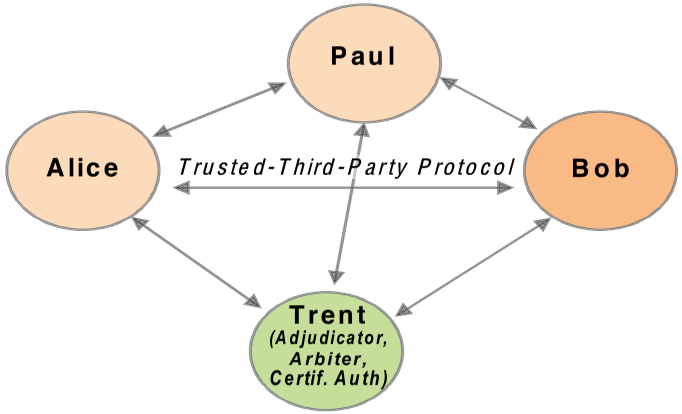
- Intrusion tolerance device: error processing or masking (3f+1, 2f+1, f+2)
- Coverage: semantics of protocol functions, underlying model assumptions

Threshold cryptography



- Intrusion tolerance device: error processing/masking (f+1 out of n)
- Coverage: crypto semantics, brute force resilience, underlying model assumptions

Trusted Third Party (TTP) protocols

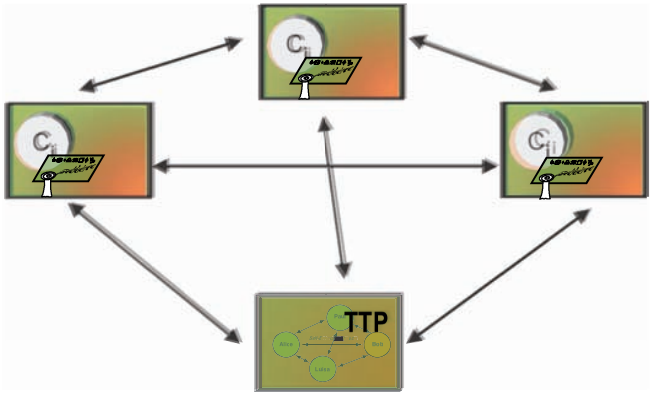


- Intrusion tolerance device: error processing/masking
- Coverage: semantics of protocol functions, underlying model assumptions, resilience of TTP

Strategies for construction of IT subsystems

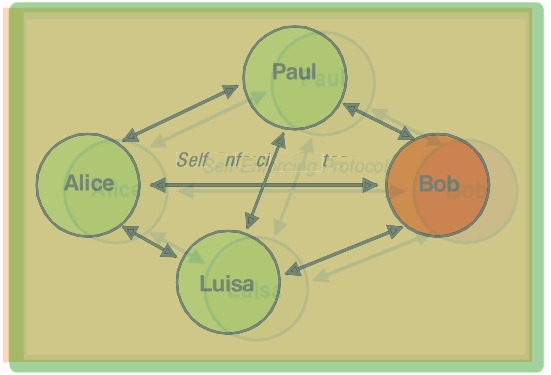
In extended version of the paper: Veríssimo, P. E., and Neves, N. F., and Correia, M. P.: Intrusion-Tolerant Architectures: Concepts and Design. In: Architecting Dependable Systems. Springer-Verlag LNCS 2677 (2003). Technical Report DI/FCUL TR03-5, Dept. of Informatics, University of Lisboa (2003). abstract - pdf

Recursive use of F. Prevention and F.Tolerance



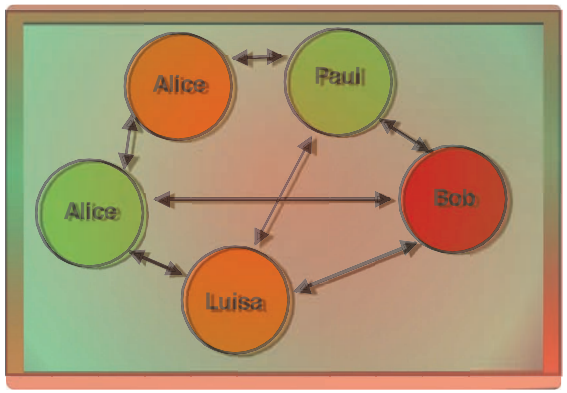
- The TTP protocol revisited
- Work at subsystem level to achieve justifiable behaviour
- Architectural hybridation w.r.t. failure assumptions

Strategies for construction of IT subsystems



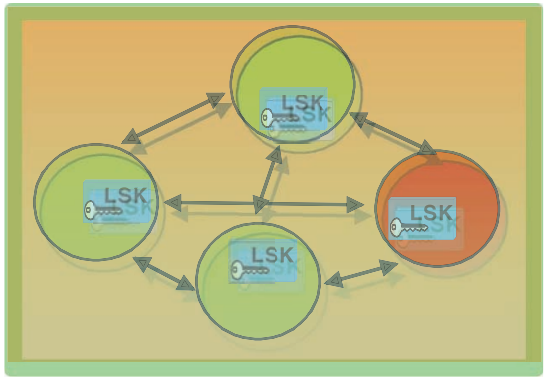
- Arbitrary model – no assumptions
- High coverage – very little to "cover"

Strategies for construction of IT subsystems



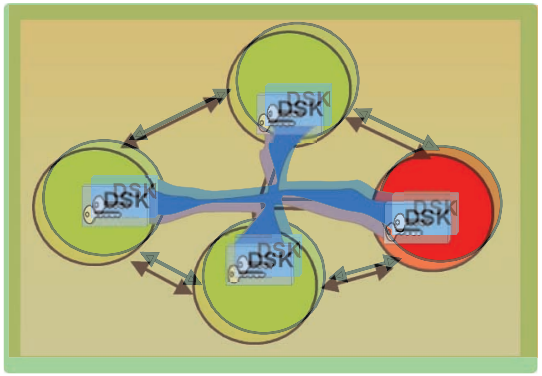
- Fail-controlled model -- unjustified environment assumptions
- Fair coverage – no enforcement

Strategies for construction of IT subsystems



- Fail-controlled model – little environment assumptions; justified component assumptions
- High coverage – enforcement by Local Trusted Comp.

Strategies for construction of IT subsystems



- Fail-controlled model – little environment assumptions; justified component assumptions
- High coverage – enforcement by Distr. Trusted Comp.

4

Advanced

Intrusion Tolerance

Paradigms

Intrusion Tolerance

Algorithmics
and the
substance of Fault Models

Recapitulating the usual path

- If you want efficient/performant solutions to F/T
  - assume controlled failure modes (omissive, fail-silent, etc.)
- If you want to build timely services (even soft R/T)
  - assume synchronous models, or at least partially sync
- Some security-related systems take this approach
  - partial synchronous environment
  - well-behaved (e.g. fortress) hosts
  - moderate level of threat in network
- They work, but only to the coverage of the assumptions
  - which must be substantiated
  - else we fall in the "*well behaved hacker syndrome*:
    - » *ello, I ll be your hacker today, here is the list of what I promise not to do.*
    - » *h thank you y the way, here are a few additional attacks we would also like you not to attempt.*

Where do we go from here?

- arbitrary failures / asynchrony thread
  - are safe, but normally inefficient
  - FLP: no deterministic solution of hard problems e.g. consensus, BA
  - does not solve timed problems (e.g., e-com, stocks, SCADA, DCS)
- controlled failures / synchrony thread
  - hard to specify for malicious faults, that brings a coverage problem
  - susceptible to attacks on timing assumptions
  - difficulty of implementation of sync. even in benign settings

Where do we go from here?

- arbitrary failures / asynchrony thread
  - are safe, but normally inefficient
  - FLP: no deterministic solution of hard problems e.g. consensus, BA
  - does not solve timed problems (e.g., e-com, stocks)
- controlled failures / synchrony thread
  - hard to specify for malicious faults, that brings a coverage problem
  - susceptible to attacks on timing assumptions
  - let alone the difficulty of implementation even in benign settings

Arbitrary failure / asynchrony assumptions

- OBJECTIVE:
  - solve most non-timed problems with high coverage
- tone down determinism:
  - randomization (Maftia/IBMZurich/Cachin-et-al)
  - semantics (+) - speed (-)
- tone down liveness expectations:
  - sacrifice liveness guarantees (MIT/Castro-Liskov)
  - termination (-) - speed (+)
- use weaker semantics
  - avoid consensus (Cornell/APSS/Schneider-et-al)
  - use quorums (Alvisi, Malki, Reiter)
  - semantics (-) - termination (+)
- Coverage:
  - very high, but still bound to crucial assumptions, such as number of failures
- Timeliness:
  - none

Controlled failure assumptions

- OBJECTIVE:
  - solve non-timed problems with high coverage
- tone down fault severity:
  - hybrid faults (IBMZurich/Cachin-et-al) (Meyer, Pradhan, Walter, Suri)
  - fault coverage (~)
- enforce hybrid behaviour ("strong" and "weak" components):
  - architectural hybridization (U.Lisboa)
  - speed (+) - termination (+) - semantics (+)
  - fault coverage (+)
- Coverage:
  - fair for hybrid fault coverage
  - can get very high if bound to the "strong" components
  - still bound to crucial assumptions, such as nr of failures
- Timeliness:
  - none

Where do we go from here?

- arbitrary failures / asynchrony thread
  - are safe, but normally inefficient
  - FLP: no deterministic solution of hard problems e.g. consensus, BA
  - does not solve timed problems (e.g., e-com, stocks)

- controlled failures / synchrony thread
  - hard to specify for malicious faults, that brings a coverage problem
  - susceptible to attacks on timing assumptions
  - let alone the difficulty of implementation even in benign settings

Controlled failures / synchrony assumptions

- OBJECTIVE:
  - solve timed and non-timed problems with high coverage

- enforce hybrid behaviour w.r.t. time:
  - protect crucial time, be indulgent with non-crucial time (U.Lisboa)
- Real-Time security kernels
  - protect time from attackers and other faults (U.Lisboa)
- indulgent timing assumptions that resist a certain level of threat
  - timely computing base (U.Lisboa)

- Coverage:
  - can get very high if bound only to the "strong" components
  - still bound to crucial assumptions, such as nr of failures
- Timeliness:
  - possible, with "strong" timed components

Taking detours…

- OBJECTIVE:
  - solve most non-timed problems with highest possible coverage

- tone down determinism (e.g., randomisation)
- tone down liveness expectations (e.g., indulgence)
- use weaker semantics (e.g., thresholds, quorums)
- tone down allowed fault severity (e.g., hybrid faults)
- tone down asynchrony (e.g., parsync protocols, FDs)

- OBJECTIVE:
  - solve timed problems with highest possible coverage

- tone down asynchrony (e.g., sync/parsync protocols)

Taking long detours…

- OBJECTIVE:
  - keep systems working long enough

(non-timed problems, arbitrary failures / asynchrony thread)

- ensuring enough replicas
- using diversity

- OBJECTIVE:
  - keep systems working long enough or in a perpetual manner

- reactive or proactive recovery (e.g., rejuvenation, refreshing)

Detours may lead to dead ends…

- f fault-tolerance means at least (n-f) correct nodes.
- Resource exhaustion: violation of a resource assumption (e.g., f+1 nodes fail), which may lead to failure
- A system is exhaustion-safe if resource exhaustion never happens.

Detours may lead to dead ends…

- An f fault-tolerant distributed system is exhaustion-safe if it terminates before f+1 faults being produced
- Obvious?
- *Impossibility of exhaustion safe asynchronous distributed systems w/ or w/o proactive recovery*

Status Quo

- In summary, taking detours has a price to pay, you lose something (performance, semantics, timeliness, coverage)
- Some detours are "so long" that you can get out of gas (exhaustion)
- With RTE systems, critical infrastructure protection, you cannot afford any of these tradeoffs, they must be:
  - highly reliable, repairing damage is costly
  - timely (R/T), let alone just timed
  - exhaustion-safe, for they will be exhausted as fast as possible
- With generic Internet systems, you have a little more slack, but not much:
  - systems are getting more demanding timeliness-wise
  - average 24/7 operation is getting a must
  - exhaustion-safety is a universal must, for its absence may compromise sheer system correctness

Shortcuts vs. detours

- Rendering the solution simpler
  (without changing the problem!)
- Architectural hybridization
- Wormholes model

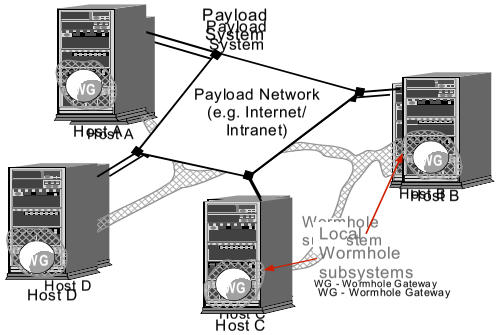
In

Paulo Veríssimo, Travelling through Wormholes: a new look at Distributed Systems Models, SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory), vol. 37, no. 1, (Whole Number 138), 2006.

Paulo Veríssimo, Uncertainty and Predictability: Can they be reconciled?, Future Directions in Distributed Computing, pp. 108-113, Springer Verlag LNCS 2584, May, 2003

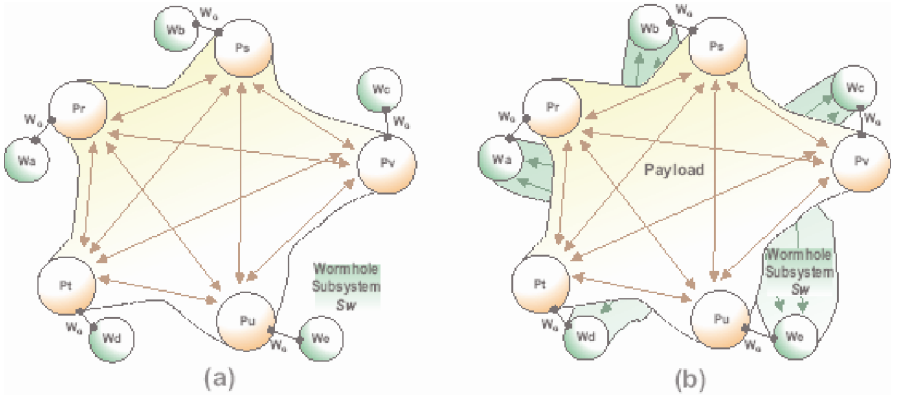
Wormholes

- New design philosophy for distributed systems:
- constructs with privileged properties which endow systems with the capability of evading the uncertainty of the environment (``taking a shortcut'') for certain crucial steps of their operation, in order to achieve the required "hard properties" (predictability)



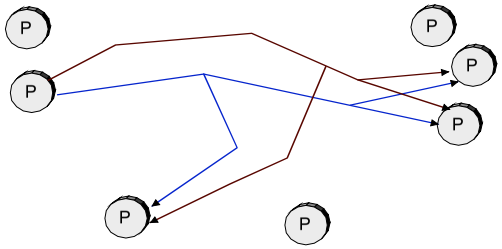
Theoretical underpinnings

- A generic hybrid distributed systems model, or Wormholes Model



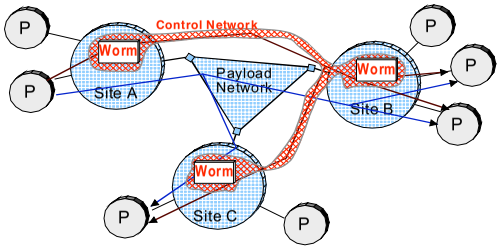
Theoretical underpinnings

- Processes and links in Wormholes models

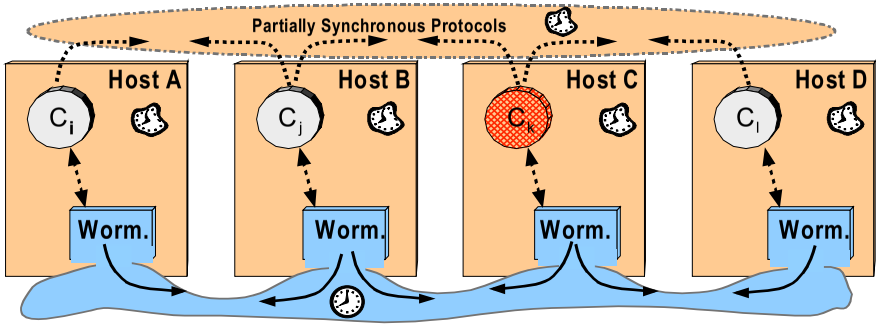


Theoretical underpinnings

- Architecture imprinting in Wormholes models



Strategy for timeliness awareness and/or assurance



- Fully synchronous, timely
- Partially synchronous, potentially untimely

Characterisation of a practical Wormhole

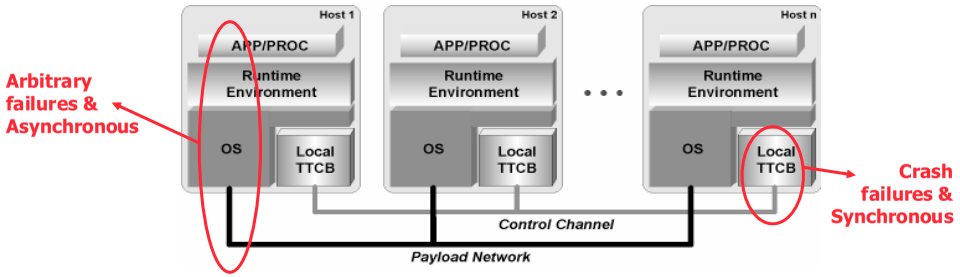
- The little part that offers 'hard' properties, e.g.:
  - synchronous: bounds on processing delays, drift rate of local clocks and delivery delay of control messages
  - secure: trusted to be tamperproof, secure processing and comms.
- Small, simple and uses few resources
  - Easier to construct and verify, with high coverage
  - Supplies simple services, like failure detection, timely execution, trusted channels, or signatures
- Acts as a coverage amplifier for the whole system

A small part of the system executes a small but critical part of its operation (a number of critical tasks) with high confidence (coverage)

Wormholes model in action

Example of deployment of a system with wormholes

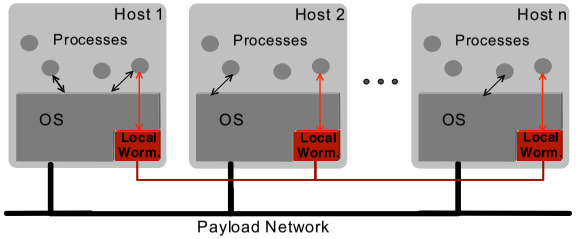
- We have played recently with two types of wormhole subsystems, to prove the concept:
  - Timely Computing Base for timeliness
  - Trusted Timely Computing Base for timeliness and security



Wormholes model in action

Example of deployment of a system with wormholes

- We have played recently with two types of wormhole subsystems, to prove the concept:
  - Timely Computing Base for timeliness
  - Trusted Timely Computing Base for timeliness and security



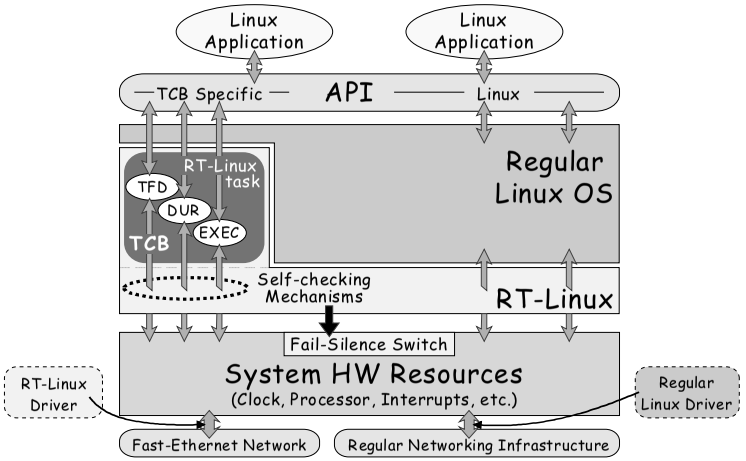
Trusted Timely Computing Base (TTCB)

- **Properties:**
  - trusted and timely execution; trusted timing failure detection
  - secure (can only fail by crashing)
  - real-time (capable of timely behavior)
  - correct processes can interact securely with the TTCB
- **TTCB can be seen as a distributed security kernel that provides a minimal set of trusted and timely services to assist the execution of fault/intrusion-tolerant algorithms, such as :**
  - provides a trusted environment for crucial steps
  - local authentication
  - agreement on a fixed sized block of data (TBA)
  - globally meaningful timestamps

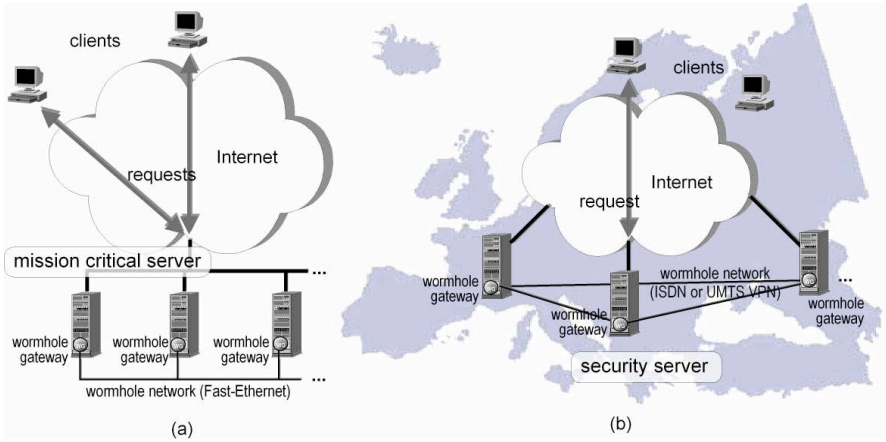
– **Can be built (there is a COTS-based prototype)**

Correia, Veríssimo, and Neves. The Design of a COTS Real-Time Distributed Security Kernel. European Dependable Computing Conf., *EDCC* , October 2002

COTS-based TCB Reference Architecture
RT-Linux example



Wormholes model in action
Example of deployment of systems with wormholes



Wormhole-Aware Byzantine Consensus Protocols

Efficient Byzantine-Resilient Reliable Multicast on a Hybrid Fault Model

Efficient Byzantine-Resilient Reliable Multicast on a Hybrid Failure Model, ***Miguel Correia, Lau Cheuk Lung, Nuno Ferreira Neves, Paulo Veríssimo.*** *Proc's of the 21st Symp. on Reliable Distributed Systems (SRDS'2002), Suita, Japan, October 2002*

Basic failure modes

- Processes can fail in a Byzantine way:
  - Crash, disobey the protocol, send contradictory messages, collude with other malicious processes,...
- Network:
  - Can corrupt packets (due to accidental faults)
  - An attacker can modify, delete, and introduce messages in the network

TTCB services

- **The *reliable multicast* protocol uses only three *TTCB services:***
- **Local authentication service**
- **Trusted block agreement**
- **Trusted absolute timestamping**

Agreement Service

- A process makes two operations:
  - propose, decide
  - this works with "small" blocks of data
- *agreement* is defined by (elist, tstart, decision)
  - elist: list of processes involved
  - tstart: instant when the TTCB stops accepting proposals
  - decision = TTCB_TBA_RMULTICAST; returns:
    - ✓ value proposed by 1st process in elist
    - ✓ mask *proposed-ok:* processes that proposed the value decided

Reliable multicast

- A reliable multicast protocol is defined formally in terms of the following properties:
- **Validity:** If a correct process multicasts a message M then some correct process in *group(M)* eventually delivers M.
- **Agreement:** If a correct process delivers a message M then all correct processes in *group(M)* eventually deliver M.
- **Integrity:** For any message M, every correct process *p* delivers M at most once and only if *p* is in *group(M)*, and if *sender(M)* is correct then M was previously multicast by *sender*(*M*).

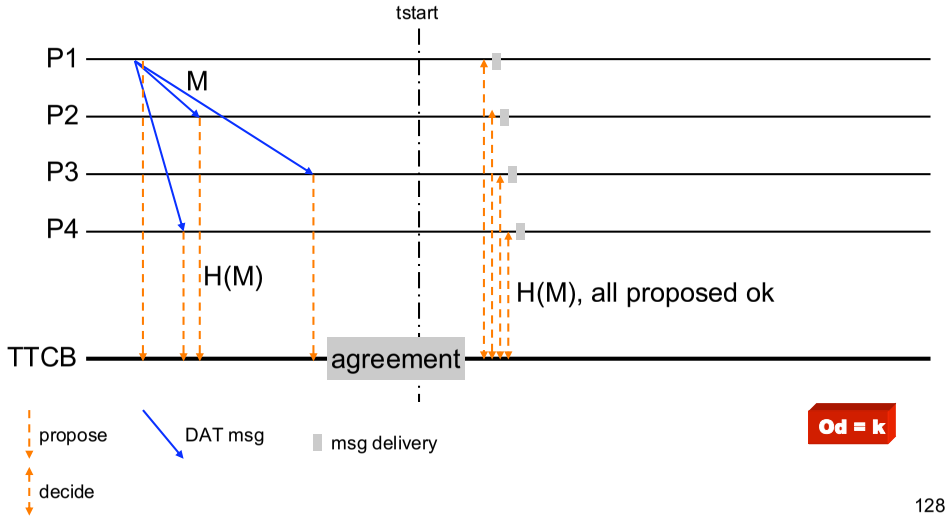
First phase

- The protocol terminates in the first phase if there are no faults or delays
- The sender:
  - sends a data message (DAT)
  - give the recipients a reliable **hash** of the message sent using the TTCB Agreement Service
- The TTCB Agreement Service acknowledges the processes that proposed the right hash
  - if all proposed the protocol terminates


```
BRM-M Sender and Recipient protocol
1  // ——— Phase 1 ———
2  if I am the sender then                          // SENDER process
3      M := (DAT, my-eid, elist, TTCB_getTimestamp() + T1, data);
4      multicast M to elist except sender; n-sends := 1;
5  else                                             // RECIPIENT processes
6      read_blocking(M); n-sends := 0;
7  propose := TTCB_propose(M.elist, M.tstart,
       TTCB_TBA_RMULTICAST, H(M));
8  do decide:=TTCB_decide(propose.tag);
       while (decide.error≠TTCB_TBA_ENDED);
9  if (decide.proposed-ok contains all recipients) then deliver M; return;
10 // ——— Phase 2 ———
11 M-deliver := ⊥;
12 mac-vector := calculate macs of (ACK, my-eid, M.elist, M.tstart,
       decide.value);
13 M-ack := (ACK, my-eid, M.elist, M.tstart, mac-vector);
14 n-acks := 0; ack-set := eids in decide.proposed-ok;
15 t-resend := TTCB_getTimestamp();
16 do
17     if (M.type = DAT) and (H(M) = decide.value) then
18         M-deliver := M;
19         ack-set := ack-set ∪ {my-eid};
20         if (my-eid ∉ decide.proposed-ok) and (n-acks < Od+1) then
21             multicast M-ack to elist except my-eid; n-acks := n-acks + 1;
22     else if (M.type = ACK) and (M.mac-vector[my-eid] is ok) then
23         ack-set := ack-set ∪ {M.sender};
24     if (M-deliver ≠ ⊥) and (TTCB_getTimestamp() ≥ t- resend) then
25         multicast M-deliver to elist except (sender and eids in ack- set);
26         t-resend := t-resend + Tresend; n-sends := n-sends + 1;
27     read_non_blocking(M); // sets M = ⊥ if no messages to be read
28 while (ack-set does not contain all recipients) and (n-sends < Od+1);
29 deliver(M-deliver);
```

Figure 2. BRM-M protocol.

Example: best case (1st phase only)

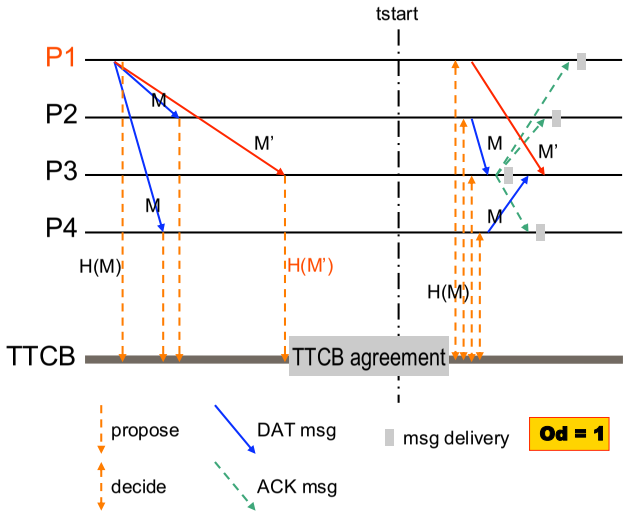


Second phase (II)

- Each process that has the message for which H(M) = value returned by the TTCB Agreement, resends M until:
  - All processes acknowledged:
    - ✓Proposing on time for the TTCB Agreement; or
    - ✓With an ACK
  - Or until it sent Od+1 times:
    - ✓Processes that do not receive are failed

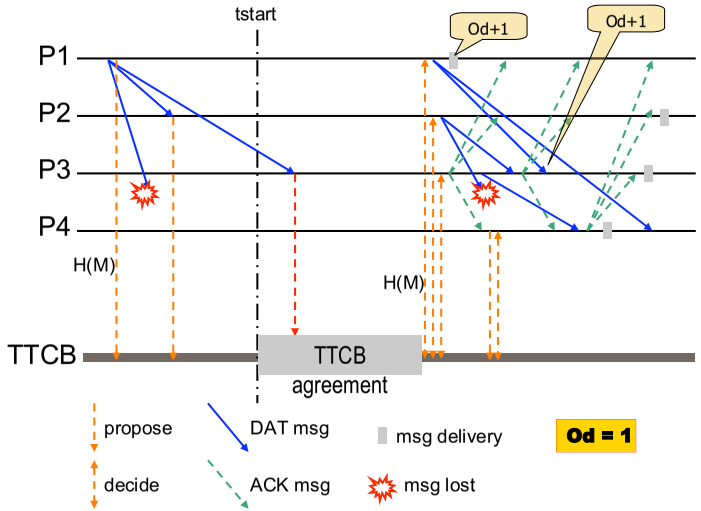
129

Example: malicious sender



130

Example: message losses/delays



131

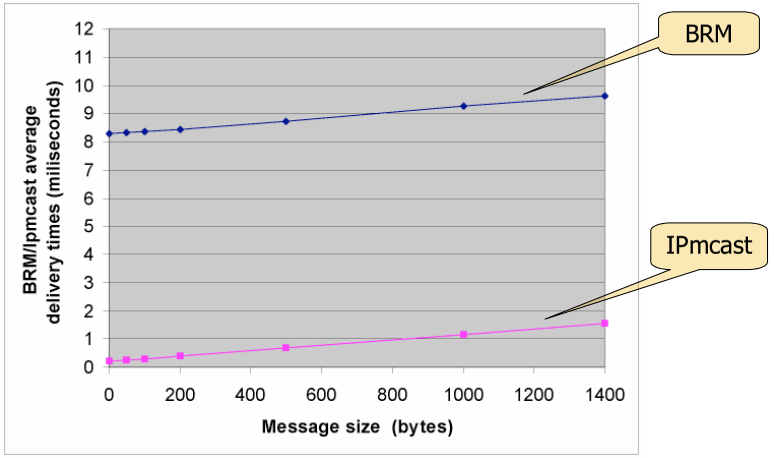
3. Protocol Performance

132

Performance evaluation scenario

- COTS-based TTCB:
  - 5 PCs: Pentium III, 450 MHz, 64 Mbytes RAM
  - Real-Time Linux
  - 2 100 Mbps Fast-Ethernet LANs (payload netw. and ctl. channel)
- Protocol implemented in C (gcc)
- Multicast using IP multicast
- Hash = MD5
- One process per host
- No failed processes
- The values are averages of 4500 measurements

Measurements



Typical values in earlier works: ~50ms

Conclusion

- Reliable multicast with Byzantine faults requires:
  - asynchronous system: n ≥ 3f+1 [Bracha&Toueg]
  - synchronous system: no limit (n ≥ f+2) [Lamport et al.]
- We follow a wormhole-aware model:
  - payload is asynchronous and byzantine-on-failure
  - TTCB is synchronous and crash-on-failure
- We achieve:
  - n ≥ f+2 without asymmetric crypto (signatures)
  - Efficiency: few phases, high performance

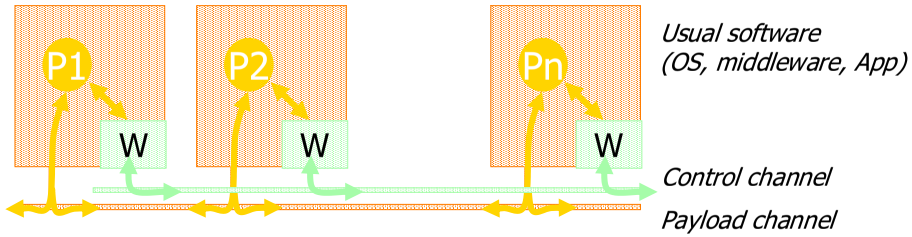
Low Complexity
Byzantine-Resilient Consensus

Low Complexity Byzantine-Resilient Consensus, ***Miguel Correia, Nuno Ferreira Neves, Paulo Veríssimo, Lau Cheuk Lung***. *Distributed Distributed Computing, vol. 17, n. 3, pp. 237--249, March 2005.*

See also:

Solving Vector Consensus with a Wormhole, ***Nuno Ferreira Neves, Miguel Correia, Paulo Veríssimo. I****EEE Transactions on Parallel and Distributed Systems, vol. 16, no. 12, pp. 1120-1131, December 2005.*

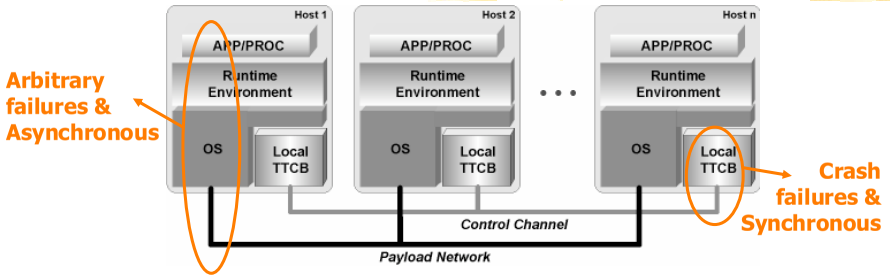
System Model with a Wormhole



Assumptions
- asynchronous system
- arbitrary failures
- at most *(n – 1) / 3* process failures

Processes can resort to the wormhole services during crucial steps of their execution

Example Wormhole: TTCB



- TTCB is a distributed security kernel that provides a minimal set of trusted and timely services, such as
  - local authentication
  - agreement on a fixed sized block of data

Solving Consensus with a TTCB Wormhole

- It leads to a solution with some nice characteristics
  - no asymmetric cripto
  - small number of message rounds
  - independence from time
- The consensus problem
  - Validity - *If all correct processes propose the same value v, then any correct process that decides, decides v*
  - Agreement - *No two correct processes decide differently*
  - Termination - *Every correct process eventually decides*

The Protocol

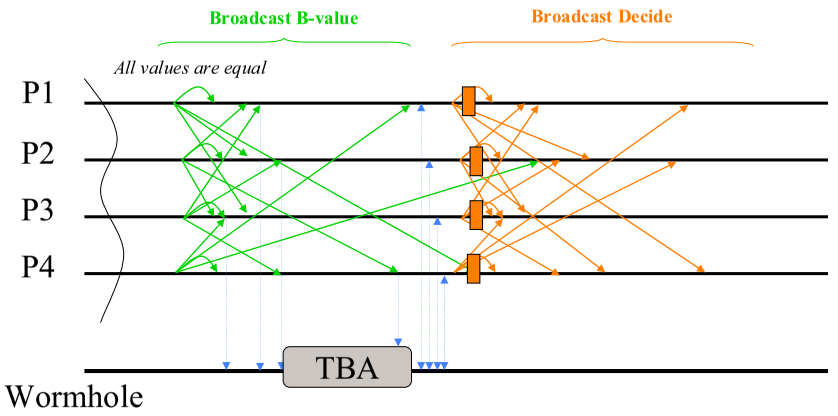
- Assumptions about payload channel messages
  - Reliably delivered (use retransmissions)
  - Integrity protected (use a MAC)
- Service provided by the wormhole
  - A low level agreement on fixed sized data (20 bytes)

Error, value, prop-ok ←
TTCB_TBA(eid, elist, aid, quorum, value)

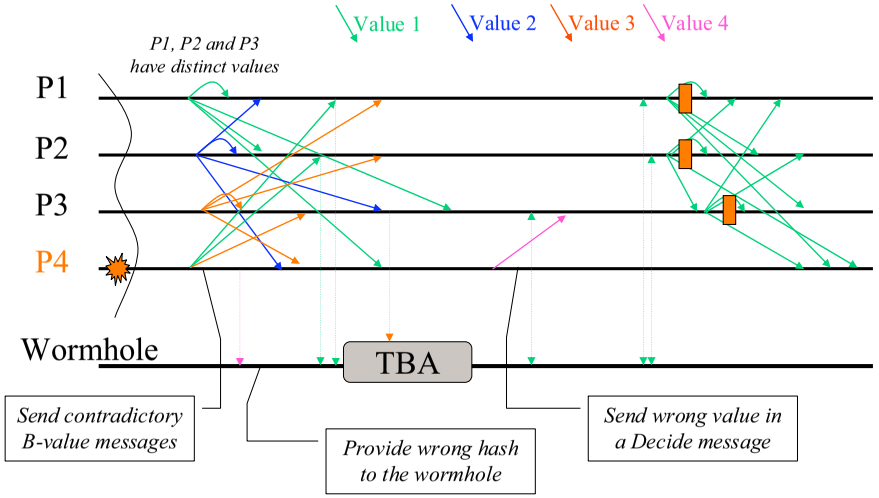
```
function consensus(elist, cid, value)
    hash-v  = ⊥
    bag = ⊥

    broadcast(B-value, i, value )
    repeat
        receive(B-value, k, value)
        bag = bag ∪ { value }
    until (bag has 2f+1 values from
    different processes)

    activate task(T1, T2)
```

```
task T1:
    v  = mostCommonValue(bag)
    out = TTCB_TBA(eid, elist, cid,
                        2f+1, Hash(v))
    if (at least f+1 proposed the
                        same value) then
        hash-v = out.value
    else
        return (default-value)

task T2:
    when (receive(Decide, k, value)) do
        bag = bag  ∪ { value }
    when (hash-v ≠ ⊥) and ( ∃ value ∈
            bag: Hash(value) = hash-v)) do
        broadcast (Decide, i, value)
        return (value)
```

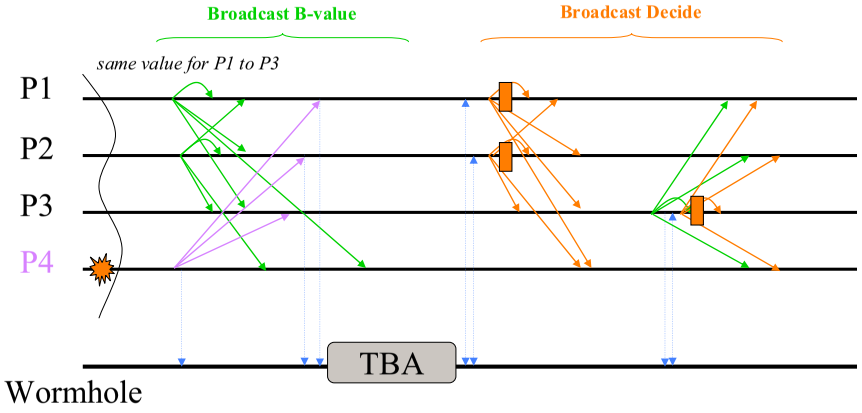

All Processes are Correct



P4 is Malicious



P3 is Delayed, P4 is Malicious



Termination & FLP result

- FLP result: *impossible to deterministically solve consensus in an asynchronous system*
- Usual solutions: *randomization, weak synchronous assumptions (e.g., partial synchronous models or unreliable failure detectors)*
- Our assumption

  2f + 1 processes eventually manage to propose before one of the tstarts

Termination & FLP result

- FLP result:
- *impossible to deterministically solve consensus in an asynchronous system*
- Usual solutions:
- *randomization, weak synchronous assumptions (e.g., partial synchronous models or unreliable failure detectors)*
- Our approach:
- *avoid violation of safety properties*
- *ensure termination by finding a way to circumvent the FLP impossibility result*
- Our assumption

  eventually there will be a round where at least 2f+1 processes manage to locally call the TTCB on time

Termination & FLP result

- Protocol executes in rounds and in each round processes attempt to propose a value to the TBA service before tstart
- If in one of the rounds enough processes are capable of providing their values on time, then they are able to complete the consensus protocol
- What happens to the remaining correct correct (but slower) processes? They will eventually manage to propose tstart, and they also terminate
- For liveness (Termination), we make a single asynchrony restriction hypothesis:

  eventually there will be a round where at least 2f+1 processes manage to call the TTCB on time
- For safety this hypothesis is not needed : if it is never verified, Agreement and Validity properties are not violated
- This hypothesis concerns local execution only, thus strictly weaker than generic restriction hypotheses (execution+comm's)

Performance Comparison

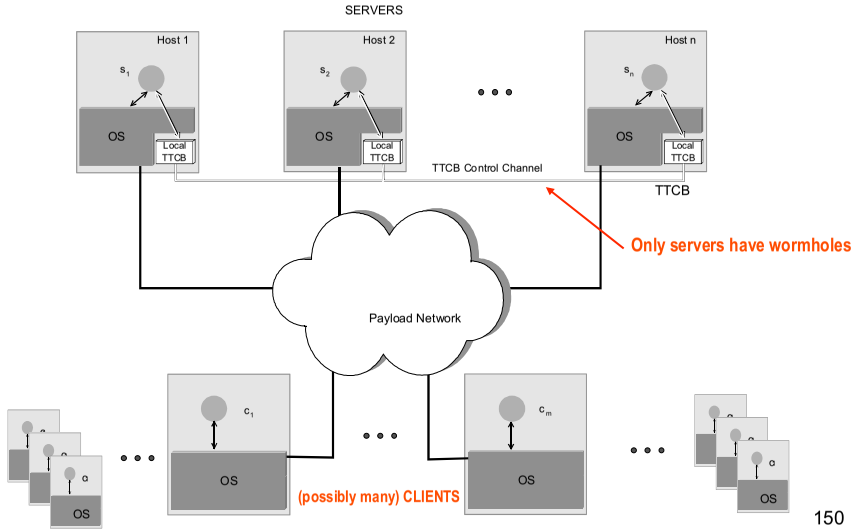
- Use *latency degree* [Schiper 97] criteria extended to include current implementation of TTCB agreement

Protocol	Latency degree	Requirements
Dwork et al.	7	
Dwork et al.	4	signed messages
Malhki & Reiter	9 or 6	signed messages
Kihlstrom et al.	4	signed messages
Block consensus	1	TTCB
General consensus	1 or 2	TTCB

State machine replication on atomic multicast

How to Tolerate Half Less One Byzantine Nodes in Practical Distributed Systems. ***Miguel Correia, Nuno Ferreira Neves, Paulo Veríssimo***. *In Proceedings of the 23rd IEEE Symposium on Reliable Distributed Systems. Florianopolis, Brasil, pages 174-183, October 2004.*

System architecture



Main Achievements

- First SMA service for practical byzantine distributed systems with resilience f out of 2f+1
  - Lower number of replicas reduces cost of hardware + cost of designing different replicas (for fault independence)
- Low time complexity
- Probable good performance since it does not resort to public key cryptography

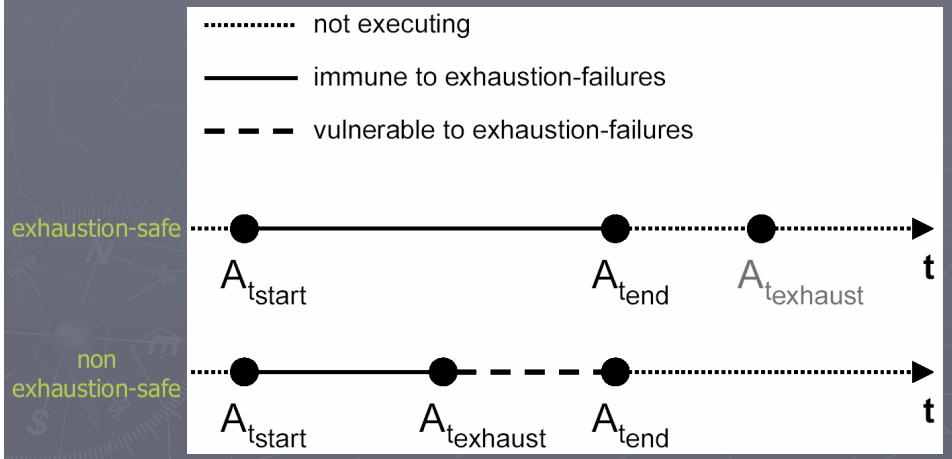
Exhaustion safety and impossibility of asynchronous proactive recovery

How Resilient are Distributed f Fault/Intrusion-Tolerant Systems? **Paulo Sousa, Nuno Ferreira Neves, Paulo Veríssimo**. In Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05). Yokohama, Japan, pages 98-107, June 2005.

Focusing on Resources

- Fault and timing assumptions are an abstraction of the required resources.
  - e.g., f fault-tolerance means (n-f) correct nodes are required.
- Resource exhaustion: violation of a resource assumption.
  - e.g., f+1 nodes fail.
- Definition: An exhaustion-failure is a failure that results from resource exhaustion.
- Definition: A system is exhaustion-safe if it ensures that exhaustion-failures never happen.

To Be or Not to Be Exhaustion-Safe



Async Proactive Recovery

- How to guarantee that rejuvenations always terminate before resource exhaustion?
  - Rejuvenation start instant may be delayed.
  - Rejuvenation actions may be delayed.
  - These delays may be enforced by a malicious adversary!
- Async proactive recovery does not guarantee exhaustion-safety.
  - namely, in a malicious environment.

Detours may lead to dead ends...

- An f fault-tolerant distributed system is exhaustion-safe if it terminates before f+1 faults being produced
- Thus:
- *Impossibility of exhaustion-safe asynchronous distributed systems (w/ or w/o proactive recovery)*
- Obvious?

From Theory to Practice (1)

- CODEX (COrnell Data EXchange) is a recent distributed service for storage and dissemination of secrets [Marsh and Schneider, 2004].
  - Assumptions: async model, malicious adversary.
- CODEX private key is shared by CODEX servers using threshold cryptography. (Each server has a share of the key).
  - Shares are periodically refreshed through an asynchronous proactive secret sharing protocol (APSS).
  - Key is compromised if an adversary collects

From Theory to Practice (2)

- CODEX in more detail
  - n servers share the private key using an (n, f+1) secret sharing scheme
    - f+1 shares are sufficient to recover the key.
    - less than f+1 shares give no knowledge about the key.
  - Assumption: at most f≤(n-1)/3 servers "are compromised at any time".
    - Excludes the possibility of an adversary controlling f+1 servers simultaneously, but "it does not rule out the adversary compromising one server and learning the CODEX private key share stored there, being evicted, compromising another, and ultimately learning f+1 shares".
      (mobile virus attack [Ostrovsky & Yung, 1991])

From Theory to Practice (3)

- APSS: CODEX defense against mobile virus attacks
  - APSS is periodically executed.
    - triggered by a local timer or by a remote notification.
  - "a mobile virus attack would succeed only if it is completed in the interval between successive executions of APSS, and this interval can be as short as a few minutes".

The problem

- when safety of an asynchronous system depends on non-substantiated timing assumptions
  - clocks with bounded rate of deviation to real-time
  - capacity of performing periodic (tiumely) executions
  - these assumptions can be violated either in the assumed async environment and/or by a malicious adversary.

From Theory to Practice (4)

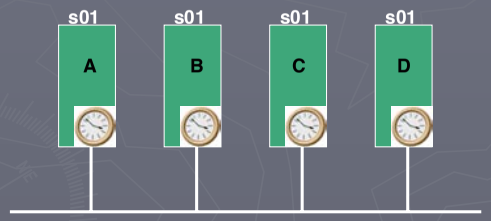
- An attack that compromises safety:
  - Two adversaries: ADV1 and ADV2.
  - Step 1: ADV1 performs a mobile virus attack against f+1 servers
    - slows the clock rate of each server.
  - Step 2: ADV1 temporally cuts off the links between the f+1 servers and the rest of the system.

From Theory to Practice (5)

- An attack that compromises CODEX safety:
  - Step 3: ADV2 performs a mobile virus attack against the same f+1 servers
    - learns, one by one, f+1 private key shares.
    - no rejuvenation occurs in between because in step 1 clocks are made as slow as needed.
  - Step 4: ADV2 discloses CODEX private key by combining the f+1 shares.
  - Important Note: ADV1 actions simply enforce a behavior that can occur in any fault-free async system.

From Theory to Practice (6)

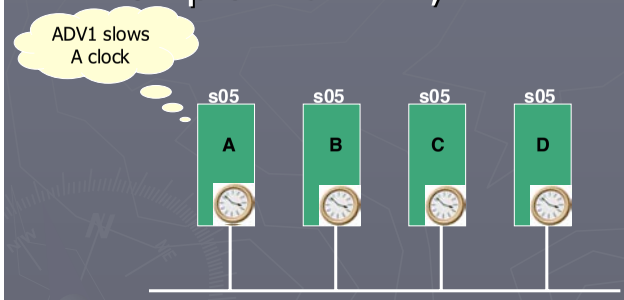
- Example with n=4, f=1



sXX: share version

From Theory to Practice (6)

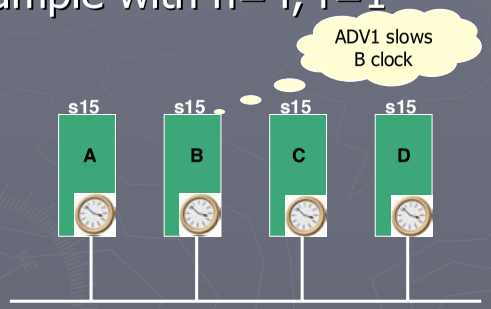
- Example with n=4, f=1



sXX: share version

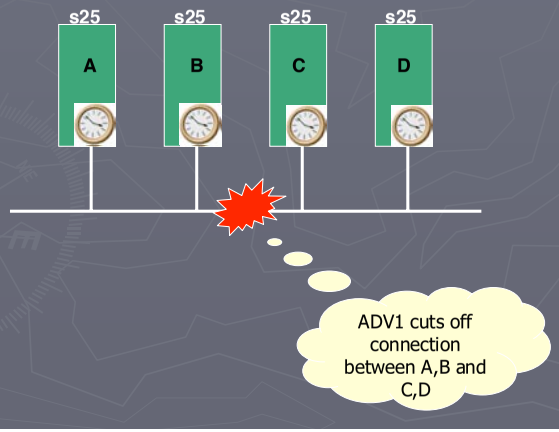
From Theory to Practice (6)

- Example with n=4, f=1



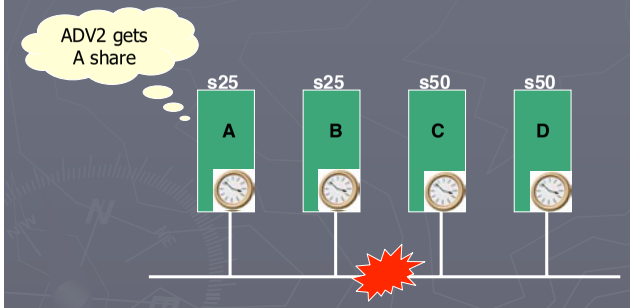
From Theory to Practice (6)

- Example with n=4, f=1



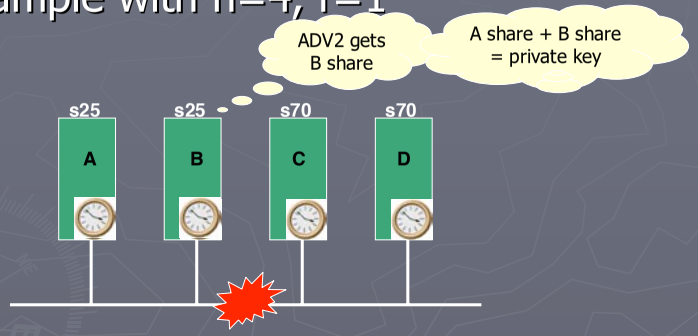
From Theory to Practice (6)

- Example with n=4, f=1



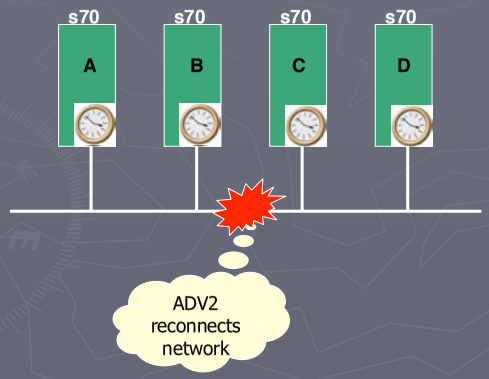
From Theory to Practice (6)

- Example with n=4, f=1



From Theory to Practice (6)

- Example with n=4, f=1



sXX: share version

Conclusions

- Current state-of-the-art does not allow to construct exhaustion-safe distributed systems in face of arbitrary faults.
  - Sync systems are vulnerable:
    - timing failures.
  - Async systems are vulnerable:
    - max number of faults + unbounded execution time.
  - Async systems with async proactive recovery are vulnerable:
    - max number of faults + unbounded rejuvenation period.

Ongoing Work

- Combining proactive recovery and wormholes
  - Proactive recovery is useful to postpone $t_{exhaust}$ as long as it has timeliness guarantees.
  - Proposal: combine async payload system with sync proactive recovery subsystem.
    - See recent tech reports and papers in our page
      - E.g. Proactive Resilience through Architectural Hybridization DI/FCUL TR 05-8, May 2005.

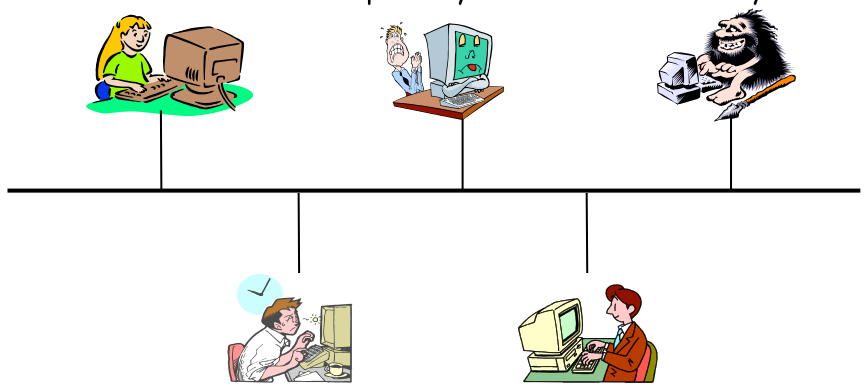
MAFTIA
Reference Intrusion-Tolerance Architecture

In D. Powell and R. Stroud, eds., ***MAFTIA Conceptual Model and Architecture,*** Deliv. D2, Project MAFTIA IST-1999-11583, Tech.Rep. DI/FCUL TR-01-10, Univ. Lisboa Nov. 2001. http://www.di.fc.ul.pt/tech-reports/abstract01-10.html

MAFTIA - Malicious and Accidental Fault Tolerance for Internet Applications

http://www.maftia.org

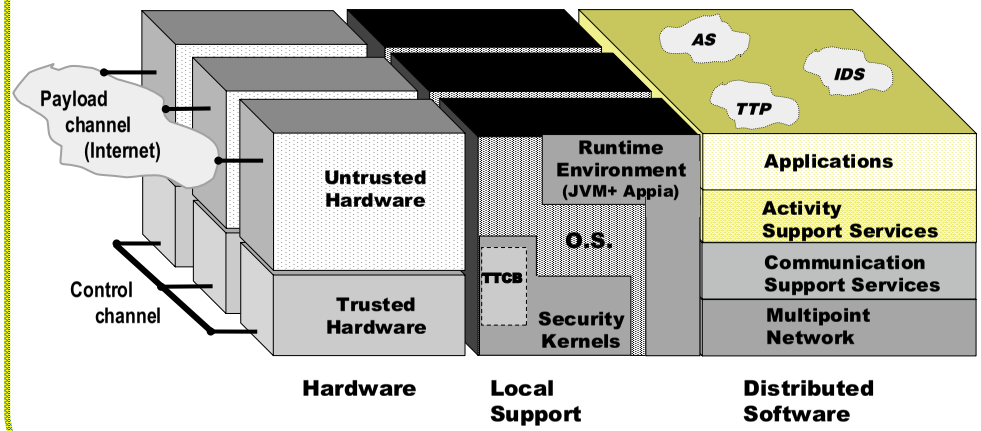
Computer systems can fail for many reasons



MAFTIA investigated ways of making computer systems automatically resist both accidental and malicious faults

Architecture Overview

Host architecture

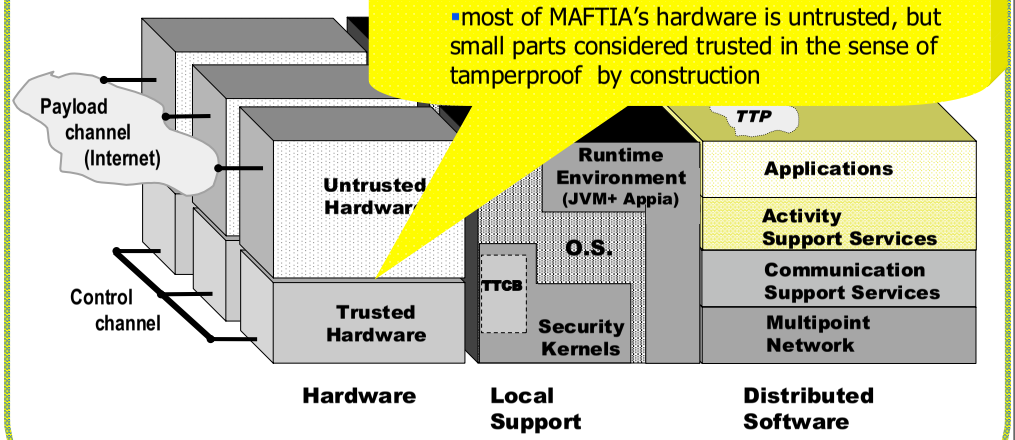


AS - Authorisation Service, IDS - Intrusion Detection Service, TTP - Trusted Third Party Service

Architecture Overview

Host architecture

- trusted— vs. untrusted— hardware
  - most of MAFTIA's hardware is untrusted, but small parts considered trusted in the sense of tamperproof by construction

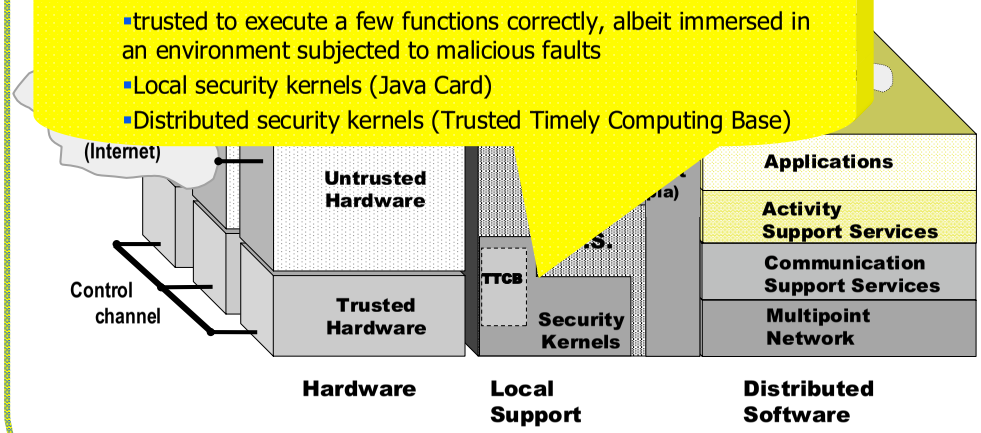


AS - Authorisation Service, IDS - Intrusion Detection Service, TTP - Trusted Third Party Service

Architecture Overview

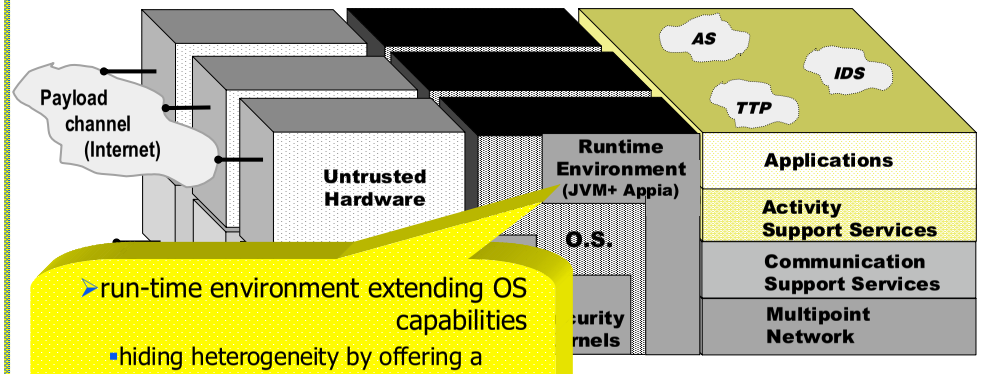
Host architecture

- security kernels materialising fail-controlled subsystems
  - trusted to execute a few functions correctly, albeit immersed in an environment subjected to malicious faults
  - Local security kernels (Java Card)
  - Distributed security kernels (Trusted Timely Computing Base)



AS - Authorisation Service, IDS - Intrusion Detection Service, TTP - Trusted Third Party Service

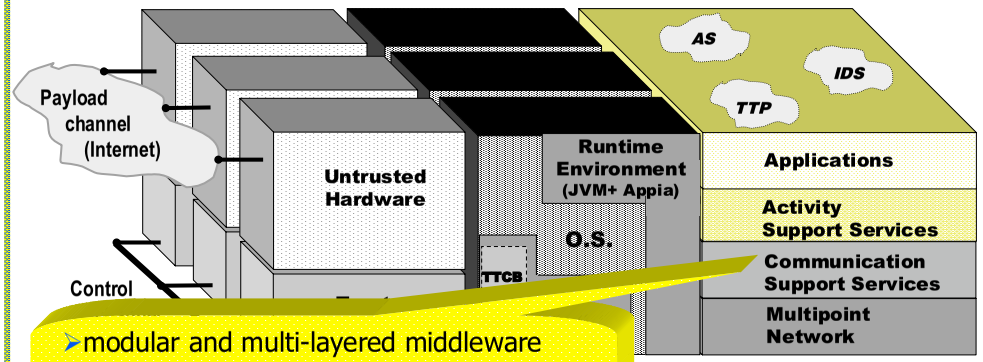
Architecture Overview
Host architecture



- run-time environment extending OS capabilities
  - hiding heterogeneity by offering a homogeneous API and framework for protocol composition

AS - Authorisation Service, IDS - Intrusion Detection Service, TTP - Trusted Third Party Service

Architecture Overview
Host architecture

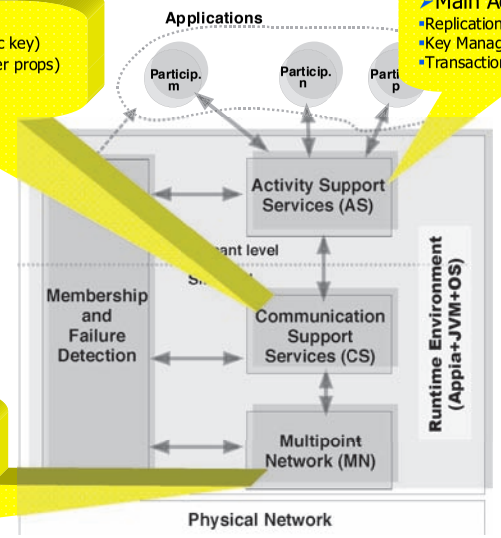


- modular and multi-layered middleware
- neat separation between different functional blocks

AS - Authorisation Service, IDS - Intrusion Detection Service, TTP - Trusted Third Party Service

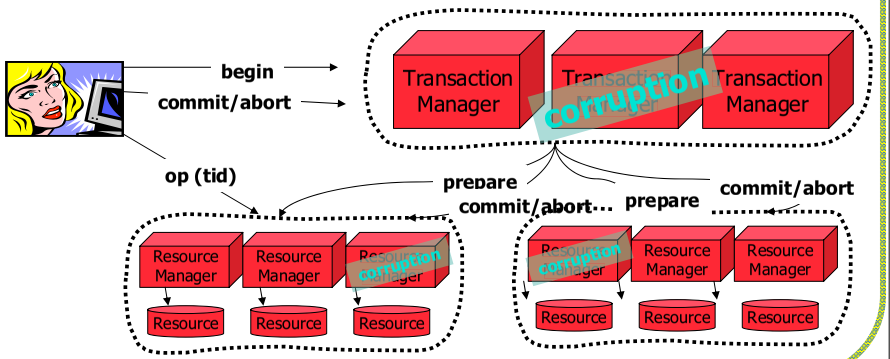
Modular Group Architecture

- Communication Services
  - Distributed Cryptography (threshold public key)
  - Group Communication (reliability and order props)
  - Byzantine Agreement
  - Time and Clock Synchronisation
- Main Activity Services
  - Replication management
  - Key Management
  - Transactional Management
- Multipoint Network
  - Multipoint addressing and routing
  - Basic secure channels and envelopes
  - Management Communication prots
  - Appia APIs for mcastIP, Ipsec, SNMP

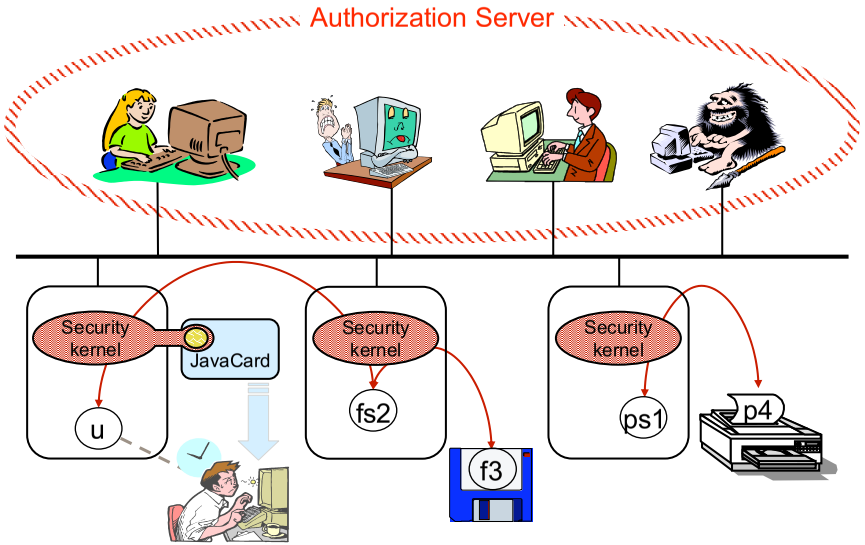


IT Transactions with Error Masking

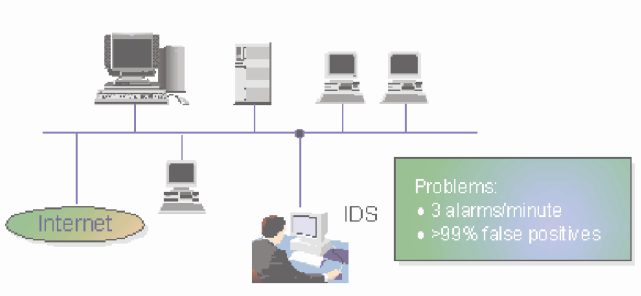
- A CORBA-style transaction service, standard ACID properties
- Support for multiparty transactions
- Uses error masking to tolerate intrusions
- Application of hybrid failure assumptions



IT Authorisation Service



IT Intrusion Detection Service



- finding solutions to the problems of the high rate of false positive and false negative alarms generated by existing solutions
- these false alarms can also be due to attacks against the IDS itself, therefore the need to design an IDS which is itself tolerant to intrusions
- study and evaluate how notions such as fault injection, diversity and distributed reasoning can address the weaknesses of existing solutions

OASIS

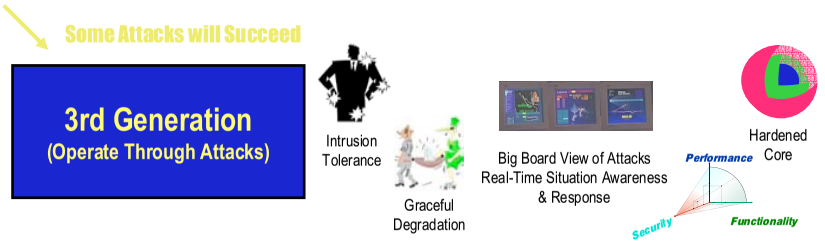
ORGANICALLY ASSURED & SURVIVABLE INFORMATION SYSTEMS

DARPA

OASIS

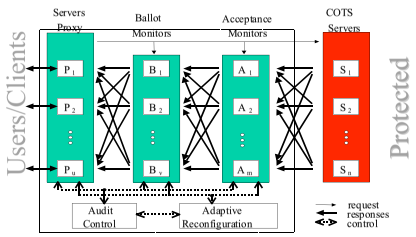
ORGANICALLY ASSURED & SURVIVABLE INFORMATION SYSTEMS

Dr. Jaynarayan Lala – jlala@darpa.mil, 703-696-7441
Organically Assured Survivable Information Systems,
OASIS Demonstration and Validation Program



OASIS

Intrusion Tolerant Architecture



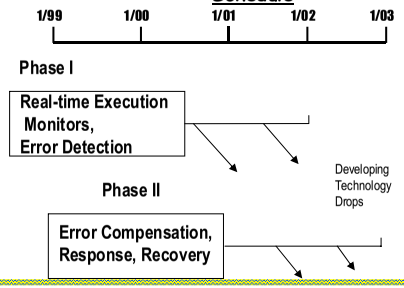
Objectives

- Construct intrusion-tolerant architectures from potentially vulnerable components
- Characterize cost-benefits of intrusion tolerance mechanisms
- Develop assessment and validation methodologies to evaluate intrusion tolerance mechanisms

Technical Approach

- Real-Time Execution Monitors: In-line reference monitors, wrappers, sandboxing, binary insertion in legacy code, proof carrying code, secure mobile protocols
- Error Detection & Tolerance Triggers: Time and Value Domain Checks, Comparison and Voting, Rear Guards
- Error Compensation, Response and Recovery: Hardware and Software Redundancy, Rollback and Roll-Forward Recovery
- Intrusion Tolerant Architectures: Design Diversity, Randomness, Uncertainty, Agility
- Assessment & Validation: Peer Review Teams, Red Team, Assurance Case (Fault Tree, Hazard Analysis, Formal Proofs, Analytical Models, Empirical Evidence)

Schedule



Further Reading

- L. Alvisi, D. Malkhi, E. Pierce, M. K. Reiter, and R. N. Wright, Dynamic Byzantine quorum systems," in Proc. Int'l Conference on Dependable Sys and Networks (FTCS-30/DCCA-8), pp. 283-292, 2000.
- Y. Amir et al. Secure group communication in asynchronous networks with failures: Integration and experiments. In Proc. The 20th IEEE International Conference on Distributed Computing Systems (ICDCS 2000), pages 330-343, Taipei, Taiwan, April 2000.
- G. Ateniese, M. Steiner, G. Tsudik: Authenticated group key agreement and friends. In Proceedings of the 5th ACM Conference on Computer and Communications Security (CCS-98), pages 17-26, New York, November 3-5 1998. ACM Press.
- Giuseppe Ateniese, Michael Steiner, and Gene Tsudik. New multi-party authentication services and key agreement protocols. IEEE Journal of Selected Areas on Communications, 18, March 2000.
- Christian Cachin. Distributing Trust on the Internet. In Procs. of the Int'l Conf. on Depend. Systems and Networks (DSN-2002), Gotteborg, Sweden, 2001.
- C. Cachin, K. Kursawe and V. Shoup, "Random oracles in Constantinople: Practical asynchronous Byzantine agreement using cryptography", in Proc. 19th ACM Symposium on Principles of Distributed Computing (PODC), pp.123-32, 2000b.
- C. Cachin and J. A. Poritz, Hydra: Secure replication on the Internet," In Procs. of the Int'l Conf. on Dependable Systems and Networks (DSN-2002), Washigton, USA, 2002.
- M. Castro and B. Liskov, Practical Byzantine fault tolerance," in Proc. Third Symp. Operating Systems Design and Implementation (OSDI), 1999.
- T. D. Chandra and S. Toueg, Unreliable failure detectors for reliable distributed systems," Journal of the ACM, vol. 43, no. 2, pp. 225-267, 1996.
- Nick Cook, Santosh Shrivastava, Stuart Wheater. Distributed Object Middleware to Support Dependable Information Sharing between Organisations. In Procs. of the Int'l Conf. on Dependable Systems and Networks (DSN-2002), Washigton, USA, 2002.

Further Reading

- Miguel Correia, Nuno Ferreira Neves, Paulo Veríssimo, Lau Cheuk Lung, Low Complexity Byzantine-Resilient Consensus, Distributed Computing, vol. 17, n. 3, pp. 237--249, March 2005.
- M. Correia, Lau Cheuk Lung, Nuno Ferreira Neves, and P. Veríssimo. Efficient Byzantine-Resilient Reliable Multicast on a Hybrid Failure Model. In Proc. of Symp. of Reliable Distributed Systems, October 2002, Japan.
- Miguel Correia, Paulo Veríssimo, and Nuno~Ferreira Neves. The architecture of a secure group communication system based on intrusion tolerance. In International Workshop on Applied Reliable Group Communication, Phoenix, Arizona, USA, April 2001.
- M. Correia, P. Veríssimo, and N. F. Neves. The design of a COTS real-time distributed security kernel. In proceedings of the EDCC-4, Fourth European Dependable Computing Conference, Toulouse, France - October 23-25, 2002.
- H. Debar, M. Dacier, A. Wespi: Towards a taxonomy of intrusion detection systems. Computer Networks, 31:805-822, 1999.
- Y. Desmedt: Society and group oriented cryptography: a new concept; Crypto '87, LNCS 293, Springer-Verlag, Berlin 1988, 120-127.
- Y. Desmedt, Threshold cryptography," European Transactions on Telecommunications, vol. 5, no. 4, pp. 449-457, 1994.
- Y. Deswarte, N. Abghour, V. Nicomette and D. Powell, "An internet authorization scheme using smart card-based security kernels", in Int'l Conf. on Research in Smart Cards (E-smart 2001), (Cannes, France), Lecture Notes in Computer Science, pp.71-82, Springer-Verlag, 2001.
- Y. Deswarte, L. Blain, J.-C. Fabre: Intrusion tolerance in distributed systems. In Proc. Symp. on Research in Security and Privacy, pages 110-121, Oakland, CA, USA, 1991. IEEE CompSoc Press.
- Durward McDonell, Brian Niebuhr, Brian Matt, David L. Sames, Gregg Tally, Szu-Chien Wang, Brent Whitmore. Developing a Heterogeneous Intrusion Tolerant CORBA System. In Procs. of the Int'l Conf. on Dependable Systems and Networks (DSN-2002), Washigton, USA, 2002.

Further Reading

- Bruno Dutertre, Hassen Saïdi and Victoria Stavridou. Intrusion-Tolerant Group Management in Enclaves. In Procs. of the Int'l Conf. on Dependable Systems and Networks (DSN-2001), Gotteborg, Sweden, 2001.
- J. Fraga and D. Powell, "A Fault and Intrusion-Tolerant File System", in IFIP 3rd Int. Conf. on Computer Security, (J. B. Grimson and H.-J. Kugler, Eds.), (Dublin, Ireland), Computer Security, pp.203-18, Elsevier Science Publishers B.V. (North-Holland), 1985.
- R. Guerraoui, M. Hurn, A. Mostefaoui, R. Oliveira, M. Raynal, and A. Schiper, Consensus in asynchronous distributed systems: A concise guided tour," in Advances in Distributed Systems (S. Krakowiak and S. Shrivastava, eds.), vol. 1752 of LNCS, pp. 33-47, Springer, 2000.
- V. Hadzilacos and S. Toueg, Fault-tolerant broadcasts and related problems," in Distributed Systems (S. J. Mullender, ed.), New York: ACM Press & Addison-Wesley, 1993. An expanded version as Technical Report TR94-1425, Department of Computer Science, Cornell University, Ithaca NY, 1994.
- HariGovind V Ramasamy, Prashant Pandey, James Lyons, Michel Cukier, William H. Sanders. Quantifying the Cost of Providing Intrusion Tolerance in Group Communication Systems, In Procs. of the Int'l Conf. on Dependable Systems and Networks (DSN-2002), Washigton, USA, 2002.
- Matti A. Hiltunen, Richard D. Schlichting and Carlos A. Ugarte. Enhancing Survivability of Security Services Using Redundancy. In Procs. of the Int'l Conf. on Dependable Systems and Networks (DSN-2002), Gotteborg, Sweden, 2001.
- K. P. Kihlstrom, L. E. Moser, and P. M. Melliar-Smith, The SecureRing protocolsfor securing group communication," in Proc. 31st Hawaii Int'l Conf. on System Sciences, pp. 317-326, IEEE, Jan. 1998.
- J. H. Lala, "A Byzantine Resilient Fault-Tolerant Computer for Nuclear Power Plant Applications", in 16th IEEE Int. Symp. on Fault Tolerant Computing (FTCS-16), (Vienna, Austria), pp.338-43, IEEE Computer Society Press, 1986.

Further Reading

- B. Madan, K. Goseva-Popstojanova, K. Vaidyanathan, K. Trivedi. Modeling and Quantification of Security Attributes of Software Systems. In Procs. of the Int'l Conf. on Dep. Syst. and Networks (DSN-2002), Washigton, USA, 2002.
- D. Malkhi and M. K. Reiter, An architecture for survivable coordination in large distributed systems," IEEE Transactions on Knowledge and Data Engineering, vol. 12, no. 2, pp. 187-202, 2000.
- Jean-Philippe Martin, Lorenzo Alvisi, Michael Dahlin. Small Byzantine Quorums. In Procs. of the Int'l Conf. on Dependable Systems and Networks (DSN-2002), Washigton, USA, 2002.
- Roy A. Maxion and Tahlia N. Townsen, Masquerade Detection Using Truncated Command Lines. In Procs. of the Int'l Conf. on Dep. Syst. and Networks (DSN-2002), Washigton, USA, 2002.
- F. Meyer and D. Pradhan, "Consensus with Dual Failure Modes," presented at The 17th International Symposium on Fault-Tolerant Computing Systems, Pittsburgh, PA, 1987, pp. 214--22.
- L. E. Moser, P. M. Melliar-Smith, and N. Narasimhan. The SecureGroup communication system. In Proceedings of the IEEE Information Survivability Conference, pages 507–516, January 2000.
- Peter G. Neumann, "Practical Architectures for Survivable Systems and Networks," Computer Science Laboratory, SRI International, Menlo Park, CA, Technical Report http://www.csl.sri.com/~neumann/private/arldraft.{pdf|ps}, October 1998.
- Nuno Ferreira Neves, Miguel Correia, Paulo Veríssimo, Solving Vector Consensus with a Wormhole IEEE Transactions on Parallel and Distributed Systems, vol. 16, no. 12, pp. 1120-1131, Dec. 2005.
- Birgit Pfitzmann and Michael Waidner. Composition and integrity preservation of secure reactive systems. 7th ACM Conference on Computer and Communications Security, Athens, November 2000, ACM Press, New York 2000, 245-254.
- P. Porras, D. Schnackenberg, S. Staniford-Chen and M. Stillman, "The Common Intrusion Detection Framework Architecture", CIDF working group, http://www.gidos.org/drafts/architecture.txt, (accessed: 5 September, 2001).
- D. Powell, G. Bonn, D. Seaton, P. Veríssimo and F. Waeselynck, "The Delta-4 Approach to Dependability in Open Distributed Computing Systems", in 18th IEEE Int. Symp. on Fault-Tolerant Computing Systems (FTCS-18), (Tokyo, Japan), pp.246-51, IEEE Computer Society Press, 1988.

Further Reading

- M. K. Reiter: Distributing trust with the Rampart toolkit; Communications of the ACM, 39/4 (1996).
- F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: a tutorial", ACM Computing Surveys, 22 (4), pp.299-319, 1990.
- Paulo Sousa, Nuno Ferreira Neves, Paulo Veríssimo, How Resilient are Distributed f Fault/Intrusion-Tolerant Systems?. In Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05). Yokohama, Japan, pages 98-107, June 2005.
- P. Verissimo, A. Casimiro and C. Fetzer, "The Timely Computing Base: Timely Actions in the Presence of Uncertain Timeliness", in Proc. of DSN 2000, the Int. Conf. on Dependable Systems and Networks, pp.533-52, IEEE/IFIP, 2000.
- Paulo Veríssimo, Nuno~Ferreira Neves, and Miguel Correia. The middleware architecture of MAFTIA: A blueprint. In Proceedings of the IEEE Third Information Survivability Workshop (ISW-2000), Boston, Massachusetts, USA, October 2000.
- Paulo Veríssimo, Travelling through Wormholes: a new look at Distributed Systems Models, SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory), vol. 37, no. 1, (Whole Number 138), 2006.
- Paulo Veríssimo, Uncertainty and Predictability: Can they be reconciled?, Future Directions in Distributed Computing, pp. 108-113, Springer Verlag LNCS 2584, May, 2003
- Chenxi Wang, Jack Davidson, Jonathan Hill and John Knight. Protection of Software-Based Survivability Mechanisms. In Procs. of the Int'l Conf. on Dependable Systems and Networks (DSN-2002), Gotteborg, Sweden, 2001.
- J.~Xu, A.~Romanovsky, and B.~Randell. Concurrent exception handling and resolution in distributed object systems. IEEE Trans. on Parallel and Distributed Systems, 10(11):1019--1032, 2000.
- L. Zhou, F. B. Schneider, and R. van Renesse, COCA: A secure distributed online certification authority,Tech. Rep. 2000-1828, CS Dpt, Cornell University, Dec. 2000. Also ACM TOCS to appear.

Where to find us

- ***Navigators Group***

  http://www.navigators.di.fc.ul.pt

- **Intrusion Tolerance related research in the site:**
- *Look-up our research in the Fault and Intrusion Tolerance in Open Distributed Systems research line*

- **Feel free to email:**
- **Paulo:** *pjv@di.fc.ul.pt*

Intrusion-Tolerant Architectures: Concepts and Design *

Paulo Esteves Veríssimo, Nuno Ferreira Neves, Miguel Pupo Correia

Univ. of Lisboa, Faculty of Sciences
Bloco C5, Campo Grande, 1749-016 Lisboa - Portugal
{pjv,nuno,mpc}@di.fc.ul.pt, http://www.navigators.di.fc.ul.pt

Abstract. There is a significant body of research on distributed computing architectures, methodologies and algorithms, both in the fields of fault tolerance and security. Whilst they have taken separate paths until recently, the problems to be solved are of similar nature. In classical dependability, fault tolerance has been the workhorse of many solutions. Classical security-related work has on the other hand privileged, with few exceptions, intrusion prevention. Intrusion tolerance (IT) is a new approach that has slowly emerged during the past decade, and gained impressive momentum recently. Instead of trying to prevent every single intrusion, these are allowed, but tolerated: the system triggers mechanisms that prevent the intrusion from generating a system security failure. The paper describes the fundamental concepts behind IT, tracing their connection with classical fault tolerance and security. We discuss the main strategies and mechanisms for architecting IT systems, and report on recent advances on distributed IT system architectures.

1 Introduction

There is a significant body of research on distributed computing architectures, methodologies and algorithms, both in the fields of dependability and fault tolerance, and in security and information assurance. These are commonly used in a wide spectrum of situations: information infrastructures, some of them critical; commercial web-based sites; embedded systems. Their operation has always been a concern, namely presently, due to the use of COTS, compressed design cycles, openness. Whilst they have taken separate paths until recently, the problems to be solved are of similar nature: keeping systems working correctly, despite the occurrence of mishaps, which we could commonly call faults (accidental or malicious); ensure that, when systems do fail (again, on account of accidental or malicious faults), they do so in a non harmful/catastrophic way. In classical dependability, and mainly in distributed settings, fault tolerance has been the workhorse of the many solutions published over the years. Classical security-related work has on the other hand privileged, with few exceptions, intrusion prevention, or intrusion detection without systematic forms of processing the intrusion symptoms.

A new approach has slowly emerged during the past decade, and gained impressive momentum recently: intrusion tolerance (IT) [1]. That is, the notion of handling— react, counteract, recover, mask— a wide set of faults encompassing intentional and malicious faults (we may collectively call them intrusions), which may lead to failure of the system security properties if nothing is done to counter their effect on the system state. In short, instead of trying to prevent every single intrusion, these are allowed, but tolerated: the system has the means to trigger mechanisms that prevent the intrusion from generating a system failure.

It is known that distribution and fault tolerance go hand in hand: one distributes to achieve resilience to common mode faults, and/or one embeds fault tolerance in a distributed system to resist the higher fault probabilities coming from distribution. Contrary to some vanishing misconceptions, security and distribution also go hand in hand: one splits and separates information and processing geographically, making life harder to an attacker. This suggests that (distributed) malicious fault tolerance, a.k.a. (distributed) intrusion tolerance is an obvious approach to achieve secure processing. If this is so obvious, why has it not happened earlier?

In fact, the term "intrusion tolerance" has been used for the first time in [19], and a sequel of that work lead to a specific system developed in the DELTA-4 project [16]. In the following years, a number of isolated works, mainly on protocols, took place that can be put under the IT umbrella [10, 31, 22, 2, 24, 4, 21], but only recently did the area develop explosively, with two main projects on both sides of the Atlantic, the OASIS and the MAFTIA projects, doing structured work on concepts, mechanisms and architectures. One main reason is concerned with the fact that distributed systems present fundamental problems in the presence of malicious faults. On the other hand, classical fault tolerance follows a framework that is not completely fit to the universe of intentional and/or malicious faults. These issues will be discussed below.

The purpose of this paper is to make an attempt to systematize these new concepts and design principles. The paper describes the fundamental concepts behind intrusion tolerance (IT), tracing their connection with classical fault tolerance and security, and identifying the main delicate issues emerging in the evolution towards IT. We discuss the main strategies and mechanisms for architecting IT systems, and report on recent advances on distributed IT system architectures. For the sake of clarifying our position, we assume an 'architecture' to be materialized by a given composition of components. Components have given functional and non-functional properties, and an interface where these properties manifest themselves. Components are placed in a given topology of the archi-

* Extended version of paper appearing in Architecting Dependable Systems, R. Lemos, C. Gacek, A. Romanovsky (eds.), LNCS, Springer Verlag, 2003. Navigators Home Page: http://www.navigators.di.fc.ul.pt. Work partially supported by the EC, through proj. IST-1999-11583 (MAFTIA), and FCT, through the Large-Scale Informatic Sys. Lab. (LaSIGE) and proj. POSI/1999/CHS/33996 (DEFEATS).

[1] Example pointers to relevant IT research: MAFTIA: http://www.maftia.org. OASIS: http://www.tolerantsystems.org.

tecture, and interact through algorithms (in a generic sense), such that global system properties emerge from these interactions.

2 The case for Intrusion Tolerance

Dependability has been defined as that property of a computer system such that reliance can justifiably be placed on the service it delivers. The service delivered by a system is its behaviour as it is perceptible by its user(s); a user is another system (human or physical) which interacts with the former[5].

Dependability is a body of research that hosts a set of paradigms, amongst which fault tolerance, and it grew under the mental framework of accidental faults, with few exceptions [19, 17], but we will show that the essential concepts can be applied to malicious faults in a coherent manner.

2.1 A brief look at classical fault tolerance and security

Paraphrasing Gray[39], *"Why do computers fail and what can we do about it?"*, and notwithstanding the fact that all that works can fail, it seems that people tend to overestimate the quality of the computational systems they rely upon. This problem does not get better with distribution. Quoting Lamport, *"A distributed system is the one that prevents you from working because of the failure of a machine that you had never heard of"*. Machines should fail independently, for a start, and thus reliability (< 1) of a system being the product of the individual component reliabilities, the chances that some component is failed in a distributed system are greater than in a monolithic one. If components influence each other when failing, then the situation gets worse.

Malicious failures make the problem of reliability of a distributed system harder: failures can no longer be considered independent, as with accidental faults, since human attackers are likely to produce "common-mode" symptoms; components may perform collusion through distributed protocols; failures themselves become more severe, since the occurrence of inconsistent outputs, at wrong times, with forged identity or content, can no longer be considered of "low probability"; furthermore, they may occur at specially inconvenient instants or places of the system, driven by an intelligent adversary's mind.

This affects long-believed dogmas in classical fault tolerance: that fault types and their distribution follow statistically definable patterns; that the environment properties can be defined in probabilistically meaningful terms. In essence, that you can rely on: pre-defined and static fault models and environment assumptions.

The first question that comes to mind when addressing fault tolerance (FT) under a malicious perspective, is thus: *How do you model the mind of an attacker?*

Let us now look at the problem under a classical security perspective. Typical security properties are: confidentiality, or the measure in which a service or piece of information is protected from unauthorized disclosure; authenticity,

or the measure in which a service or piece of information is genuine, and thus protected from personification or forgery; integrity, or the measure in which a service or piece of information is protected from illegitimate and/or undetected modification; availability, or the measure in which a service or piece of information is protected from denial of authorized provision or access. The purpose of a sound system design is to secure all or some of these properties.

Traditionally, security has evolved as a combination of: preventing certain attacks from occurring; removing vulnerabilities from initially fragile software; preventing attacks from leading to intrusions. For example, in order to preserve confidentiality, it would be unthinkable to let an intruder read any confidential data at all. Likewise, integrity would assume not letting an intruder modify data at all. That is, with few exceptions, security has long been based on the prevention paradigm. However, let us tentatively imagine the tolerance paradigm in security[1]:

- assuming (and accepting) that systems remain to a certain extent vulnerable;
- assuming (and accepting) that attacks on components/sub-systems can happen and some will be successful;
- ensuring that the overall system nevertheless remains secure and operational.

Then, another question can be put: *How do we let data be read or modified by an intruder, and still ensure confidentiality or integrity?*

2.2 Dependability as a common framework

Let us observe the well-known fault-error-failure sequence in Figure 1. Dependability aims at preventing the failure of the system. This failure has a remote cause, which is a fault (e.g. a bug in a program, a configuration error) which, if activated (e.g. the program execution passes through the faulty line of code), leads to an error in system state. If nothing is done, failure will manifest itself in system behaviour.

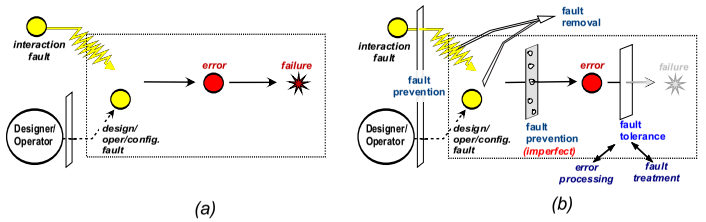


Fig. 1. Fault–> Error–> Failure sequence

In consequence, achieving dependability implies the use of combinations of: fault prevention, or how to prevent the occurrence or introduction of faults; fault removal, or how to reduce the presence (number, severity) of faults; fault forecasting, or how to estimate the presence, creation and consequences of faults; and last but not least, fault tolerance, or how to ensure continued correct service provision despite faults. Thus, achieving dependability vis-a-vis malicious faults (e.g. attacks and vulnerabilities) will mean the combined use of classical prevention and removal techniques with tolerance techniques.

Let us analyse the foundations of modular and distributed fault tolerance, and see how they fit in this new scenario:

- Topological separation, aiming at achieving failure independence and graceful degradation.
- Replication, as a form of redundancy, either software or hardware, with coarser or finer granularity.
- Configurability, through number of assumed faults, number of replicas, type of replica control (active, semi-active, passive, round robin, voting discipline).
- Resource optimisation, by software-to-hardware component mapping, and incremental F/T classes (omissive, assertive, arbitrary).

Topological separation makes the intruder's life more difficult, in terms of attack effort. For example, a secret can be split through several sites, and getting part of it reveals nothing about the whole. Replication makes components more resilient to damage, in terms of integrity and availability, but can also (though counter-intuitively) benefit confidentiality and authenticity, if we think of replicated execution of decisions (e.g. whether or not to render a piece of data, or to perform authentication. For example, one contaminated site alone (e.g., reference monitor) will not be able to decide authorization of access to a piece of data. Configurability is key to achieving incremental resilience to different strengths and kinds of intrusions (f+1 replication to withstand denial-of-service attacks, 3f+1 for resilience to Byzantine faults, and so forth), and to achieve different types of recovery (active being the most prompt, but also semi-active and passive). Resource optimisation reduces the cost of achieving intrusion tolerance.

This roadmap seems convincing, but in concrete terms, *how can tolerance be applied in the context of attacks, vulnerabilities, intrusions?*

2.3 Open problems

Let us analyse a few open problems that arise when intrusion tolerance is analysed from a security or fault tolerance background.

To start with, what contributes to the risk of intrusion? Risk is a combined measure of the probability of there being intrusions, and of their severity, that is, of the impact of a failure caused by them. The former is influenced by two factors that act in combination: the level of threat to which a computing or communication system is exposed; and the degree of vulnerability it possesses. The correct measure of how potentially insecure a system can be (in other words,

of how hard it will be to make it secure) depends: on the number and nature of the flaws of the system (vulnerabilities); on the potential for there existing attacks on the system (threats). Informally, the probability of an intrusion is given by the probability of there being an attack activating a vulnerability that is sensitive to it. The latter, the impact of failure, is measured by the cost of an intrusion in the system operation, which can be equated in several forms (economical, political, etc.).

Consider the following two example systems:

System Vault has a degree of vulnerability vvault=0.1, and since it has such high resilience, its designers have put it to serve anonymous requests in the Internet, with no control whatsoever to whoever tries to access it, that is, subject to a high level of threat, tvault=100.

System Sieve, on the other hand, is a vulnerable system, vsieve=10, and in consequence, its designers have safeguarded it, installing it behind a firewall, and controlling the accesses made to it, in what can be translated to a level of threat of tsieve=1.

Which of them offers a lower operational risk? Consider the product threat x vulnerability in our academic example: with the imaginary values we attributed to each system, it equates to the same value in both systems (10), although system Vault is a hundred times less vulnerable than system Sieve!

Should we try and bring the risk to zero? And is that feasible at all? This is classical prevention/removal: of the number, power, and severity of the vulnerabilities and the attacks the system may be subjected to. The problem is that neither can be made arbitrarily low, for several reasons: it is too costly and/or too complex (e.g., too many lines of code, hardware constraints); certain attacks come from the kind of service being deployed (e.g., public anonymous servers on the Internet); certain vulnerabilities are attached to the design of the system proper (e.g., mechanisms leading to races in certain operating systems).

And even if we could bring the risk to zero, would it be worthwhile? It should be possible to talk about *acceptable risk*: a measure of the probability of failure we are prepared to accept, given the value of the service or data we are trying to protect. This will educate our reasoning when we architect intrusion tolerance, for it establishes criteria for prevention/removal of faults and for the effort that should be put in tolerating the residual faults in the system. Further guidance can be taken for our system assumptions if we think that the hacker or intruder also incurs in a *cost of intruding*. This cost can be measured in terms of time, power, money, or combinations thereof, and clearly contributes to equating 'acceptable risk', by establishing the relation between 'cost of intruding' and 'value of assets'.

Secure Sockets Layer (SSL) reportedly ensures secure client-server interactions between browsers and WWW servers. Is SSL secure? Users have tended to accept that the interactions are secure (assumed low degree of vulnerability), without quantifying how secure. Several companies have built their commerce servers around SSL, some of them to perform financially significant transactions on the Internet (high level of threat). Netscape's SSL implementation was broken because of a bug that allowed to replicate session keys and thus decrypt any

communication. The corrected version was then broken at least twice through brute-force attacks on the 40-bit keys version (the only generically available at that time due to the U.S.A. export restrictions on cryptographic material). This initial situation led to a high risk.

Is SSL secure enough? Netscape issued a corrected version of SSL, and reported that it would cost at least USD10,000 to break an Internet session in computing time. The cost of intruding a system versus the value of the service being provided allows the architect to make a risk assessment. Someone who spends 10 000 EURO to break into a system and get 100 EURO worth of bounty, is doing a bad business. This defined the acceptable risk. Unfortunately, these estimates may fail: shortly after Netscape's announcement, a student using a single but powerful desktop graphics workstation, broke it for just USD600. However, what went wrong here was not the principle and the attitude of Netscape, just the risk assessment they made, which was too optimistic.

How tamper-proof is 'tamper-proof'? Classically, 'tamper-proof' means that a component is shielded, i.e. it cannot be penetrated. Nevertheless, in order to handle the difficulty of finding out that some components were "imperfectly" tamper-proof, experts in the area introduced an alternative designation, 'tamper-resistant', to express that fact. However, the imprecision of the latter is uncomfortable, leading to what we call the "watch-maker syndrome":

- *"Is this watch water-proof?"*
- *"No, it's water-resistant."*
- *"Anyway, I assume that I can swim with it!"*
- *"Well yes, you can! But... I wouldn't trust that very much..."*

A definition is required that attaches a quantifiable notion of "imperfect" to tamper-proofness, without necessarily introducing another vague term.

How can something be trusted and not trustworthy? Classically, in security one aims at building trust between components, but the merits of the object of our trust are not always analysed. This leads to what we called the "unjustified reliance syndrome":

- *"I trust Alice!"*
- *"Well Bob, you shouldn't, she's not trustworthy."*

What is the problem? Bob built trust on Alice through some means that may be correct at a high level (for example, Alice produced some signed credentials). However, Bob is being alerted to a fact he forgot (e.g., that Alice is capable of forging the credentials). It is necessary to establish the difference between what is required of a component, and what the component can give.

How do we model the mind of a hacker? Since the hacker is the perpetrator of attacks on systems, a fault model would be a description of what he/she can do. Then, a classical attempt at doing it would lead to the "well-behaved hacker syndrome":

- *"Hello, I'll be your hacker today, and here is the list of what I promise not to do."*
- *"Thank you, here are a few additional attacks we would also like you not to attempt."*

In consequence, a malicious-fault modelling methodology is required that refines the kinds of faults that may occur, and one that does not make naive assumptions about how the hacker can act. The crucial questions put in this Section will be addressed in the rest of the paper.

3 Intrusion Tolerance concepts

What is Intrusion Tolerance? As said earlier, the tolerance paradigm in security: assumes that systems remain to a certain extent vulnerable; assumes that attacks on components or sub-systems can happen and some will be successful; ensures that the overall system nevertheless remains secure and operational, with a quantifiable probability. In other words:

- faults— malicious and other— occur;
- they generate errors, i.e. component-level security compromises;
- error processing mechanisms make sure that security failure is prevented.

Obviously, a complete approach combines tolerance with prevention, removal, forecasting, after all, the classic dependability fields of action!

3.1 AVI composite fault model

The mechanisms of failure of a system or component, security-wise, have to do with a wealth of causes, which range from internal faults (e.g. vulnerabilities), to external, interaction faults (e.g., attacks), whose combination produces faults that can directly lead to component failure (e.g., intrusion). An intrusion has two underlying causes:

Vulnerability - fault in a computing or communication system that can be exploited with malicious intention

Attack - malicious intentional fault attempted at a computing or communication system, with the intent of exploiting a vulnerability in that system

Which then lead to:

Intrusion - a malicious operational fault resulting from a successful attack on a vulnerability

It is important to distinguish between the several kinds of faults susceptible of contributing to a security failure. Figure 2a represents the fundamental sequence of these three kinds of faults: attack → vulnerability → intrusion → failure. This well-defined relationship between attack/vulnerability/intrusion is what we call the *AVI composite fault model*. The AVI sequence can occur recursively in a coherent chain of events generated by the intruder(s), also called an intrusion campaign. For example, a given vulnerability may have been introduced in the course of an intrusion resulting from a previous successful attack.

Vulnerabilities are the primordial faults existing inside the components, essentially requirements, specification, design or configuration faults (e.g., coding

faults allowing program stack overflow, files with root setuid in UNIX, naive passwords, unprotected TCP/IP ports). These are normally accidental, but may be due to intentional actions, as pointed out in the last paragraph. *Attacks* are interaction faults that maliciously attempt to activate one or more of those vulnerabilities (e.g., port scans, email viruses, malicious Java applets or ActiveX controls).

The event of a successful attack activating a vulnerability is called an *intrusion*. This further step towards failure is normally characterized by an erroneous state in the system which may take several forms (e.g., an unauthorized privileged account with telnet access, a system file with undue access permissions to the hacker). Intrusion tolerance means that these errors can for example be unveiled by intrusion detection, and they can be recovered or masked. However, if nothing is done to process the errors resulting from the intrusion, failure of some or several security properties will probably occur.

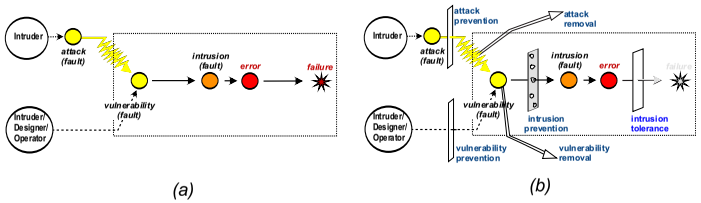


Fig. 2. (a) AVI composite fault model; (b) Preventing security failure

Why a composite model? The AVI model is a specialization of the generic fault → error → failure sequence, which has several virtues. Firstly, it describes the mechanism of intrusion precisely: without matching attacks, a given vulnerability is harmless; without target vulnerabilities, an attacks is irrelevant. Secondly, it provides constructive guidance to build in dependability against malicious faults, through the combined introduction of several techniques. To begin with, we can prevent some attacks from occurring, reducing the level of threat, as shown in Figure 2b. *Attack prevention* can be performed, for example, by shadowing the password file in UNIX, making it unavailable to unauthorized readers, or filtering access to parts of the system (e.g., if a component is behind a firewall and cannot be accessed from the Internet, attack from there is prevented). We can also perform *attack removal*, which consists of taking measures to discontinue ongoing attacks. However, it is impossible to prevent all attacks, so reducing the level of threat should be combined with reducing the degree of vulnerability, through *vulnerability prevention*, for example by using best-practices in the design and configuration of systems, or through *vulnerability removal* (i.e., debugging, patching, disabling modules, etc.) for example

it is not possible to prevent the attack(s) that activate(s) a given vulnerability. The whole of the above-mentioned techniques prefigures what we call *intrusion prevention*, i.e. the attempt to avoid the occurrence of intrusion faults.

Figure 2b suggests, as we discussed earlier, that it is impossible or infeasible to guarantee perfect prevention. The reasons are obvious: it may be not possible to handle all attacks, possibly because not all are known or new ones may appear; it may not be possible to remove or prevent the introduction of new vulnerabilities. For these intrusions still escaping the prevention process, forms of *intrusion tolerance* are required, as shown in the figure, in order to prevent system failure. As will be explained later, these can assume several forms: detection (e.g., of intruded account activity, of trojan horse activity); recovery (e.g., interception and neutralization of intruder activity); or masking (e.g., voting between several components, including a minority of intruded ones).

3.2 Trust and Trustworthiness

The adjectives "trusted" and "trustworthy" are central to many arguments about the dependability of a system. They have been often used inconsistently and up to now, exclusively in a security context[1]. However, the notions of "trust" and "trustworthiness" can be generalized to point to generic properties and not just security; and there is a well-defined relationship between them— in that sense, they relate strongly to the words "dependence" and "dependability".

Trust - the accepted dependence of a component, on a set of properties (functional and/or non-functional) of another component, subsystem or system

In consequence, a trusted component has a set of properties that are relied upon by another component (or components). If A trusts B, then A accepts that a violation in those properties of B might compromise the correct operation of A. Note that trust is not absolute: the degree of trust placed by A on B is expressed by the set of properties, functional and non-functional, which A trusts in B (for example, that a smart card: P1- Gives a correct signature for every input; P2- Has an MTTF of 10h (to a given level of threat...)).

Observe that those properties of B trusted by A might not correspond quantitatively or qualitatively to B's actual properties. However, in order for the relation implied by the definition of trust to be substantiated, trust should be placed *to the extent of* the component's trustworthiness. In other words, trust, the belief that B is dependable, should be placed in the measure of B's dependability.

Trustworthiness - the measure in which a component, subsystem or system, meets a set of properties (functional and/or non-functional)

The trustworthiness of a component is, not surprisingly, defined by how well it secures a set of functional and non-functional properties, deriving from its architecture, construction, and environment, and evaluated as appropriate. A

smart card used to implement the example above should actually meet or exceed P1 and P2, in the envisaged operation conditions.

The definitions above have obvious (and desirable) consequences for the design of intrusion tolerant systems: trust is not absolute, it may have several degrees, quantitatively or qualitatively speaking; it is related not only with security-related properties but with any property (e.g., timeliness); trust and trustworthiness lead to complementary aspects of the design and verification process. In other words, when A trusts B, A assumes something about B. The trustworthiness of B measures the *coverage* of that assumption.

In fact, one can reason separately about trust and trustworthiness. One can define chains or layers of trust, make formal statements about them, and validate this process. In complement to this, one should ensure that the components involved in the above-mentioned process are endowed with the necessary trustworthiness. This alternative process is concerned with the design and verification of components, or of verification/certification of existing ones (e.g., COTS). The two terms establish a separation of concerns on the failure modes: of the higher level algorithms or assertions (e.g., authentication/authorization logics); and of the infrastructure running them (e.g., processes/servers/communications).

The intrusion-tolerance strategies should rely upon these notions. The assertion 'trust on a trusted component' inspires the following guidelines for the construction of modular fault tolerance in complex systems: components are trusted to the extent of their trustworthiness; there is separation of concerns between what to do with the trust placed on a component (e.g., building fault-tolerant algorithms), and how to achieve or show its trustworthiness (e.g., constructing the component). The practical use of these guidelines is exemplified in later sections.

Let us revisit the example of tamper-proof device, under the light of these concepts. Tamperproofness is the property of a system/component of being shielded, i.e. whose attack model is that attacks can only be made at the regular interface. The component may not be perfect, but instead of using "tamper-resistant", we equate that in terms of the relative trustworthiness of the component, measured by the notion of *coverage* of the 'tamper-proofness' assumption.

As an example, assume the implementation of an authorisation service using JavaCards to store private keys. We assume JavaCards are tamper-proof, and so we argue that they are trustworthy not reveal these keys to an unauthorised party. Hence we build the authorisation service trusting the JavaCards, but only to the extent of their trustworthiness, that is, up to the coverage given by the tamper-proofness of the JavaCards. For example, an estimated figure could be put on the resilience of the JavaCards in a given environment (attack model), for example, the time or computational power to be broken.

3.3 Coverage and separation of concerns

Let us analyse how to build justified trust under the AVI model. Assume that component C has predicate P that holds with a coverage Pr, and this defines the component's trustworthiness, $\langle P, Pr \rangle$. Another component B should thus trust C to the extent of C possessing P with a probability Pr. So, there can be failures

consistent with the limited trustworthiness of C (i.e., that $Pr < 1$): these are "normal", and who/whatever depends on C, like B, should be aware of that fact, and expect it (and maybe take provisions to tolerate the fact in a wider system perspective).

However, it can happen that B trusts C to a greater extent than it should: trust was placed on C to an extent greater than its trustworthiness, perhaps due to a wrong or neglecting perception of the latter. This is a mistake of who/whatever uses that component, which can lead to unexpected failures.

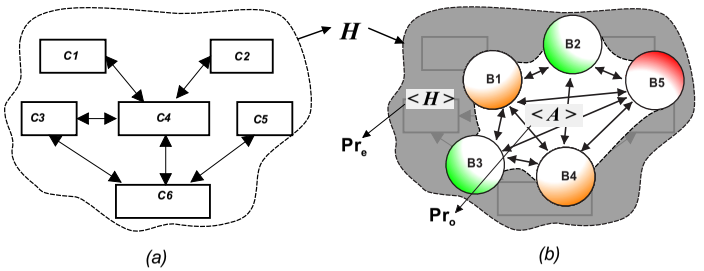


Fig. 3. Building trust

Finally, it can happen that the claim made about the trustworthiness of C is wrong (about predicate P, or its coverage Pr, or both). The component fails in worse, earlier, or more frequent modes than stated in the claim made about its resilience. In this case, even if B trusts C to the extent of $\langle P, Pr \rangle$ there can also be unexpected failures. However, this time, due to a mistake of whoever architected/built the component.

Note that this separation of concerns is important in a component-based approach at architecting dependable systems, since it allows an "OEM" or "COTS" approach: one "buys" a component with a certain specification, and the "manufacturer" (even if only another project design team) is ultimately responsible for the component meeting that specification.

Ultimately, what does it mean for component B to trust component C? It means that B assumes something about C. Generalizing, assume a set $\mathcal{B}$ of participants $(B_1 - B_n)$, which run an algorithm offering a set of properties A, on a run-time support environment composed itself of a set $\mathcal{C}$ of components $(C_1 - C_n)$. This modular vision is very adequate for, but not confined to, distributed systems. Imagine the environment as depicted in Figure 3a: $\mathcal{C}$ is architected so as to offer a set of properties, call it H. This serves as the support environment on which $\mathcal{B}$ operates, as suggested by the shaded cushion in Figure 3b.

Observe that $\mathcal{B}$ trusts $\mathcal{C}$ to provide H: $\mathcal{B}$ depends on the environment's properties H to implement the algorithm securing properties A. Likewise, a user of $\mathcal{B}$ trusts the latter to provide A. Without further discourse, this chain of trust would be: if $\mathcal{C}$ is trusted to provide H, then $\mathcal{B}$ is trusted to provide A.

Now let us observe the trustworthiness side. H holds with a probability Pr_e, the environmental assumption coverage[30]:

$Pr_e = Pr(H|f)$, f - any fault

Pr_e measures the trustworthiness of $\mathcal{C}$ (to secure properties H). Given H, A has a certain probability (can be 1 if the algorithm is deterministic and correct, can be less than one if it is probabilistic, and/or if it has design faults) of being fulfilled, the coverage Pr_o or operational assumption coverage:

$Pr_o = Pr(A|H)$

Pr_o measures the confidence on $\mathcal{B}$ securing properties A (given H as environment). Then, the trustworthiness of individual component $\mathcal{B}$ (to secure properties A given H) would be given by Pr_o.

As we propose, these equations should place limits on *the extent* of trust relations. $\mathcal{B}$ should trust $\mathcal{C}$ to the extent of providing H with confidence $Pr_e \leq 1$. However, since the user's trust on $\mathcal{B}$ is implied by $\mathcal{B}$'s trust on $\mathcal{C}$, then the user should trust $\mathcal{B}$ not in isolation, but conditioned to $\mathcal{C}$'s trustworthiness, that is, to the extent of providing A with confidence:

$Pr_a = Pr_o \times Pr_e = Pr(A|H) \times Pr(H|f) = Pr(A|f)$, f - any fault

The resulting chain could go on recursively. Pr_a is the probability that a user of the system composed of $\mathcal{B}$ and $\mathcal{C}$ enjoys properties A, in other words, it measures its trustworthiness.

4 IT frameworks and mechanisms

After introducing intrusion tolerance concepts, we begin this section by briefly analysing the main frameworks with which the architect can work in order to build intrusion tolerant systems: secure and fault-tolerant communication; software-based intrusion tolerance; hardware-based intrusion tolerance; auditing and intrusion detection. We will also look at several known security frameworks[33] under an IT perspective. Then we review error processing mechanisms in order to recover from intrusions.

4.1 Secure and fault-tolerant communication

This is the framework concerning the body of protocols ensuring intrusion tolerant communication. Essentially, relevant to this framework are secure channels and secure envelopes, and classic fault tolerant communication.

Secure channels are normally set up for regular communications between principals, or communications that last long enough for the concept of session or connection to make sense. For instance, file transfers or remote sessions. They live on a resilience/speed tradeoff, because they are on-line, and may use combinations of physical and virtual encryption. Secure channels adopt per-session

security, and normally use symmetric communication encryption, and signature or cryptographic checksum (MAC)-based channel authentication. Secure envelopes are used mainly for sporadic transmissions, such as email. They resort to per-message security and may make use of a combination of symmetric and asymmetric cryptography (also called hybrid) as a form of improving performance, especially for large message bodies.

Several techniques assist the design of fault-tolerant communication protocols. Their choice depends on the answer to the following question: *What are the classes of failures of communication network components?*

For the architect, this establishes the fundamental link between security and fault tolerance. In classical fault tolerant communication, it is frequent to see omissive fault models (crash, omissions, etc.). In IT the failure mode assumptions should be oriented by the AVI fault model, and by the way specific components' properties may restrict what should be the starting assumption: arbitrary failure (combination of omissive and assertive behaviour). In fact, this is the most adequate baseline model to represent malicious intelligence.

4.2 Software-based intrusion tolerance

Software-based fault tolerance has primarily been aimed at tolerating hardware faults using software techniques. Another important facet is software fault tolerance, aimed at tolerating software design faults by design diversity. Finally, it has long been known that software-based fault tolerance by replication may also be extremely effective at handling transient and intermittent software faults[33].

Software-based fault tolerance is the basis of modular FT, which underpins the main paradigms of distributed fault tolerance. The main players are software modules, whose number and location in several sites of the system depends on the dependability goals to be achieved.

Let us analyse what can be done under an IT perspective. In the case of design or configuration faults, simple replication would apparently provide little help: errors would systematically occur in all replicas. This is true from a vulnerability viewpoint: it is bound to exist in all replicas. However, the common-mode syndrome under the AVI model concerns intrusions, or attack-vulnerability pairs, rather than vulnerabilities alone.

This gives the architect some chances. Consider the problem of common-mode vulnerabilities, and of common-mode attacks, i.e. attacks that can be cloned and directed automatically and simultaneously to all (identical) replicas. Design diversity can be applied, for example, by using different operating systems, both to reduce the probability of common-mode vulnerabilities (the classic way), and to reduce the probability of common-mode attacks (by obliging the attacker to master attacks to more than one architecture)[9]. Both reduce the probability of common-mode intrusion, as desired.

This can be taken farther: different system architectures are used, or execution results are tested against assertions about the desired outcome. For example, each replica of a given component can be designed and developed by a different

team of programmers. Software design diversity is rather expensive (most software products already cost too much, even when a single development team is involved) and as such it is only employed when the stakes are very high.

However, even mere replication with homogeneous components can yield significant results. How? When components have a high enough trustworthiness that claims can be made about the hardness of achieving a successful attack-vulnerability match on one of them (e.g. "breaking" it). In this case, we could apply the classical principle of achieving a much higher reliability of a replica set than the individual replicas' reliability. For example, simple replication can be used to tolerate attacks, by making it difficult and lengthy for the attacker to launch simultaneous attacks to all replicas with success.

The caveat is to derive the precise formula. The canonic formula assumes independent failure modes. In IT, in our opinion, the truth lies in the middle: they are neither independent, nor common mode. Exactly where, is still a research topic. In following this approach, one must not forget the considerations of the previous section on fault models complying with malicious behaviour. That is, replication management (e.g., voting) criteria should normally encompass assertive (value domain) behaviour, and of a subtle nature (e.g. inconsistent or Byzantine).

Finally, since replicas of a same component can reside in different hardware and/or operating system architectures, and execute at different moments and in different contexts, this implicit "diversity" is enough to tolerate many faults, especially those faults that lead to an intermittent behaviour. It follows that transient and intermittent faults can also be tolerated this way, even certain malicious faults such as low intensity sporadic attacks.

The introduction of a vulnerability as a first step of an intrusion campaign (to be further exploited by subsequent attacks) can also be discussed under the attacker power perspective. However, it can methodically be introduced in a stealth way, unlike the attacks which require synchronization, and this renders such attacks potentially difficult to evaluate, and thus more dangerous than the originating vulnerability analysed individually.

4.3 Hardware-based intrusion tolerance

Software-based and hardware-based fault tolerance are not incompatible design frameworks[33]. In a modular and distributed systems context, hardware fault tolerance today should rather be seen as a means to construct *fail-controlled* components, in other words, components that are prevented from producing certain classes of failures. This contributes to establish improved levels of trustworthiness, and to use the corresponding improved trust to achieve more efficient fault-tolerant systems.

Distributed algorithms that tolerate arbitrary faults are expensive in both resources and time. For efficiency reasons, the use of hardware components with enforced controlled failure modes is often advisable, as a means for providing an infrastructure where protocols resilient to more benign failures can be used,

without that implying a degradation in the resilience of the system to malicious faults.

4.4 Auditing and intrusion detection

Logging system actions and events is a good management procedure, and is routinely done in several operating systems. It allows a posteriori diagnosis of problems and their causes, by analysis of the logs. Audit trails are a crucial framework in security.

Not only for technical, but also for accountability reasons, it is very important to be able to trace back the events and actions associated with a given time interval, subject, object, service, or resource. Furthermore, it is crucial that all activity can be audited, instead of just a few resources. Finally, the granularity with which auditing is done should be related with the granularity of possible attacks on the system. Since logs may be tampered with by intruders in order to delete their own traces, logs should be tamperproof, which they are not in many operating systems.

Intrusion Detection (ID) is a classical framework in security, which has encompassed all kinds of attempts to detect the presence or the likelihood of an intrusion. ID can be performed in real-time, or off-line. In consequence, an intrusion detection system (IDS) is a supervision system that follows and logs system activity, in order to detect and react (preferably in real-time) against any or all of: attacks (e.g. port scan detection), vulnerabilities (e.g. scanning), and intrusions (e.g. correlation engines).

Definition of ID given by NSA (1998): Pertaining to techniques which attempt to detect intrusion into a computer or network by observation of actions, security logs, or audit data. Detection of break-ins or attempts either manually or via software expert systems that operate on logs or other information available on the network.

An aspect deserving mention under an IT viewpoint is the dichotomy between error detection and fault diagnosis, normally concealed in current ID systems[1]. Why does it happen, and why is it important? It happens because IDS are primarily aimed at complementing prevention and triggering manual recovery. It is important because if automatic recovery (fault tolerance) of systems is desired, there is the need to clearly separate: what are errors as per the security policy specification; what are faults, as per the system fault model. Faults (e.g., attacks, vulnerabilities, intrusions) are to be diagnosed, in order that they can be treated (e.g. passivated, removed). Errors are to be detected, in order that they can be automatically processed in real-time (recovered, masked).

To understand the problem better, consider the following example situations, in an organization that has an intranet with an extranet connected to the public Internet, and is fit with an IDS: (a) the IDS detects a port scan against an extranet host, coming from the Internet; (b) the IDS detects a port scan against an internal host, coming from the Internet; (c) the IDS detects a port scan against an internal host, coming from the intranet. What is the difference between (a), (b) and (c)? In fact, (a) must (currently) be considered "normal"

Internet activity, and would at most be an attack, not an intrusion, if the fault model includes, for example, given thresholds or patterns for external activity. On the other hand, (b) implies an error (in the outside protection mechanisms) if the security policy (as expected...) rules out the permission for making internal port scans from the Internet. Finally, (c) also prefigures an error, since it would also be expected that the security policy forbids port scans from inside the institution.

ID as error detection will be detailed later in the paper. It addresses detection of erroneous states in a system computation, deriving from malicious action e.g., modified files or messages, OS penetration by buffer overflow. ID as fault diagnosis seeks other purposes, and as such, both activities should not be mixed. Regardless of the error processing mechanism (recovery or masking), administration subsystems have a paramount action w.r.t. fault diagnosis. This facet of classical ID fits into fault treatment[1]. It can serve to give early warning that errors may occur (vulnerability diagnosis, attack forecasting), to assess the degree of success of the intruder in terms of corruption of components and subsystems (intrusion diagnosis), or to find out who/what performed an attack or introduced a vulnerability (attack diagnosis).

Diagnosis can be done proactively, before errors happen. For example, by activating faults (e.g. vulnerability scanning) and post-processing (forecasting their effects) one can get a metrics of resilience (subject to the method coverage...). On the other hand, by analysing external activity one can try and predict attacks (e.g. external port scan analysis).

4.5 Some security frameworks under an IT look

Observe that some mechanisms pertaining to known frameworks in security (secure channels and envelopes, authentication, protection, cryptographic communication)[33] can be revisited under the IT perspective and thus constitute useful conceptual tools for the architect of IT systems.

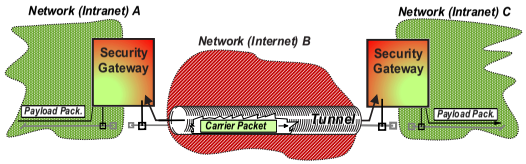


Fig. 4. Tunnels, Secure Channels and Envelopes

Secure tunnels, e.g. those built on the Internet with secure IP-over-IP channels, are intrusion prevention devices (Figure 4): they enforce confidentiality, integrity (and sometimes authenticity) between access points, despite intrusion

attempts. Coverage is given by: the resilience of the tunnelling method, and of the access point gateways.

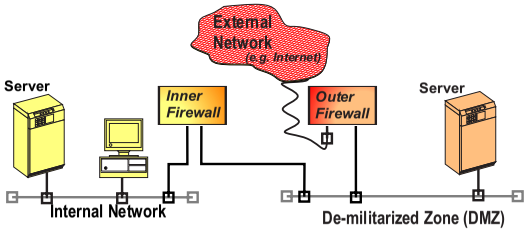


Fig. 5. Firewalls

Firewalls are intrusion prevention devices (Figure 5): they prevent attacks on inside machines that might exploit vulnerabilities leading to an intrusion. Their coverage is given by: the power of the semantics of firewall functions, and the resilience of bastions.

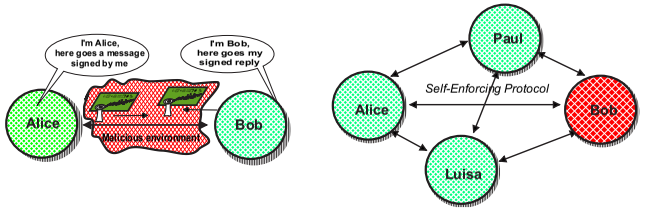


Fig. 6. (a) Authentication; (b) Communication and agreement

Mechanisms and protocols providing authentication between two or more entities (signature, message authentication codes (MAC)) are also intrusion prevention devices (Figure 6a): they enforce authenticity preventing forging of identity of participants or authorship/origin of data. Coverage is given by: the resilience of the signature/authentication method.

Last but not least, some cryptographic protocols are very important intrusion tolerance building blocks that can be used recursively. Seen as building blocks, self-enforcing protocols such as Byzantine agreement or atomic multicast (Figure 6b), are intrusion tolerance devices: they perform error processing or masking

($3f+1$, $2f+1$, $f+2$, depending on the fault model) and ensure message delivery despite actual intrusions. Coverage is given by: semantics of protocol functions, underlying model assumptions. Trusted Third Party (TTP) protocols are also intrusion tolerance devices which perform error processing/masking, but depend on a TTP for their correct operation (Figure 7a). Coverage depends on: semantics of protocol functions, underlying model assumptions, resilience of TTP. Finally, threshold cryptographic protocols are intrusion tolerance devices (Figure 7b): they perform error processing/masking under a threshold assumption of no more than $f+1$ out of n intrusions. Their coverage is given by: semantics of the cryptographic functions, brute force resilience of the cipher, underlying model assumptions.

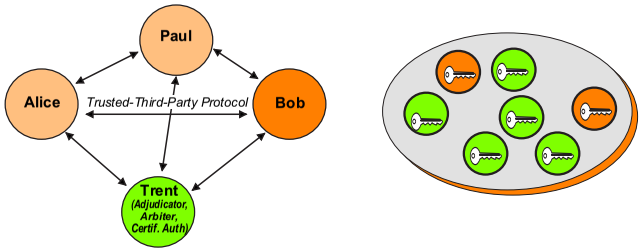


Fig. 7. (a) Trusted Third Party; (b) Threshold cryptography

4.6 Processing the errors deriving from intrusions

Next we review classes of mechanisms for processing errors deriving from intrusions. Essentially, we discuss the typical error processing mechanisms used in fault tolerance, under an IT perspective: error detection; error recovery; and error masking.

Error detection is concerned with detecting the error after an intrusion is activated. It aims at: confining it to avoid propagation; triggering error recovery mechanisms; triggering fault treatment mechanisms. Examples of typical errors are: forged or inconsistent (Byzantine) messages; modified files or memory variables; phoney OS accounts; sniffers, worms, viruses, in operation.

Error recovery is concerned with recovering from the error once it is detected. It aims at: providing correct service despite the error; recovering from effects of intrusions. Examples of backward recovery are: the system goes back to a previous state known as correct and resumes; the system having suffered DoS (denial of service) attack, re-executes the affected operation; the system having detected corrupted files, pauses, reinstalls them, goes back to last correct point. Forward

recovery can also be used: the system proceeds forward to a state that ensures correct provision of service; the system detects intrusion, considers corrupted operations lost and increases level of security (threshold/quorums increase, key renewal); the system detects intrusion, moves to degraded but safer operational mode.

Error masking is a preferred mechanism when, as often happens, error detection is not reliable or can have large latency. Redundancy is used systematically in order to provide correct service without a noticeable glitch. As examples: systematic voting of operations; Byzantine agreement and interactive consistency; fragmentation-redundancy-scattering; sensor correlation (agreement on imprecise values).

4.7 Intrusion detection mechanisms

As to the methodology employed, classic ID systems belong to one (or a hybrid) of two classes: behaviour-based (or anomaly) detection systems; and knowledge-based (or misuse) detection systems.

Behaviour-based (anomaly) detection systems are characterized by needing no knowledge about specific attacks. They are provided with knowledge about the normal behaviour of the monitored system, acquired e.g., through extensive training of the system in correct operation. As advantages: they do not require a database of attack signatures that needs to be kept up-to-date. As drawbacks: there is a significant potential for false alarms, namely if usage is not very predictable with time; they provide no information (diagnosis) on type of intrusion, they just signal that something unusual happened.

Knowledge-based (misuse) systems rely on a database of previously known attack signatures. Whenever an activity matches a signature, an alarm is generated. As advantages: alarms contain diagnostic information about the cause. The main drawback comes from the potential for omitted or missed alarms, e.g. unknown attacks (incomplete database) or new attacks (on old or new vulnerabilities).

Put under an IT perspective, error detection mechanisms of either class can and should be combined. Combination of ID with automated recovery mechanisms is a research subject in fast progress[1, 14, 23, 11].

This can be systematized and generalized as in Figure 8. System activity patterns are followed, and compared against reference patterns[1]: normal and abnormal. Whenever there is a match with any of the abnormal patterns, an error is reported (this is akin to the misuse style of detection). Likewise, whenever system activity falls outside the normal patterns, an error is also reported (this falls into the anomaly category). Note that both methodologies are seamlessly combined.

Modern intrusion detection should address errors deriving or not from malicious action. In fact, a detector of errors caused by malicious faults should detect errors caused by non-malicious ones. This puts emphasis on the result— the observable failure of some component to provide correct service— rather than on the cause. The possible causes must have been defined previously, when devising

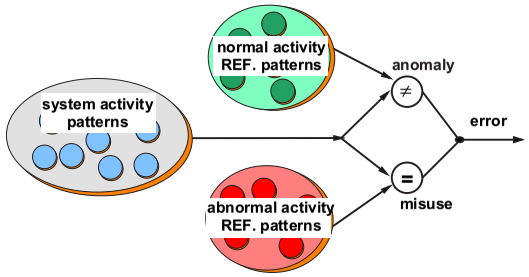


Fig. 8. Intrusion detection methodologies

the fault model (AVI - attack, vulnerability, intrusion). Example: a Byzantine failure detector in a distributed system, detects an abnormal behaviour of components, such as sending inconsistent data to different participants. Whether or not it is caused by malicious entities, is irrelevant. The quality of such detectors should be measured by parameters such as: false alarm rate; omitted alarm rate; detection latency.

5 Intrusion Tolerance strategies

Not surprisingly, intrusion tolerance strategies derive from a confluence of classical fault tolerance and security strategies[33]. Strategies are conditioned by several factors, such as: type of operation, classes of failures (i.e., power of intruder); cost of failure (i.e., limits to the accepted risk); performance; cost; available technology. Technically, besides a few fundamental tradeoffs that should always be made in any design, the grand strategic options for the design of an intrusion-tolerant system develop along a few main lines that we discuss in this section. We describe what we consider to be the main strategic lines that should be considered by the architect of IT systems, in a list that is not exhaustive. Once a strategy is defined, design should progress along the guidelines suggested by the several intrusion-tolerance frameworks just presented.

5.1 Fault Avoidance vs. Fault Tolerance

The first issue we consider is oriented to the system construction, whereas the remaining are related with its operational purpose. It concerns the balance between faults avoided (prevented or removed) and faults tolerated.

On the one hand, this is concerned with the 'zero-vulnerabilities' goal taken in many classical security designs. The Trusted Computing Base paradigm[36], when postulating the existence of a computing nucleus that is impervious to

hackers, relies on that assumption. Over the years, it became evident that this was a strategy impossible to follow in generic system design: systems are too complex for the whole design and configuration to be mastered. On the other hand, this balance also concerns attack prevention. Reducing the level of threat improves on the system resilience, by reducing the risk of intrusion. However, for obvious reasons, this is also a very limited solution. As an example, the firewall paranoia of preventing attacks on intranets also leaves many necessary doors (for outside connectivity) closed in its way.

Nevertheless, one should avoid falling in the opposite extreme of the spectrum —assume the worst about system components and attack severity— unless the criticality of the operation justifies a 'minimal assumptions' attitude. This is because arbitrary failure protocols are normally costly in terms of performance and complexity.

The strategic option of using some trusted components— for example in critical parts of the system and its operation— may yield more performant protocols. If taken under a tolerance (rather than prevention) perspective, very high levels of dependability may be achieved. But the condition is that these components be made trustworthy (up to the trust placed on them, as we discussed earlier), that is, that their faulty behaviour is indeed limited to a subset of the possible faults. This is achieved by employing techniques in their construction that lead to the prevention and/or removal of the precluded faults, be them vulnerabilities, attacks, intrusions, or other faults (e.g. omission, timing, etc.).

The recursive (by level of abstraction) and modular (component-based) use of fault tolerance and fault prevention/removal when architecting a system is thus one of the fundamental strategic tradeoffs in solid but effective IT system design. This approach was taken in previous architectural works[29], but has an overwhelming importance in IT, given the nature of faults involved.

5.2 Confidential Operation

When the strategic goal is confidentiality, the system should preferably be architected around error masking, resorting to schemes that despite allowing partial unauthorised reads of pieces of data, do not reveal any useful information. Or schemes that by requiring a quorum above a given threshold to allow access to information, withstand levels of intrusion to the access control mechanism that remain below that threshold. Schemes relying on error detection/recovery are also possible. However, given the specificity of confidentiality (once read, read forever...), they will normally imply some form of forward, rather than backward recovery, such as rendering the unduly read data irrelevant in the future. They also require low detection latency, to mitigate the risk of error propagation and eventual system failure (in practical terms, the event of information disclosure).

5.3 Perfect Non-stop Operation

When no glitch is acceptable, the system must be architected around error masking, as in classical fault tolerance. Given a set of failure assumptions, enough

space redundancy must be supplied to achieve the objective. On the other hand, adequate protocols implementing systematic error masking under the desired fault model must be used (e.g. Byzantine-resilient, TTP-based, etc.). However, note that non-stop availability against general denial-of-service attacks is still an ill-mastered goal in open systems.

5.4 Reconfigurable Operation

Non-stop operation is expensive and as such many services resort to cheaper redundancy management schemes, based on error recovery instead of error masking. These alternative approaches can be characterized by the existence of a visible glitch. The underlying strategy, which we call reconfigurable operation, is normally addressed at availability- or integrity-oriented services, such as transactional databases, web servers, etc.

The strategy is based on intrusion detection. The error symptom triggers a reconfiguration procedure that automatically replaces a failed component by a correct component, or an inadequate or incorrect configuration by an adequate or correct configuration, under the new circumstances (e.g. higher level of threat). For example, if a database replica is attacked and corrupted, it is replaced by a backup. During reconfiguration the service may be temporarily unavailable or suffer some performance degradation, whose duration depends on the recovery mechanisms. If the AVI sequence can be repeated (e.g., while the attack lasts), the service may resort to configurations that degrade QoS in trade for resilience, depending on the policy used (e.g., temporarily disabling a service that contains a vulnerability that cannot be removed, or switching to more resilient but slower protocols).

5.5 Recoverable Operation

Disruption avoidance is not always mandatory, and this may lead to cheaper and simpler systems. Furthermore, in most denial-of-service scenarios in open systems (Internet), it is generically not achievable.

Consider that a component crashes under an attack. An intrusion-tolerant design can still be obtained, if a set of preconditions hold for the component: (a) it takes a lower-bounded time T_c to fall; (b) it takes a upper-bounded time T_r to recover; (c) the duration of blackouts is short enough for the application's needs.

Unlike what happens with classic FT recoverable operation[33], where (c) only depends on (b), here the availability of the system is defined in a more elaborate way, proportionate to the level of threat, in terms of attack severity and duration. Firstly, for a given attack severity, (a) determines system reliability under attack. If an attack lasts less than T_c, the system does not even crash. Secondly, (a) and (b) determine the time for service restoration. For a given attack duration T_a, the system may either recover completely after T_r ($T_a < T_c+T_r$), or else cycle up-down, with a duty cycle of $T_c/(T_c+T_r)$ (longer attacks).

Moreover, the crash, which is provoked maliciously, must not give rise to incorrect computations. This may be achieved through several techniques, amongst which we name secure check-pointing and logging. Recoverable exactly-once operation can be achieved with intrusion-tolerant atomic transactions[33]. In distributed settings, these mechanisms may require secure agreement protocols.

This strategy concerns applications where at the cost of a noticeable temporary service outage, the least amount of redundancy is used. The strategy also serves long-running applications, such as data mining or scientific computations, where availability is not as demanding as in interactive applications, but integrity is of primary concern.

5.6 Fail-Safe

In certain situations, it is necessary to provide for an emergency action to be performed in case the system can no longer tolerate the faults occurring, i.e. it cannot withstand the current level of threat. This is done to prevent the system from evolving to a potentially incorrect situation, suffering or doing unexpected damage. In this case, it is preferable to shut the system down at once, what is called *fail-safe* behaviour. This strategy, often used in safety- and mission-critical systems, is also important in intrusion tolerance, for obvious reasons. It may complement other strategies described above.

6 Modelling malicious faults

A crucial aspect of any fault-tolerant architecture is the fault model upon which the system architecture is conceived, and component interactions are defined. The fault model conditions the correctness analysis, both in the value and time domains, and dictates crucial aspects of system configuration, such as the placement and choice of components, level of redundancy, types of algorithms, and so forth. A system fault model is built on assumptions about the way system components fail.

What are malicious faults? In the answer to this question lies the crux of the argument with regard to "adequate" intrusion fault models. The term 'malicious' is itself very suggestive, and means a special intent to cause damage. But how do we model the mind and power of the attacker? Indeed, many works have focused on the 'intent', whereas from an IT perspective, one should focus on the 'result'. That is, what should be extracted from the notion of 'maliciousness' is a technical definition of its objective: the *violation of several or all of the properties of a given service*, attempted in any possible manner within the power available to the intruder.

Classically, failure assumptions fall into essentially two kinds: controlled failure assumptions, and arbitrary failure assumptions.

Controlled failure assumptions specify qualitative and quantitative bounds on component failures. For example, the failure assumptions may specify that components only have timing failures, and that no more than f components

fail during an interval of reference. Alternatively, they can admit value failures, but not allow components to spontaneously generate or forge messages, nor impersonate, collude with, or send conflicting information to other components. In the presence of accidental faults this approach is realistic, since it represents very well how common systems work, failing in a benign manner most of the time. However, it can hardly be directly extrapolated to malicious faults, under the above definition of maliciousness.

Arbitrary failure assumptions ideally specify no qualitative or quantitative bounds on component failures. In this context, an arbitrary failure means the capability of generating an interaction at any time, with whatever syntax and semantics (form and meaning), anywhere in the system. Arbitrary failure assumptions adapt perfectly to the notion of maliciousness, but they are costly to handle, in terms of performance and complexity, and thus are not compatible with the user requirements of the vast majority of today's on-line applications.

Obviously, this should be understood in the context of a universe of "possible" failures of the concerned operation mode of the component. For example, the possible failure modes of interactions between components of a distributed system might be limited to combinations of timeliness, form, meaning, and target of those interactions (let us call them messages), and might not encompass the arbitrary cloning of system components. On the other hand, practical systems based on arbitrary failure assumptions must specify quantitative bounds on the number of failed components, or at least equate tradeoffs between resilience of their solutions and the number of failures eventually produced.

Note that the problem lies in how representative are our assumptions vis-a-vis what happens in reality. That is, a problem of *coverage* of our assumptions. So, how to proceed?

6.1 Arbitrary failure assumptions

Consider operations of very high value and/or criticality, such as: financial transactions; contract signing; provision of long term credentials; state secrets. The risk of failure due to violation of assumptions should not be incurred. This justifies considering arbitrary failure assumptions, and building the system around arbitrary-failure resilient building blocks (e.g. Byzantine agreement protocols), despite a possible performance penalty.

In consequence, no assumptions are made on the existence of trusted components such as security kernels or other fail-controlled components. Likewise, a time-free or asynchronous approach must be followed, i.e. no assumptions about timeliness, since timing assumptions are susceptible to be attacked. This limits the classes of applications that can be addressed under these assumptions: asynchronous models cannot solve timed problems.

In practice, many of the emerging applications we see today, particularly on the Internet, have interactivity or mission-criticality requirements. Timeliness is part of the required attributes, either because of user-dictated quality-of-service requirements (e.g., network transaction servers, multimedia rendering, synchronised groupware, stock exchange transaction servers), or because of safety

constraints (e.g., air traffic control). So we should seek alternative fault model frameworks to address these requirements under malicious faults.

6.2 Hybrid failure assumptions considered useful

Hybrid assumptions combining several kinds of failure modes would be desirable. There is a body of research, starting with [25] on hybrid failure models that assume different failure type distributions for different nodes. For instance, some nodes are assumed to behave arbitrarily while others are assumed to fail only by crashing. The probabilistic foundation of such distributions might be hard to sustain in the presence of malicious intelligence, unless their behaviour is constrained in some manner. Consider a component or sub-system for which given controlled failure assumptions were made. How can we enforce trustworthiness of the component vis-a-vis the assumed behaviour, that is, coverage of such assumptions, given the unpredictability of attacks and the elusiveness of vulnerabilities?

A composite (AVI) fault model with hybrid failure assumptions is one where the presence and severity of vulnerabilities, attacks and intrusions varies from component to component. Some parts of the system would justifiably exhibit fail-controlled behaviour, whilst the remainder of the system would still be allowed an arbitrary behaviour. This might best be described as *architectural hybridisation*, in the line of works such as [28, 34, 13], where failure assumptions are in fact enforced by the architecture and the construction of the system components, and thus substantiated. That is (*see* Section 3) the component is made *trustworthy* enough to match the *trust* implied by the fail-controlled assumptions.

The task of the architect is made easier since the controlled failure modes of some components vis-a-vis malicious faults restrict the system faults the component can produce. In fact a form of fault prevention was performed at system level: some kinds of system faults are simply not produced. Intrusion-tolerance mechanisms can now be designed using a mixture of arbitrary-failure (fail-uncontrolled or non trusted) and fail-controlled (or trusted) components.

Hybrid failure assumptions can also be the key to secure timed operation. With regard to timeliness and timing failures, hybridisation yields forms of partial synchrony: (i) some subsystems exhibit controlled failure modes and can thus supply timed services in a secure way; (ii) the latter assist the system in fulfilling timeliness specifications; (iii) controlled failure of those specifications is admitted, but timing failure detection can be achieved with the help of trusted components[13].

7 Architecting intrusion-tolerant systems

In this section, we discuss a few notions on architecting intrusion-tolerant systems.

7.1 (Almost) no assumptions

The fail-uncontrolled or arbitrary failure approach to IT architecture is based on assuming as little as possible about the environment's behaviour (faults, synchronism), with the intent of maximizing coverage. It provides a conceptually simple framework for developing and reasoning about the correctness of an algorithm, satisfying safety under any conditions, and providing liveness under certain conditions, normally defined in a probabilistic way.

Randomised Byzantine agreement protocols are an example of typical protocols in this approach. They may not terminate with non-zero probability, but this probability can be made negligible. In fact, a protocol using cryptography always has a residual probability of failure, determined by the key lengths. Of course, for the system as a whole to provide useful service, it is necessary that at least some of the components are correct. This approach is essentially parametric: it will remain correct if a sufficient number of correct participants exist, for any hypothesised number of faulty participants f. Or in other words, with almost no assumptions one is able to achieve extremely resilient protocols.

This has some advantages for the design of secure distributed systems, which is one reason for pursuing such an approach. In fact, sometimes it is necessary and worthwhile to sacrifice performance or timeliness for resilience, for example for very critical operations (key distribution, contract signing, etc.)

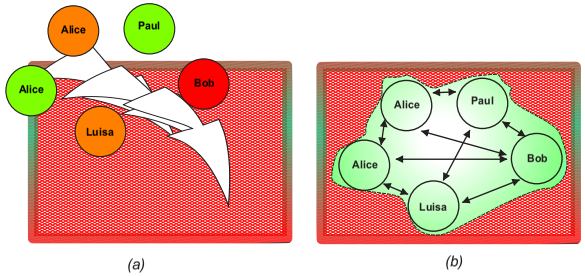


Fig. 9. Arbitrary failure approach

Figure 9 shows the principle in simple terms. The metaphore used from now on is: greyer for hostile, malicious, and whiter for benign, correct. Figure 9a shows the participants being immersed in a hostile and asynchronous environment. The individual hosts and the communication environment are not trusted. Participants may be malicious, and normally the only restriction assumed is in the number of ill-behaved participants. Figure 9b suggests that the protocol, coping with the environment's deficiencies, ensures that the participants collectively provide a correct service (whiter shade).

For a protocol to be able to provide correct service, it must cope with arbitrary failures of components and the environment. For example, component Ck is malicious, but this may be because the component itself or host C have been tampered with, or because an intruder in the communication system simulates that behaviour.

7.2 Non-justified assumptions, or the power of faith

Alternatively, IT architecture may take the fail-controlled approach. Sometimes, it may simply be assumed that the environment is benign, without substantiating those assumptions. This is often done in accidental fault tolerance, when the environment is reasonably well-known, for example, from statistic measurements. Is it a reasonable approach for malicious faults?

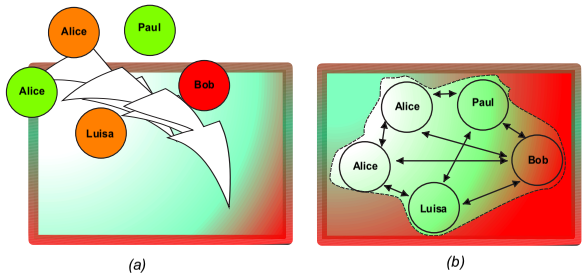


Fig. 10. Non-justified assumptions

Figure 10a shows the participants being immersed in an assumed moderately benign environment (essentially white, with a thin dark part, according to our metaphors). For example, it is usual to consider that the individual hosts (local environment) are trusted, and that the communication environment, though not trusted has a given limited attack model. Some user participants may be malicious.

The implementation is bound to work most of the times. However, it should not be surprising that a behaviour that is assumed out of statistic evidence (or worse, out of faith...) and not by enforcement, can be defrauded by an intruder attacking the run-time environment. Thus, it may turn out that the latter behaves in a manner worse than assumed (e.g., hosts were not that trustworthy, or the communication support was more severely attacked than the model assumed), as suggested in Figure 10b where, say upon an attack, the environment is shown actually more aggressive than initially thought in Figure 10a.

In consequence, making assumptions that are not substantiated in a strong manner may in many cases lead to the lack of trustworthiness (coverage) on the properties of a component or subsystem (suggested in our example by the dark shade partially hitting the participants and protocol). This may be problematic, because it concerns failures not assumed, that is, for which the protocol is not prepared, and which may be orchestrated by malicious intelligence. Their consequences may thus be unpredictable. We discuss a correct approach below.

7.3 Architectural hybridisation

Architectural hybridisation is a solid guiding principle for architecting fail-controlled IT systems. One wishes to avoid the extreme of arbitrary assumptions, without incurring the risks of lack of coverage. Assuming something means trusting, as we saw earlier on, and so architectural hybridisation is an enabler of the approach of *using trusted components*, by making them *trustworthy* enough.

Essentially, the architect tries to make available black boxes with benign behaviour, of omissive or weak fail-silent class[33]. These can have different capabilities (e.g. synchronous or not; local or distributed), and can exist at different levels of abstraction. A good approach is to dress them as run-time environment components, which can be accessed by system calls but provide trustworthy results, in contrast with calls to an untrusted environment. Of course, fail-controlled designs can yield fault-tolerant protocols that are more efficient than truly arbitrary assumptions protocols, but more robust than non-enforced controlled failure protocols.

The tolerance attitude in the design of hybrid IT systems can be characterized by a few aspects:

- assuming as little as possible from the environment or other components;
- making assumptions about well-behaved (trusted) components or parts of the environment whenever strictly necessary;
- enforcing the assumptions on trusted components, by construction;
- unlike classical prevention-based approaches, trusted components do not intervene in all operations, they assist only crucial steps of the execution;
- protocols run thus in an non-trusted environment, single components can be corrupted, faults (intrusions) can occur;
- correct service is built on distributed fault tolerance mechanisms, e.g., agreement and replication amongst participants in several hosts.

7.4 Prevention, Tolerance, and a bit of salt

On achieving trustworthy components, the architect should bear in mind a recipe discussed earlier: the good balance between prevention and tolerance. Let us analyze the principles of operation of a trusted third party (TTP) protocol, as depicted in Figure 11a. Participants Alice, Paul and Bob, run an IT protocol amongst themselves, and trust Trent, the TTP component, to provide a few

services that assist the protocol in being intrusion tolerant. What the figure does not show and is seldom asked is: is the TTP trustworthy?

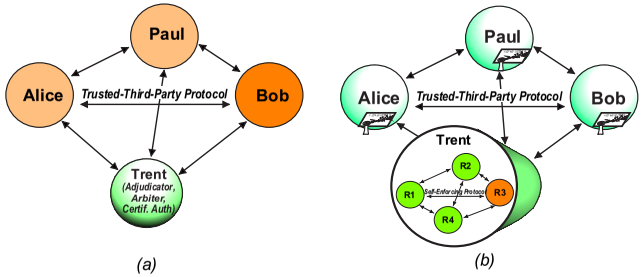


Fig. 11. (a) TTP protocol; (b) Enforcing TTP trustworthiness

In fact, the TTP is the perfect example of a trusted component that is sometimes (often?) trusted to an extent greater than its trustworthiness.

In Figure 11b we "open the lid" of the TTP and exemplify how a good combination of prevention and tolerance can render it trustworthy. To start with, we require certificate-based authentication, as a means to prevent certain failures from ocurring in the point-to-point interaction of participants with the TTP (e.g., impersonation, forging, etc.). Then, if we replicate the TTP, we make it resilient to crashes, and to a certain level of attacks on the TTP server replicas, if there is enough redundancy. Furthermore, the replicas should communicate through self-enforcing protocols of the Byzantine-resilient kind, if malicious faults can be attempted at subsets of server replicas.

The user need not be aware of the additional complexity and distribution of the TTP, a usual principle in fault tolerance. In fact, we should "close the lid" so that participants see essentially a single logical entity which they trust (as in Figure 11a). However, by having worked at component level (TTP), we achieve trustworthy behaviour of the component as seen at a higher level (system). Note that in fact, we have prevented some system faults from occurring. This duality prevention/tolerance can be applied recursively in more than one instance. Recently, there has been extensive research on making trustworthy TTPs, for example by recursively using intrusion tolerance mechanisms[1, 38].

7.5 Using trusted components

The relation of trust/trustworthiness can be applied in general when architecting IT systems, as we saw in the last section. However, particular instantiations of trusted components deserve mention here.

IT protocols can combine extremely high efficiency with high resilience if supported by *locally accessible* trusted components. For example, the notion of security kernel in IT would correspond to a fail-controlled local subsystem trusted to execute a few security-related functions correctly, albeit immersed in the remaining environment, subjected to malicious faults.

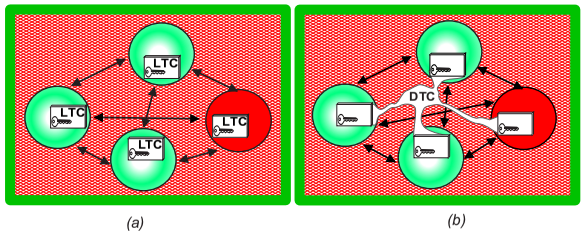


Fig. 12. Using trusted components: (a) Local; (b) Distributed

This can be generalised to any function, such as time-keeping, or failure detection. In that sense, a local trusted component would encapsulate, and supply in a trusted way, a set of functions, considered crucial for protocols and services having to execute in a hostile environment. The use of trusted hardware (e.g. smart cards, appliance boards) may serve to amplify the trustworthiness of these special components. In Figure 12a we see an example of an architecture featuring LTCs (local trusted components). Inter-component communication should ensure that correct components enjoy the properties of the LTC despite malicious faults. On the other hand, the implementation of the LTC should ensure that malicious components, such as the one on the right of Figure 12a, do not undermine the operation of the LTC, making it work incorrectly.

Figure 12b shows a distributed trusted component (DTC). It amplifies the power of a LTC, since it assumes the existence of not only local trusted execution, but also a trusted channel among LTCs. This makes it possible to implement distributed trust for low-level operations (e.g., distribution of message authentication codes- MACS). It can be built for example with appliance boards with a private control channel, such as a second network attachment in a host.

A DTC can assist protocols in number of ways, which we discuss with more detail in later sections of the paper, but the fundamental rationale is the following:

- protocol participants have to exchange messages in a world full of threats, some of them may even be malicious and cheat (the normal network);
- there is a channel that correct participants trust, and which they can use to get in touch with each other, even if for rare and short moments;

- they can use this channel to synchronise, disseminate, and agree on, simple but crucial facts of the execution of a protocol, and this limits the potential for Byzantine actions from malicious participants.

8 Some Example Systems

The term "intrusion tolerance" appeared originally in a paper by Fraga and Powell [19]. Later their scheme –Fragmentation-Redundancy-Scattering– was used in the DELTA-4 project to develop an intrusion-tolerant distributed server composed by a set of insecure sites [16].

In the following years a number of isolated IT protocols and systems emerged. BFT [10] is an efficient state-machine replication algorithm [32]. It has been used to implement an intrusion-tolerant NFS server. Rampart provides tools for building IT distributed services: reliable multicast, atomic multicast and membership protocols [31]. SecureRing is a view-synchronous group communication system based on the Totem single-ring protocols [22]. Both Rampart and SecureRing can be used to build servers using the state-machine replication approach. Fleet [24] use Byzantine quorum systems [2] to build IT data stores, respectively for data abstractions like variables and locks, and for Java objects. The protocol suite CLIQUES supports group key agreement operations for dynamic groups of processes [4, 3]. More recently, two projects have focused on intrusion tolerance, OASIS and MAFTIA, developing several results that will be detailed ahead.

8.1 OASIS

Organically Assured and Survivable Information System (OASIS) [2] is a US DARPA program with the goal of providing "defence capabilities against sophisticated adversaries to allow sustained operation of mission critical functions in the face of known and future cyber attacks against information systems". The program has a strong focus in intrusion tolerance. Its objectives are:

- to construct intrusion-tolerant systems based on potentially vulnerable components;
- to characterize the cost-benefits of intrusion tolerance mechanisms;
- to develop assessment and validation methodologies to evaluate intrusion tolerance mechanisms.

OASIS is financing something like 30 projects. It is not possible to describe all of them so we survey a few that we find interesting and representative.

Intrusion Tolerance by Unpredictable Adaptation (ITUA) aims to develop a middleware to help design applications that tolerate certain classes of attacks [14]. The ITUA architecture is composed by security domains, that abstract the notion of boundaries that are difficult by an attacker to cross (e.g., a LAN protected by a firewall). An intrusion-tolerant application usually has to adapt

[2] http://www.tolerantsystems.org/

when there are attacks. ITUA proposes unpredictable adaptation as a means to tolerate attacks that try to predict and take advantage of that adaptation. Adaptation in ITUA is handled by the QuO middleware and group communication is implemented as intrusion-tolerant layers in the Ensemble toolkit.

Intrusion Tolerant Architectures has the objective to develop a methodology based on architectural concepts for constructing intrusion-tolerant systems. The project developed an IT version of Enclaves, a middleware for supporting secure group applications in insecure networks, like the Internet [18]. IT-Enclaves has several leaders from which at most f out of $n \geq 3f+1$ are allowed to be compromised. The leaders provide all group-management services: user authentication, member join and leave, group-key generation, distribution, and refreshment. Each member of the group is in contact with $2f + 1$ leaders.

COCA is an on-line certification-authority for local and wide-area networks [38]. COCA uses replicated servers for availability and intrusion-tolerance. The certificates that it produces are signed using a threshold cryptography algorithm. COCA assumes an adversary takes a certain time to corrupt a number of servers, therefore from time to time keys are changed (proactive security). Replication is based on a Byzantine quorum system.

Integrity Through Mediated Interfaces is a project that has the objective of providing data integrity [40]. The approach consists in using an Integrity Manager to monitor the programs that manipulate the data, and to record all data transformation. The records produced can be used with several purposes, including the reconstruction of corrupted data. The project designed wrappers for COTS applications, that can be used both with the above mentioned goals, and with the purpose of protecting the environment from malicious code (email attachments, macros, etc.).

ITDBMS is engineering an experimental intrusion-tolerant database system using COTS components in order to provide comprehensive, integrated, and cost effective solutions for IT DBMSs [41]. The approach is based on a multi-layered defence strategy that combines several mechanisms: transaction-level intrusion detection, intrusion isolation, intrusion masking, damage location and confinement, and self-stabilization.

Agile Objects is a framework to construct IT applications based on the ideas of location, interface and dynamic elusiveness [11]. Location elusiveness is the capability of the application components to move across different hosts in order to evade from attacks and corrupted nodes. Interface elusiveness allows the middleware to automatically change the interface of the components. Dynamic elusiveness is the capability of managing the dimensions of location and interface elusiveness.

8.2 MAFTIA

Malicious- and Accidental-Fault Tolerance for Internet Applications (MAFTIA)[3] is a recently finished EU IST project with the general objective of systematically

[3] http://www.maftia.org/

investigating the 'tolerance paradigm' for constructing large-scale dependable distributed applications. The project had a comprehensive approach that includes both accidental and malicious faults. MAFTIA followed three main lines of action:

- definition of an architectural framework and a conceptual model;
- the design of mechanisms and protocols;
- formal validation and assessment.

The first line aimed to develop a coherent set of concepts for an architecture that could tolerate malicious faults [1]. Work has been done on the definition of a core set of intrusion tolerance concepts, clearly mapped into the classical dependability concepts. The AVI composite fault model presented above was defined in this context. Other relevant work included the definition of synchrony and topological models, the establishment of concepts for intrusion detection and the definition of a MAFTIA node architecture. This architecture includes components such as trusted and untrusted hardware, local and distributed trusted components, operating system and runtime environment, software, etc.

Most MAFTIA work was on the second line, the design of IT mechanisms and protocols. Part of that work was the definition of the *MAFTIA middleware*: architecture and protocols [7]. An asynchronous suite of protocols, including reliable, atomic and causal multicast was defined [8], providing Byzantine resilience by resorting to efficient solutions based on probabilistic execution. Work was also done on protocols based on a timed model, which relies on an innovative concept, the *wormholes*, enhanced subsystems which provide components with a means to obtain a few simple privileged functions and/or channels to other components, with "good" properties otherwise not guaranteed by the "normal" weak environment [35]. For example, the Trusted Timely Computing Base developed in MAFTIA (see next two sections) is based on a wormhole providing timely and secure functions on enviroments that are asynchronous and Byzantine-on-failure. Architectural hybridisation discussed earlier is used to implement the TTCB. In the context of MAFTIA middleware, an IT transaction service with support for multiparty transactions[37] was also designed.

Intrusion detection is assumed as a mechanism for intrusion tolerance but also as a service that has to be made intrusion-tolerant. MAFTIA developed a distributed IT intrusion detection system [15]. Problems like handling high rates of false alarms and combining several IDSs were also explored.

Trusted Third Parties (TTPs) such as certification authorities are important building blocks in today's Internet. MAFTIA designed a generic distributed certification authority that uses threshold cryptography and IT protocols in order to be intrusion-tolerant. Another TTP, the distributed optimistic fair exchange service, was also developed.

MAFTIA defined an *authorization service* based on fine grain protection, i.e., on protection at the level of the object method call [26]. The authorization service is a distributed TTP which can be used to grant or deny authorization for complex operations combining several method calls. The service relies on a local security kernel.

The third line of work was on formalizing the core concepts of MAFTIA and verifying and assessing the work on dependable middleware [27]. A novel rigorous model for the security of reactive systems was developed and protocols were modelled using CSP and FDR.

In the next sections, we describe some of our own work in more detail: the construction of a Trusted Timely Computing Base using the principle of architectural hybridisation, and a protocol using the TTCB wormhole.

Architectural Hybridisation in Practice The Trusted Timely Computing Base (TTCB) is a real-time secure wormhole [13]. The TTCB is a simple component providing a limited set of services. Its architecture is presented in Figure 13. The objective is to support the execution of IT protocols and applications using the architectural hybridisation approach introduced before.

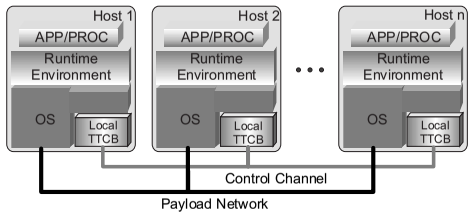


Fig. 13. System architecture with a TTCB

This experimental implementation of the TTCB was based on COTS components. The hosts are common Pentium PCs with a real-time kernel, RT-Linux or RTAI. The hosts are interconnected by two Fast-Ethernet LANs. One corresponds to the payload network in Figure 13, while the other is the TTCB control-channel. It is thus a configuration aimed at local environments, such as sites, campuses, etc. Wide-area configurations are also possible, as discussed in [35].

The design of a system has both functional and non-functional aspects. Next we describe the functionality of the TTCB –its services– and later we discuss the how the security and timeliness (real-time) are enforced in the COTS based TTCB.

The TTCB provides a limited set of services. From the point of view of programming they are a set of functions in a library that can be called by processes in the usual way. We use the word "process" to denominate whatever uses the TTCB services: a normal process, a thread, or another software component.

The TTCB provides three security-related services. The Local Authentication Service allows processes to communicate securely with the TTCB. The service

authenticates the local TTCB before a process and establishes a shared symmetric key between both, using a simple authenticated key establishment protocol. This symmetric key is used to secure all their further communication. Every local TTCB has an asymmetric key pair, and we assume that the process manages to get a correct copy of the local TTCB public key. The Trusted Block Agreement Service is the main building block for IT protocols. This service delivers a value obtained from the agreement of values proposed by a set of processes. The service is not intended to replace agreement protocols in the payload system: it works with "small" blocks of data (currently 160 bits), and the TTCB has limited resources to execute it. The service provides a set of functions that can be used to calculate the result. For instance, it can select the value proposed by more processes. A parameter of the service is a timestamp that indicates the last instant when the service starts to be executed. This prevents malicious processes from delaying the service execution indefinitely. The last security-related service is the Random Number Generation Service that provides uniformly distributed random numbers. These numbers can be used as nonces or keys for cryptographic primitives such as authentication protocols.

The TTCB provides also four time services. The Trusted Absolute Timestamping Service provides globally meaningful timestamps. It is possible to obtain timestamps with this characteristic because local TTCBs clocks are synchronized. The Trusted Duration Measurement Service measures the time of the execution of an operation. The Trusted Timing Failure Detection Service checks if a local or distributed operation is executed in an interval of time. The Trusted Timely Execution Service executes special operations securely and within an interval of time inside the TTCB.

The Trusted Block Agreement Service and the Trusted Timing Failure Detection Service are distributed, therefore they are implemented using communication protocols that run in the TTCB control channel. We do not present these protocols here for lack of space.

RT-Linux and RTAI are two similar real-time engineerings of Linux. Linux was modified so that a real-time executive takes control of the hardware, to enforce real-time behaviour of some real-time tasks. RT tasks were defined as special Linux loadable kernel modules so they run inside the kernel. The scheduler was changed to handle these tasks in a preemptive way and to be configurable to different scheduling disciplines. Linux runs as the lowest priority task and its interruption scheme was changed to be intercepted by RT-Linux/RTAI. The local part of a COTS-based TTCB is basically a (non-real-time) local kernel module, that handles the service calls, and a set of two or more RT tasks that execute all time constrained operations.

The local TTCB is protected by protecting the kernel. From the point of view of security, RT-Linux/RTAI are very similar to Linux. Their main vulnerability is the ability a superuser has to control any resource in the system. This vulnerability is usually reasonably easy to exploit, e.g., using race conditions. Linux capabilities are privileges or access control lists associated with processes that allow a fine grain control on how they use certain objects. However, currently

the practical way of using this mechanism is quite basic. There is a system wide *capability bounding set* that bounds the capabilities that can be held by any system process. Removing a capability from that set disables the ability to use an object until the next reboot. Although basic, this mechanism is sufficient to protect the local TTCB. Removing the capability CAP_SYS_MODULE from the capability bounding set we prevent any process from inserting code in the kernel. Removing CAP_SYS_RAWIO we prevent any process from reading and modifying the kernel memory.

For the COTS-based TTCB we make the assumption that the control channel is not accessed physically. Therefore, security has to be guaranteed only in its access points. To be precise, we must prevent an intruder from reading or writing in the control channel access points. This is done by removing the control network device from the kernel so that it can only be accessed by code in the kernel, i.e., by the local TTCB.

The control channel in the COTS-based TTCB is a switched Fast-Ethernet LAN. The timeliness of that network packet is guaranteed preventing packet collisions which would cause unpredictable delays. This requires that: (1) only one host can be connected to each switch port (hubs cannot be used); and (2) the traffic load has to be controlled. The first requirement is obvious. The second is solved by an access control mechanism, that accepts or rejects the execution of a service taking into account the availability of resources (buffers and bandwidth).

This is a brief presentation of the implementation of the design of the COTS-based TTCB. Further details, including the enforcement of the TTCB properties and the discussion of implementations in other networks and local architectures, can be found in [13].

A Wormhole-Aware Protocol This section presents an IT protocol based on the TTCB wormhole [4]. This protocol illustrates the approach based on hybrid failure assumptions: most of the system is assumed to fail in an arbitrary way, while the wormhole is assumed to be secure, i.e, to fail only by crashing. The system is also assumed to be asynchronous, except for the TTCB which is synchronous.

The protocol is a *reliable multicast*, a classical problem in distributed systems. Each execution of a multicast has one sender process and several recipient processes. In the rest of the section, we will make the classical separation of *receiving* a message from the network and *delivering* a message – the result of the protocol execution.

A reliable multicast protocol enforces the following two properties [6]: (1) all correct processes deliver the same messages; (2) if a correct sender transmits a message then all correct processes deliver this message. These rules do not imply any guarantees of delivery in case of a malicious sender. However, one of two things will happen, either the correct processes never complete the protocol execution and no message is ever delivered, or if they terminate, then they will all deliver the same message. No assumptions are made about the behaviour of

[4] The protocol is a simplified version of the protocol presented in [12].

malicious (recipient) processes. They might decide to deliver the correct message, a distinct message or no message.

The protocol –BRM (Byzantine Reliable Multicast)– is executed by a set of distributed processes. The processes can fail arbitrarily, e.g., they can crash, delay or not transmit some messages, generate messages inconsistent with the protocol, or collude with other faulty processes with malicious intent. Their communication can also be arbitrarily attacked: messages can be corrupted, removed, introduced, and replayed.

Let us see the process failure modes in more detail. A process is *correct* basically if it follows the protocol until the protocol terminates. Therefore, a process is *failed* if it crashes or deviates from the protocol. There are some additional situations in which we also consider the process to be failed. A process has an identity before the TTCB which is associated to the shared key. If that pair *(id,key)* is captured by an attacker, the process can be impersonated before the TTCB, therefore it has to be considered failed.

Another situation in which we consider a process to be failed is when an attacker manages to disrupt its communication with the other processes. Protocols for asynchronous systems typically assume that messages are repeatedly retransmitted and eventually received (reliable channels). In practice, usually a service which is too delayed is useless. Therefore, BRM retransmits messages a limited number of times and then we assume "isolated" processes to be failed. In channels prone only to accidental faults it is usually considered that no more than Od messages are corrupted/lost in a reference interval of time. Od is the *omission degree* and tests can be made in concrete networks to determine Od with the desired probability. For malicious faults, if a process does not receive a message after $Od + 1$ retransmissions from the sender, with Od computed considering only accidental faults, then it is reasonable to assume that either the process crashed, or an attack is under way. In any case, we will consider the receiver process as failed. The reader, however, should notice that Od is just a parameter of the protocol. If Od is set to a very high value, then BRM will start to behave like the protocols that assume reliable channels.

Formally, a reliable multicast protocol has the properties below [20]. The predicate $sender(M)$ gives the message field with the sender, and $group(M)$ gives the "group" of processes involved, i.e., the sender and the recipients (note that we consider that the sender also delivers).

- *Validity:* If a correct process multicasts a message M, then some correct process in $group(M)$ eventually delivers M.
- *Agreement:* If a correct process delivers a message M, then all correct processes in $group(M)$ eventually deliver M.
- *Integrity:* For any message M, every correct process p delivers M at most once and only if p is in $group(M)$, and if $sender(M)$ is correct then M was previously multicast by $sender(M)$.

An implementation of BRM can be found in Figure 14. The sender securely transmits a hash of the message ($H(M)$) to the recipients through the TTCB

BRM-T Sender protocol

```
1  tstart = TTCB_getTimestamp() + T0;
2  M := (elist, tstart, data);
3  propose := TTCB_propose(elist, tstart, TTCB_TBA_RMULTICAST, H(M));
4  repeat Od+1 times do multicast M to elist except sender od
5  deliver M;
```

BRM-T Recipient protocol

```
6  read_blocking(M);
7  propose := TTCB_propose(M.elist, M.tstart, TTCB_TBA_RMULTICAST, ⊥);
8  do decide := TTCB_decide(propose.tag);
9        while (decide.error ≠ TTCB_TBA_ENDED);
10 while (H(M) ≠ decide.value) do read_blocking(M) od
11 repeat Od+1 times do multicast M to elist except sender od
12 deliver M;
```

Fig. 14. BRM protocol.

Agreement Service and then multicasts the message $Od + 1$ times. This hash code is used by the recipients to ensure the integrity and authenticity of the message. When they get a correct copy of the message they multicast it $Od + 1$ times. The pseudo-code is pretty much straightforward so we do not describe it with detail and refer the reader to [12].

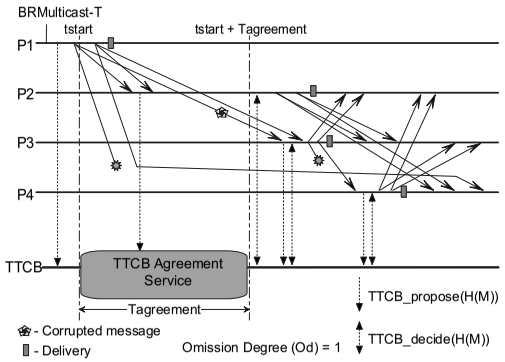


Fig. 15. Protocol execution

Figure 15 illustrates the behavior of the protocol. The horizontal lines represent the execution of processes through time. The thicker line represents the

TTCB as a whole, even though, each process calls a separate local TTCB in its host (this representation is used for simplicity). The sender calls the TTCB agreement and then multicasts the message twice ($Od = 1$). These messages are received in the following way: P2 receives the two copies of the message, P3 receives the first copy corrupted and the second well, and P4 does not receive the first copy and the second is delayed. The example assumes that the first message sent to P3 is corrupted only in the *data* part, and for that reason it is still possible to determine this protocol instance. When a message arrives, the recipient calls the TTCB agreement to get the result with the reliable value of $H(M)$. Both processes P2 and P3 get this value almost immediately after the end of the agreement. They use the hash to select which of the messages they received is correct, and then they multicast the message to all the other recipients. P4 asks for the result of the agreement later, when it receives the first message from the protocol. Then, it multicasts the message.

Bracha and Toueg have shown that, assuming arbitrary faults, it is impossible to send reliable multicasts if there are more than $f = \frac{n-1}{3}$ faulty processes in a system with n processes [6]. BRM imposes constraints on the number of process failures that are similar to accidental fault-tolerant protocols: for f faults, BRM requires $n \geq f + 2$ processes, instead of $n \geq 3f + 1$. In reality, BRM does not impose a minimum number of correct processes, but we say that the number of processes has to be $n \geq f + 2$ to denote the notion that the problem is vacuous if there are less than two correct processes. $n \geq f + 2$ is also the constrain for reliable multicast with arbitrary faults in synchronous systems [42].

The fact that BRM uses the TTCB wormhole allows it also not to need to use asymmetric cryptography, a well known bottleneck in IT protocols. BRM was tested using the COTS-based TTCB [12]. The latencies for 5 processes (one per host) ranged from around 8 to 10ms. This is some times faster than asynchronous protocols in the literature. The current, preliminary, implementation of the TTCB Agreement Service was also identified as the main overhead of the protocol and a faster design is being made. Protocols that use asymmetric cryptography also degrade their performance considerably when we increase the number of processes involved, while protocols based on the TTCB have a very light degradation.

9 Conclusion

We have presented an overview of the main concepts and design principles relevant to intrusion tolerant (IT) architectures. In our opinion, Intrusion Tolerance as a body of knowledge is, and will continue to be for a while, the main catalyst of the evolution of the area of dependability. The challenges put by looking at faults under the perspective of "malicious intelligence" have brought to the agenda hard issues such as uncertainty, adaptivity, incomplete knowledge, interference, and so forth. Under this thrust, researchers have sought replies, sometimes under new names or slight nuances of dependability, such as trustworthiness or survivability.

We believe that fault tolerance will witness an extraordinary evolution, which will have applicability in all fields and not only security-related ones. We will know that we got there when we will no longer talk about accidental faults, attacks or intrusions, but just (and again)... faults.

Acknowledgements

Many of the concepts and design principles presented here derive both from past experience with fault-tolerant and secure system architectures, and from more recent work and challenging discussions within the European IST MAFTIA project. We wish to warmly thank all members of the team, several of whom contributed to IT concepts presented here, and collectively have represented a fenomenal thinking tank.

References

1. Adelsbach, A., Alessandri, D., Cachin, C., Creese, S., Deswarte, Y., Kursawe, K., Laprie, J.C., Powell, D., Randell, B., Riordan, J., Ryan, P., Simmonds, W., Stroud, R., Veríssimo, P., Waidner, M., Wespi, A.: Conceptual Model and Architecture of MAFTIA. Project MAFTIA IST-1999-11583 deliverable D21. (2002) http://www.research.ec.org/maftia/deliverables/D21.pdf.
2. Alvisi, L., Malkhi, D., Pierce, E., Reiter, M.K., Wright, R.N.: Dynamic Byzantine quorum systems. In: Proceedings of the IEEE International Conference on Dependable Systems and Networks. (2000) 283–292
3. Amir, Y., Kim, Y., Nita-Rotaru, C., Schultz, J., Stanton, J., Tsudik, G.: Exploring robustness in group key agreement. In: Proceedings of the 21th IEEE International Conference on Distributed Computing Systems. (2001) 399–408
4. Ateniese, G., Steiner, M., Tsudik, G.: New multi-party authentication services and key agreement protocols. IEEE J. of Selected Areas on Communications **18** (2000)
5. Avizienis, A., Laprie, J.C., Randell, B.: Fundamental concepts of dependability. Technical Report 01145, LAAS-CNRS, Toulouse, France (2001)
6. Bracha, G., Toueg, S.: Asynchronous consensus and broadcast protocols. Journal of the ACM **32** (1985) 824–840
7. Cachin, C., Correia, M., McCutcheon, T., Neves, N., Pfitzmann, B., Randell, B., Schunter, M., Simmonds, W., Stroud, R., Veríssimo, P., Waidner, M., Welch, I.: Service and Protocol Architecture for the MAFTIA Middleware. Project MAFTIA IST-1999-11583 deliverable D23. (2001) http://www.research.ec.org/maftia/deliverables/D23final.pdf.
8. Cachin, C., Poritz, J.A.: Hydra: Secure replication on the internet. In: Proceedings of the International Conference on Dependable Systems and Networks. (2002)
9. Canetti, R., Gennaro, R., Herzberg, A., Naor, D.: Proactive security: Long-term protection against break-ins. RSA CryptoBytes **3** (1997) 1–8
10. Castro, M., Liskov, B.: Practical Byzantine fault tolerance. In: Proceedings of the Third Symposium on Operating Systems Design and Implementation. (1999)
11. Connelly, K., Chien, A.A.: Breaking the barriers: High performance security for high performance computing. In: Proc. New Security Paradigms Workshop. (2002)
12. Correia, M., Lung, L.C., Neves, N.F., Veríssimo, P.: Efficient Byzantine-resilient reliable multicast on a hybrid failure model. In: Proceedings of the 21st IEEE Symposium on Reliable Distributed Systems. (2002) 2–11

13. Correia, M., Veríssimo, P., Neves, N.F.: The design of a COTS real-time distributed security kernel. In: Proceedings of the Fourth European Dependable Computing Conference. (2002) 234–252
14. Cukier, M., Lyons, J., Pandey, P., Ramasamy, H.V., Sanders, W.H., Pal, P., Webber, F., Schantz, R., Loyall, J., Watro, R., Atighetchi, M., Gossett, J.: Intrusion tolerance approaches in ITUA (fast abstract). In: Supplement of the 2001 International Conference on Dependable Systems and Networks. (2001) 64–65
15. Debar, H., Wespi, A.: Aggregation and correlation of intrusion detection alerts. In: 4th Workshop on Recent Advances in Intrusion Detection. Volume 2212 of Lecture Notes in Computer Science. Springer-Verlag (2001) 85–103
16. Deswarte, Y., Blain, L., Fabre, J.C.: Intrusion tolerance in distributed computing systems. In: Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy. (1991) 110–121
17. Dobson, J., Randell, B.: Building reliable secure computing systems out of unreliable insecure components. In: Proceedings of the International Symposium on Security and Privacy, IEEE (1986) 187–193
18. Dutertre, B., Crettaz, V., Stavridou, V.: Intrusion-tolerant Enclaves. In: Proceedings of the IEEE International Symposium on Security and Privacy. (2002)
19. Fraga, J.S., Powell, D.: A fault- and intrusion-tolerant file system. In: Proceedings of the 3rd International Conference on Computer Security. (1985) 203–218
20. Hadzilacos, V., Toueg, S.: A modular approach to fault-tolerant broadcasts and related problems. Technical Report TR94-1425, Cornell University, Department of Computer Science (1994)
21. Hiltunen, M., Schlichting, R., Ugarte, C.A.: Enhancing survivability of security services using redundancy. In: Proceedings of the IEEE International Conference on Dependable Systems and Networks. (2001) 173–182
22. Kihlstrom, K.P., Moser, L.E., Melliar-Smith, P.M.: The SecureRing group communication system. ACM Transactions on Information and System Security **4** (2001) 371–406
23. Knight, J., Heimbigner, D., Wolf, A., Carzaniga, A., Hill, J., Devanbu, P.: The Willow survivability architecture. In: Proceedings of the 4th Information Survivability Workshop. (2001)
24. Malkhi, D., Reiter, M.K., Tulone, D., Ziskind, E.: Persistent objects in the Fleet system. In: Proceedings of the 2nd DARPA Information Survivability Conference and Exposition (DISCEX II). (2001)
25. Meyer, F., Pradhan, D.: Consensus with dual failure modes. In: Proc. of the 17th IEEE International Symposium on Fault-Tolerant Computing. (1987) 214–222
26. Nicomette, V., Deswarte, Y.: An Authorization Scheme for Distributed Object Systems. In: IEEE Symposium on Research in Privacy and Security. (1996) 31–40
27. Pfitzmann, B., Waidner, M.: A model for asynchronous reactive systems and its application to secure message transmission. In: Proceedings of the IEEE Symposium on Research in Security and Privacy. (2001) 184–200
28. Powell, D., Seaton, D., Bonn, G., Veríssimo, P., Waeselynk, F.: The Delta-4 approach to dependability in open distributed computing systems. In: Proceedings of the 18th IEEE International Symposium on Fault-Tolerant Computing. (1988)
29. Powell, D., ed.: Delta-4: A Generic Architecture for Dependable Distributed Processing. Springer-Verlag (1991) Research Reports ESPRIT.
30. Powell, D.: Fault assumptions and assumption coverage. In: Proceedings of the 22nd IEEE International Symposium of Fault-Tolerant Computing. (1992)

31. Reiter, M.K.: The Rampart toolkit for building high-integrity services. In: Theory and Practice in Distributed Systems. Volume 938 of Lecture Notes in Computer Science. Springer-Verlag (1995) 99–110
32. Schneider, F.B.: The state machine approach: A tutorial. Technical Report TR86-800, Cornell University, Computer Science Department (1986)
33. Veríssimo, P., Rodrigues, L.: Distributed Systems for System Architects. Kluwer Academic Publishers (2001)
34. Veríssimo, P., Rodrigues, L., Casimiro, A.: Cesiumspray: a precise and accurate global clock service for large-scale systems. Journal of Real-Time Systems **12** (1997) 243–294
35. Veríssimo, P.: Uncertainty and predictability: Can they be reconciled? In: Future Directions in Distributed Computing. Springer-Verlag LNCS 2584 (2003) –
36. Veríssimo, P., Casimiro, A., Fetzer, C.: The Timely Computing Base: Timely actions in the presence of uncertain timeliness. In: Proceedings of the International Conference on Dependable Systems and Networks. (2000) 533–542
37. Xu, J., Randell, B., Romanovsky, A., Rubira, C., Stroud, R.J., Wu, Z.: Fault tolerance in concurrent object-oriented software through coordinated error recovery. In: Proceedings of the 25th IEEE International Symposium on Fault-Tolerant Computing. (1995) 499–508
38. Zhou, L., Schneider, F., van Renesse, R.: COCA: A secure distributed on-line certification authority. ACM Trans. on Computer Systems **20** (2002) 329–368
39. Gray, J.: Why do computers stop and what can be done about it? In: Proceedings of the 5th IEEE Symposium on Reability in Distributed Software and Database Systems. (1986) 3–12
40. Tallis, M., Balzer, R.: Document integrity through mediated interfaces. In: Proceedings of the 2nd DARPA Information Survivability Conference and Exposition (DISCEX II). (2001)
41. Lui, P.: General design of ItDBMS. Technical report, UMBC (2000)
42. Lamport, L., Shostak, R., Pease, M.: The Byzantine generals problem. ACM Transactions on Programming Languages and Systems **4** (1982) 382–401

Intrusion-Tolerant Middleware: the MAFTIA approach

Paulo E. Veríssimo, Nuno F. Neves,
Christian Cachin, Jonathan Poritz,
David Powell, Yves Deswarte,
Robert Stroud, Ian Welch

DI-FCUL TR–04–14

November 2004

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa
Campo Grande, 1700 Lisboa
Portugal

Technical reports are available at http://www.di.fc.ul.pt/tech-reports. The files are stored in PDF, with the report number as filename. Alternatively, reports are available by post from the above address.

Intrusion-Tolerant Middleware: the MAFTIA approach *

Paulo E. Verissimo, Nuno F. Neves, FCUL- U. Lisboa, Lisboa-Portugal, pjv@di.fc.ul.pt, nuno@di.fc.ul.pt

Christian Cachin, Jonathan Poritz, IBM Zurich Research, Rüschlikon-Switzerland, cca@zurich.ibm.com, jap@zurich.ibm.com

David Powell, Yves Deswarte, LAAS-CNRS, Toulouse-France, dpowell@laas.fr, Yves.Deswarte@laas.fr

Robert Stroud, Ian Welch†, U. Newcastle upon Tyne, Newcastle-UK, R.J.Stroud@ncl.ac.uk, ian.welch@mcs.vuw.ac.nz

Abstract

The pervasive interconnection of systems all over the world has given computer services a significant socio-economic value, which can be affected both by accidental faults and by malicious activity. It would be appealing to address both problems in a seamless manner, through a common approach to security and dependability. This is the proposal of 'intrusion tolerance', where it is assumed that systems remain to some extent faulty and/or vulnerable and subject to attacks that can be successful, the idea being to ensure that the overall system nevertheless remains secure and operational.

In this paper, we report some of the advances made in the European project MAFTIA, namely in what concerns a basis of concepts unifying security and dependability, and a modular and versatile architecture, featuring several intrusion-tolerant middleware building blocks. We describe new architectural constructs and algorithmic strategies, such as: the use of trusted components at several levels of abstraction; new randomization techniques; new replica control and access control algorithms. The paper concludes by exemplifying the construction of intrusion-tolerant applications on the MAFTIA middleware, through a transaction support service.

1 Introduction

The generalized use of computer networks for communication, access to commercial services, research, or simply for entertainment became a reality during the last decade. Some facts emerging from this scenario deserve notice: the explosion of the number of users; the explosion of the number of services provided and/or implemented in a distributed way; the pervasive interconnection of systems all over the world.

Some of these services have significant criticality, not only economic (transactions, e-commerce, industry), but also societal (utilities, telecommunications, transportation), demanding well-defined levels of quality of service. However, this objective may be impaired both by accidental faults and by malicious activity: viruses, worms, direct hacker attacks...[1]

The scenario just described is causing a renewed interest in distributed systems security and dependability, which have taken separate paths until recently. The classical approach to Security has mostly consisted in trying to prevent bad things from happening. In other words, the objective has been to try and develop "perfect" systems, systems without vulnerabilities, and/or to detect attacks and intrusions and deploy ad-hoc countermeasures before the system is affected.

It would be appealing to consider an approach where security and dependability would be taken commonly. After all, the problems to be solved are of similar nature: keeping systems working correctly, despite the occurrence of mishaps, which we could commonly call faults (accidental or malicious); ensure that, when systems do fail (again, on account of accidental or malicious faults), they do so in a non harmful/catastrophic way. This is the proposal of *intrusion tolerance*, a new approach that has emerged during the past decade, and gained impressive momentum recently. The idea can be explained very simply [4]:

- to assume and accept that the systems remain to some extent vulnerable;
- to assume and accept that attacks on components can happen and some will be successful;
- to ensure that the overall system nevertheless remains secure and operational.

In this paper, we report some of the advances made in the European project MAFTIA [2]. The path to understanding and conceiving intrusion-tolerant systems starts with revisiting the basic dependability concepts and "reading" them under a security-related perspective, incorporating specific security properties, fault classifications, and security methods. Under the light of these revised concepts, the words "dependence" and "dependability" relate strongly to notions like "trust" and "trustworthiness", giving the latter a powerful and precise meaning: pointing to generic properties and not just security; defining clear relationships between them. These issues are discussed further in Sections 2 and 3.

Surprising as it may seem, intrusion tolerance is not just another instantiation of accidental fault tolerance. Architecting intrusion-tolerant systems, to arrive at some notion of *intrusion-tolerant middleware* for application support, presents multiple challenges, primarily because several of the paradigms and

*This work was partially supported by the EC, through project IST-1999-11583 (MAFTIA), and by the FCT, through LASIGE and projects POSI/1999/CHS/33996 (DEFEATS) and POSI/CHS/39815/2001 (COPE).

†Currently at Victoria U., New Zealand.

[1] See, e.g., the CERT Coordination Center statistics at http://www.cert.org/stats/.

[2] MAFTIA portal: http://http://www.newcastle.research.ec.org/maftia.

models used in accidental fault tolerance are not adequate for malicious faults: potential for maliciously caused common-mode faults makes probabilistic assumptions risky (number of "independent" faulty components, fault types); error propagation is the rule rather than the exception (error detection delay, progressive intrusion); typical severity of malicious faults (Byzantine behavior, attacks on timing, contamination of runtime support environment). In addressing these challenges, we devised new architectural constructs and algorithmic strategies, for example, programming models based on the use of trusted components at several levels of abstraction; new randomization and cryptographic techniques; new replica and access control algorithms.

Through the rest of the paper, we start by presenting the rationale behind the main architectural options in MAFTIA, and its blocks and functionality, in Section 3. Section 4 describes different strategies that can be followed in such modular and versatile architectures, to create several instances of the *MAFTIA middleware*. In Sections 5 through Section 7, we digress through instantiations of these strategies in several MAFTIA middleware building blocks, reporting on mechanisms and algorithms researched and prototyped in the project, described with detail in other publications. Finally, in Section 8 we exemplify the construction of intrusion-tolerant applications on the MAFTIA middleware, through a transaction support service that appears to the user as a CORBA-style service, intrusion tolerance being achieved transparently.

2 Basic Concepts

The idea that the *tolerance* approach could be applied to intrusions dates back to the 1980's in some early work on the combination of concepts of dependability and fault-tolerance with security [24, 31, 22]. The sequels of this work [1, 6, 30], which we designate hereafter as the *core dependability concepts*, and contributions from the security and intrusion detection communities [17, 29, 38] have been fundamental in establishing the conceptual and terminological framework that has guided the design process of the MAFTIA architecture. In this section, we outline the main elements of this framework [5].

2.1 Faults, errors and failures

Fundamental to the core dependability concepts is the idea that, at a given level of system abstraction or decomposition, there are three causally related impairments to system dependability that need to be considered. A system *failure* is an event that occurs when the service delivered by the system deviates from correct service. An *error* is that part of the system state that may cause a subsequent failure, whereas a *fault* is the adjudged or hypothesized cause of an error. The notion is recursive in that a failure at one level can be viewed as a fault at the next higher level (e.g., a component failure is a fault seen from the containing system).

The labels given to these concepts conform to standard usage within the dependability community, but the important point we would like to stress is not the words but the fact that there are *three* concepts.

First, it is essential to be able to distinguish the internally observable phenomenon (error) from the externally observable one (failure), which tolerance techniques aim to avert. Indeed, any tolerance technique must be based on some form of detection and recovery acting on internal perturbations before they reach the system's interface to the outside world. The alternative viewpoint, in which any detectable anomaly is deemed to make the system "insecure" in some sense, would make intrusion-*tolerance* an unattainable objective.

Second, the distinction between the internally observable phenomenon (error) and its root cause (fault) is vital since it emphasizes the fact that there may be various plausible causes for the same observed anomaly, including not only an intentionally malicious fault, but also an accidental fault, or an atypical usage profile.

These three notions also have an interesting interpretation in terms of a *security policy*, which we consider as comprising both goals and rules. The *goals* are intended to capture security requirements whose violation would be considered as a *security failure*. Typically, those goals are defined in terms of confidentiality, integrity and availability properties on system services, data or metadata. The *rules* defined in a security policy are lower level constraints on system behavior that aim to ensure that the goals are fulfilled. Violations of the rules do not correspond to security failures but are indicative of errors that could lead to security failures if no precautions are taken.

2.2 Attacks, vulnerabilities and intrusions

We consider an intrusion to be a deliberately malicious software-domain operational[3] fault that has two underlying causes (we refer thus to a *composite* fault model):

- A malicious act or *attack* that attempts to exploit a potential weakness in the system,
- At least one weakness, flaw or *vulnerability*.

Vulnerabilities are the primordial faults within the system, in particular design or configuration faults (e.g., coding faults allowing stack overflow, files with root setuid in Unix, naïve passwords, unprotected TCP/IP ports). Vulnerabilities may be introduced during development of the system, or during operation. They may be introduced accidentally or deliberately, with or without malicious intent. As a step in his overall plan of attack, an attacker might introduce vulnerabilities in the form of malicious logic [29].

Attacks may be viewed either at the level of human activity (of the attacker), or at that of the resulting technical activity that is observable within the considered computer system. Attacks (in the technical sense) are malicious faults that attempt to exploit one or more vulnerabilities (e.g., port scans, email viruses, malicious Java applets or ActiveX controls). An attack that successfully exploits a vulnerability results in an intrusion. This further step towards failure is normally characterized by an erroneous state in the system that may take several forms (e.g., an unauthorized privileged account with telnet access, a

[3] As opposed to faults in the hardware domain, e.g., physical sabotage, or to faults introduced during system development, e.g., trapdoors.

system file with undue access permissions for the attacker). Such erroneous states may be corrected or masked by intrusion tolerance but if nothing is done to handle the errors resulting from the intrusion, a security failure will probably occur.

We only qualify the human/technical nature of attacks when necessary; in the absence of qualification, we consider "attack" in its technical sense. *Attacker* is always taken in its human sense, i.e., the malicious person or organization at the origin of attacks. When a technical attack is perpetrated on behalf of the attacker by some piece of code, we refer to the latter as an *attack agent*. Attack agents can be classified according to the following dimensions:

- *dissemination*: propagating (i.e., as in virus or worm); non-propagating (one-off result of an intrusion);
- *trigger conditions*: continuously activated (e.g., an illicit sniffer); serendipitous activation by unsuspecting victim (e.g., Trojan horse); other conditions (specific time, input value, etc.) (i.e., a bomb or a zombie);
- *target of attack*: local (e.g., a bomb or a Trojan horse) or distant (i.e., a zombie);
- *aim of attack*: disclosure (confidentiality); alteration (integrity), denial of service (availability).

A security failure at one level of decomposition of the system may be interpreted as an intrusion propagating to the next upper level. Depending on the adopted viewpoint at that level, the propagated intrusion may also be viewed as an attack, as the installation of a vulnerability, or as an attack agent. Indeed, the propagated intrusion may manifest itself as a further attack (the attacker directly exploits his successful attack in order to proceed towards his final goal); by the creation of new vulnerabilities (e.g., a system file with undue access permissions for the attacker, or malicious logic creating a *trapdoor*) or by the insertion of malicious logic that can act as an agent for the attacker sometime in the future (e.g., a *zombie*).

Finally, when we consider intrusions from an authorization policy viewpoint, we note that they can be subdivided into two types, according to whether an intrusion corresponds to an unauthorized increase in the privilege (set of access rights) of the attacker (or his agent) or to an improper use of authorized operations. We refer to these respectively as *theft* and *abuse* of privilege. Note that theft and abuse of privilege are more general concepts than the often-used notions of "outsider" vs. "insider" intrusions, since (a) a complete "outsider" in an open Internet setting is somewhat difficult to imagine, and (b) an "insider" can attempt both types of intrusions.

2.3 Security methods

The methods underpinning the development of a dependable computing system are classified in the core dependability concepts according to four categories:

- *fault prevention*: how to prevent the occurrence or introduction of faults,
- *fault tolerance*: how to deliver correct service in the presence of faults,
- *fault removal*: how to reduce the number or severity of faults,
- *fault forecasting*: how to estimate the present number, the future incidence, and the likely consequences of faults.

Fault prevention and fault removal are sometimes grouped together as *fault avoidance*; fault tolerance and fault forecasting constitute *fault acceptance*. Note that avoidance and acceptance should be considered as complementary rather than alternative strategies.

It is enlightening to equate "fault" in these definitions with the notions of attack, vulnerability and intrusion defined above. Taking "attack" in both its human and technical senses leads to ten distinct security-building methods out of a total of sixteen (cf. Table 1).

The focus in this paper is on intrusion-*tolerance* techniques, which should be seen as an additional defense mechanism rather than an alternative to the classic set of techniques grouped under the heading intrusion-*prevention* on Table 1.

2.4 Intrusion-tolerance primitives

In the core dependability concepts, fault-tolerance is defined in terms of error detection and subsequent system recovery, the latter consisting of error handling (aimed at eliminating errors from the system state) and fault handling (aimed at preventing faults from being activated again).

By definition, *error-detection* (and error handling) need to be applied to all errors irrespectively of the specific faults that caused them. However, the *design* of an error-detection technique needs to take into account the hypothesized fault model. For example, detection of errors caused by physical faults requires physical redundancy; detection of those caused by design faults requires diversification redundancy; etc. Detection of errors due to intrusions similarly needs an independent reference to which system activity can be compared. One or more of the following may provide that reference:

- *normal activity profiles*, as in anomaly-detection or behavior-based techniques for intrusion detection [20, 27];
- *undesired activity profiles*, as in misuse-detection or knowledge-based techniques for intrusion detection [20, 27];
- *rules* contained within the system's security policy;
- *activity of peer entities* in trust distribution approaches to intrusion-tolerance such as secret-sharing [48], fragmentation-redundancy-scattering [24], etc.

Error-handling may take three forms:

	Attack (human sense)	**Attack** (technical sense)	**Vulnerability**	**Intrusion**
Prevention (how to prevent occurrence or introduction of. . .)	deterrence, laws, social pressure, secret service. . .	firewalls, authentication, authorization. . .	semi-formal and formal specification, rigorous design and management. . .	= attack & vulnerability prevention & removal
Tolerance(how to deliver correct service in the presence of. . .)	= vulnerability prevention & removal, intrusion tolerance		= attack prevention & removal, intrusion tolerance	error detection & recovery, fault masking, intrusion detection and response, fault handling
Removal(how to reduce number or severity of. . .)	physical countermeasures, capture of attacker	preventive & corrective maintenance aimed at removal of attack agents	1. formal proof, model-checking, inspection, test. . . 2. preventive & corrective maintenance, including security patches	$\subseteq$ attack & vulnerability removal, i.e., preventive & corrective maintenance
Forecasting(how to estimate present number, future incidence, likely consequences of. . .)	intelligence gathering, threat assessment. . .	assessment of presence of latent attack agents, potential consequences of their activation	assessment of: presence of vulnerabilities, exploitation difficulty, potential consequences. . .	= vulnerability & attack forecasting

Table 1. Classification of security methods

- *roll-back*: state transformation is carried out by bringing the system back to a previously occupied state, for which a copy (a recovery point, or "checkpoint") has been previously saved — extreme examples include operating system reboots, process re-initialization, and TCP/IP connection resets;
- *roll-forward*: state transformation is carried out by finding a new state from which the system can operate — replacement of compromised key shares in threshold-cryptography schemes is an example of this form of error-handling in the context of intrusion tolerance;
- *compensation*: state transformation is carried out by exploiting redundancy in the data representing the erroneous state — masking is the most common form of compensation and is ideally suited for intrusion-tolerance since it can accommodate arbitrarily (e.g., Byzantine) faulty behavior; specific examples include voting, fragmentation-redundancy-scattering and other trust distribution approaches (cf. Sections 5- 8).

Fault handling covers the set of techniques aimed at preventing faults from being re-activated. Whereas error handling is aimed at averting imminent failure, fault handling aims to tackle the underlying causes, whether or not error handling was successful, or even attempted. Three fault-handling primitives can be defined: fault diagnosis, fault isolation and system reconfiguration.

Fault diagnosis is concerned with identifying the type and location of faults that need to be isolated before carrying out system reconfiguration or initiating corrective maintenance. For an intrusion-tolerant system, an essential aspect of diagnosis is the decision as to whether the underlying cause of detected errors was a deliberate attack or an accidental fault. According to the composite fault model presented earlier in this section, fault diagnosis can be further decomposed into:

- intrusion diagnosis, i.e., trying to assess the degree of success of the intruder in terms of system corruption;
- vulnerability diagnosis, i.e., trying to understand the channels through which the intrusion took place so that corrective maintenance can be carried out;
- attack diagnosis, i.e., finding out who or what organization is responsible for the attack in order that appropriate litigation or retaliation may be initiated.

Fault isolation aims to ensure that the source of the detected error(s) is prevented from producing further error(s). In terms of intrusions, this might involve, for example: blocking traffic from components diagnosed as corrupt by changing the settings of firewalls or routers; removing corrupted data from the system; uninstalling software versions with newly-found vulnerabilities; arresting the attacker; etc.

System reconfiguration consists of a redeploying fault-free resources so as to: (a) provide an acceptable, but possibly degraded service while corrective maintenance is carried out on faulty resources, and (b) restore nominal service after corrective maintenance. In an intrusion-tolerant system, possible reconfiguration actions include: software downgrades or upgrades; changing a voting threshold; deployment

of countermeasures including probes and traps (honey-pots) to gather further information about the intruder, and so assist in attack diagnosis.

3 MAFTIA Architecture

The purpose of this section is to introduce the basic models and assumptions underlying the design of the MAFTIA architecture, and then to present an overview of the architecture itself from various perspectives. The section details both the functional aspects of the architecture, and the constructs aimed at achieving intrusion tolerance.

The adjectives “trusted” and “trustworthy” are central to many arguments about the dependability of a system. In the security literature, the terms are often used inconsistently. The MAFTIA notions of “trust” and “trustworthiness” point to generic properties and not just security, and there is a well-defined relationship between them — in that sense, they relate strongly to the words “dependence” and “dependability”. *Trust* is the reliance put by a component on some properties of another component, subsystem or system. In consequence, a *trusted component* has a set of properties that are relied upon by another component (or components), i.e., there is an *accepted dependence*. The term *trustworthiness* is essentially synonymous to dependability, but is often the preferred term when the focus is on external faults such as attacks.

The definitions above have consequences for the design of intrusion tolerant systems [55] since one can reason separately about trust and trustworthiness. There is separation of concerns between what to do with the trust placed on a component (e.g., designing algorithms that assume that the component exhibits given properties), and how to achieve or show its trustworthiness (e.g., constructing and validating the component that displays the assumed properties). The practical use of these guidelines is exemplified in later sections.

3.1 Models and assumptions

3.1.1 Failure assumptions

A crucial aspect of any fault-tolerant architecture is the fault model upon which the system architecture is conceived, and component interactions are defined. A system fault model is built on assumptions about the way system components fail. Classically, these assumptions fall into two kinds: *controlled failure* assumptions, and *arbitrary failure* assumptions.

Controlled failure assumptions specify constraints on component failures. For example, it may be assumed that components only have timing failures. This approach represents very well how common systems work under the presence of accidental faults, failing in a benign manner most of the time. However, it is difficult to model the behavior of a hacker, so there is a problem of coverage that does not recommend this approach for malicious faults, unless a trustworthy solution can be found.

Arbitrary failure assumptions ideally specify no constraints on component failures. In this context, an arbitrary failure means the capability of generating a message at any time, with whatever syntax and semantics (form and meaning), and sending it to anywhere in the system. Practical systems based on arbitrary failure assumptions do however specify quantitative bounds on the number of failed components. Arbitrary failure assumptions are costly to handle, in terms of performance and complexity, and thus are not compatible with the user requirements of the vast majority of today’s on-line applications.

Hybrid failure assumptions combining both kinds of failures are a way out of this dilemma [33]. For instance, some nodes are assumed to behave arbitrarily while others are assumed to fail only by crashing. The probabilistic foundation of such distributions might be hard to sustain in the presence of malicious intelligence, unless this behavior is constrained in some manner.

With hybrid assumptions some parts of the system are justifiably assumed to exhibit fail-controlled behavior, whilst the remainder of the system is still allowed an arbitrary behavior. This is an interesting approach for modular and distributed system architectures such as MAFTIA, but one that is only feasible when the fault model is substantiated, that is, the behavior assumed for every single subset of the system can be modeled and/or enforced with high coverage. As a matter of fact, a system normally fails by its weakest link, and naïve assumptions about a component’s behavior will be easy prey to hackers.

A first step towards our objective is the organization of the diverse causes of security failure into a *composite fault model* (see Section 2.2), with a well-defined relationship between attack, vulnerability, and intrusion. Such a model allows us to modularize our approach to achieving dependability, by combining different techniques and methods tackling the different classes of faults defined (see Table 1).

3.1.2 Enforcing hybrid failure assumptions

The second step is the enforcement of hybrid failure assumptions. A composite fault model with hybrid failure assumptions is one where the presence and severity of vulnerabilities, attacks and intrusions varies from component to component. Our work might best be described as *architectural hybridization*, in the line of precursor works such as [40] where failure assumptions are in fact enforced by the architecture and the construction of the system components, and thus substantiated.

Consider a component or sub-system for which a given controlled failure assumption is made. How can we achieve coverage of such an assumption, given the unpredictability of attacks and the elusiveness of vulnerabilities? The answer lies in the combined use of intrusion prevention techniques and the implementation of internal intrusion-tolerance mechanisms. The combination of these techniques should be guided by the composite fault model mentioned above (i.e., removing vulnerabilities that are matched by attacks we cannot prevent; preventing or tolerating attacks on vulnerabilities we cannot remove, etc.). In the end, we should justifiably achieve confidence that the component behaves as assumed, failing in a controlled manner, i.e., that the component can be *trusted* because it is *trustworthy*. The measure of this trust is the coverage of the controlled failure assumption.

3.1.3 Intrusion tolerance under hybrid failure assumptions

The approach outlined in the previous sections: establishes a divide-and-conquer strategy for building modular fault-tolerant systems, with regard to failure assumptions; can be applied to achieve different behaviors in different components; can be applied recursively at as many levels of abstraction as are found to be useful. We are now ready to implement our system-level intrusion-tolerance mechanisms, using a mixture of arbitrary-failure and controlled-failure components. By construction, the behavior of the latter vis-à-vis malicious faults is restricted.

As said earlier, in MAFTIA we trust components or subsystems (we will just use the word component henceforth) *to the extent of* their trustworthiness, as perceived at the adequate instances: by the designer, tester, reviewer, user (human or another component), etc. These components can subsequently be used in the construction of fault-tolerant protocols under architectural hybrid failure assumptions.

This is an innovative aspect in MAFTIA that we explore in the following sections. Note that the soundness of the approach does not depend on our making possibly naïve assumptions about what a hacker can or cannot do to a component. In properly designed systems, the trust placed on a component should be qualitatively and/or quantitatively commensurate to its trustworthiness. Likewise, although the accurate provision of such quantification is currently beyond the state-of-the-art, research here is very active, and constitutes one of the most interesting challenges in intrusion-tolerant system architecture and design.

This approach allows us to construct implementations of fault-tolerant protocols that are more efficient than protocol implementations that have to deal with truly arbitrary failure assumptions for all components, and more robust than designs that make controlled failure assumptions without enforcing them.

In our architectural experiments, we devised three main instances of trusted components. The first is based on a Java Card, and is a local component designed to assist the crucial steps of the execution of services and applications. The second is a distributed component (named Trusted Timely Computing Base), based on appliance boards with private network adapters, that is designed to assist crucial steps of the operation of middleware protocols. Whereas these two instances could be best seen as low-level run-time support components, the third instance concerns distributed trusted components in the middleware, recursively built over the low-level trusted components, through distributed fault-tolerance mechanisms.

3.1.4 Arbitrary failure assumptions considered necessary

Notice that the hybrid failure approach, no matter how resilient, relies on the coverage of the fail-controlled assumptions. Definitely, there will be a significant number of operations whose value and/or criticality is such that the risk of failure due to violation of these assumptions cannot be incurred.

In consequence, an important area of research we pursued is related to arbitrary-failure resilient building blocks, namely communication protocols of the Byzantine class, which do not make assumptions

on the existence of trusted or controlled-failure components. They reason in terms of admitting any behavior from the participants, and allow the corruption of a parameterizable number of participants, say f. The system works correctly as long as there exist $n > 3f$ participants. These protocols do not make assumptions about timeliness either, and are in essence time-free. This has implications on the operational aspects, which will be further discussed in Section 5.

3.2 Architecture description

In this section, we provide an overview of the MAFTIA architecture and discuss the various options that it offers at the hardware, local executive and distributed software levels. The MAFTIA architecture is highly modular. This is an accepted design principle for building distributed fault tolerance into systems. It facilitates the definition of different redundancy strategies for different components, and the placement of the relevant replicas.

3.2.1 Main architectural options

The structure of a MAFTIA host relies on a few main architectural options, some of which are natural consequences of the discussions in the previous section:

The notion of *trusted* — versus untrusted — *hardware*. Most of MAFTIA's hardware is considered to be untrusted, but small parts of it are considered to be trusted to the extent of some quantifiable measure of trustworthiness, for example, being *tamper-proof* by construction. Note that this notion does not necessarily imply proprietary hardware, but for example COTS hardware whose architecture and interface with the rest of the system justifies the aforementioned assumption.

The notion of *trusted support software.* This particular kind of trusted component materializes the notion of a fail-controlled subsystem in the run-time support. It is trusted to execute a few functions correctly (which, given the scope of MAFTIA, will normally be *security-related*) albeit immersed in an environment subjected to malicious faults. The use of trusted hardware may help to substantiate this assumption.

The notion of *run-time environment*, extending operating system capabilities and hiding heterogeneity amongst host operating systems by offering a homogeneous API and framework for protocol composition. Functions supplied by the above-mentioned trusted support software are offered through the run-time API.

Modular and multi-layered *middleware*, with a neat separation between: the multipoint network abstraction, the communication support services, and the activity support services. A given middleware layer may implement another instantiation of a trusted MAFTIA component: a *trusted distributed component* that overcomes the faulty behavior of lower layers and provides certain functions in a trustworthy way, characterized by a given failure semantics (resilience in number and severity of faults).

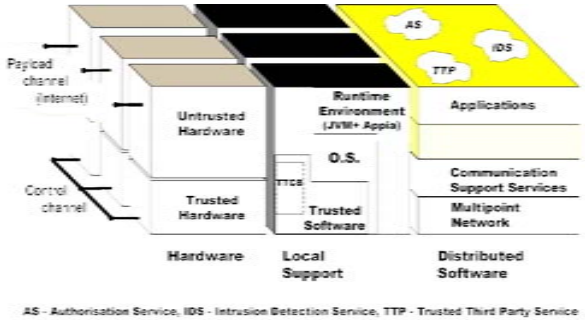


Figure 1. MAFTIA architecture dimensions

The MAFTIA architecture can be depicted in at least three different dimensions (*see* Figure 1). First, there is the *hardware* dimension, which includes the host and networking devices that make up the physical distributed system. Second, within each node, there are the *local support* services provided by the operating system and the run-time platform. These may vary from host to host in a heterogeneous system, and some services may even not be available on some hosts or may have to be accessed via the network using protocols providing an appropriate degree of trust. However, at a minimum, the local services include typical operating system functionality such as the ability to run processes, send messages across the network, access local persistent storage (if it exists), etc. Third, there is the *distributed software* provided by MAFTIA: the layers of *middleware*, running on top of the run-time support mechanisms provided by each host; and MAFTIA's native *services*, depicted in the picture — authorization, intrusion detection, and trusted third party services. Applications built to run on top of MAFTIA use the abstractions provided by the middleware and the application services to operate securely across several hosts, and/or be accessed securely by users running on remote nodes, even in the presence of malicious faults.

3.2.2 Hardware

We assume that the hardware in individual MAFTIA hosts is untrusted in general. Most of a host's operations run on untrusted hardware, e.g., the usual machinery of a PC or workstation, connected through the normal networking infrastructure to the Internet, which we call the *payload channel*. However, some hosts (*see* Figure 1) may have pieces of hardware that are trusted to the extent of being regarded as tamper-proof, i.e., we assume that intruders do not have direct access to the inside of the component.

Some hosts, for example, servers, will have trusted hardware components. Currently, we consider two

incarnations of such hardware, both readily available as COTS components. One is a *Java Card reader*, connected to the machine's hardware, and interfaced by the operating system. The Java Card stores keys and executes software functions to which an attacker does not have access. The other type of trusted hardware is an *appliance board with processor*. Such a board is a common accessory in the PC family that has its own resources and is interfaced by the operating system. The board has a network adapter to a private network, which we call a *control channel* (to differentiate it from the payload channel). We assume that an attacker does not have access either to the interior of the board or to the information circulating in the control channel.

Note that, contrary to the traditional security view of the term "tamper-resistance" to denote a downgraded version of "tamper-proof-ness", we separate concerns between what is assumed ("tamper- proofness") and the merit of that assumption (its coverage), which may be imperfect. For example, the Java Card is assumed in MAFTIA terminology to be tamper-proof, but this quality is trusted to the extent we believe it is worthy of that trust. The next section shows that trust to be a limited one.

3.2.3 Local support

The local support dimension of the architecture (*see* Figure 1) consists essentially of the operating system augmented with appropriate extensions. We have adopted Java as a platform-independent and object-oriented programming environment, and thus our middleware, service and application software modules are constructed to run on the Java Virtual Machine (JVM) run-time environment. The MAFTIA run-time support also includes the APPIA protocol kernel [34] which supports the construction of middleware protocols from the composition of micro-protocols. The run-time support thus includes abstractions of typical local platform services such as process execution, inter-process communication, access to local persistent storage, and protocol management, enhanced with specialized functions provided by the trusted support software, implemented in two components, the Java Card Module (JCM) and the Trusted Timely Computing Base (TTCB).

The Java Card Module (JCM) is used to assist the operation of a reference monitor, which supports the MAFTIA Authorization Service (*see* Section 7). The reference monitor checks all accesses to local objects, whether persistent or transient, and autonomously manages all access rights for local transient objects. The JCM runs partly on the operating system kernel (the reader interface part) and partly on the Java Card (the function's logic and the data structures, e.g., keys). Software components interact with it through the run-time support (the JVM). The Java Card is trusted to the following extent: the effort, in means or time, necessary to subvert it is incommensurate with the consequences of violating the assumption of JCM resilience.

The Trusted Timely Computing Base (TTCB) is a distributed trusted support component responsible for providing a basic set of trusted services related to time and security, to middleware protocols (communication and activity support). The TTCB is designed to act as an assistant for parts of the execution

of the protocols and applications supported by the MAFTIA middleware, and consequently it can be called from any level of the middleware dimension of the architecture. It aims to support malicious-fault tolerant protocols of any synchrony built to a fail-controlled model, such as reliable multicast, by supplying reliable failure-detection and other control information dissemination. In essence, this component implements some degree of distributed trust for low-level operations. That is, protocol participants essentially exchange their messages in a world full of threats, some of them may even be malicious and cheat, but there is an oracle that correct participants can trust, and a channel that they can use to get in touch with each other, even for rare moments. Moreover, this oracle also acts a point of synchronization for all participants, which limits the potential for Byzantine action (inconsistent value faults) by malicious protocol participants. The other important characteristic is that the TTCB is synchronous, in the sense of having reliable clocks and being able to execute timely functions. Furthermore, the control channel provides timely (synchronous) and ordered communication among TTCB modules, providing simple ways to work around the FLP impossibility result. A local TTCB runs partly on the operating system kernel (the appliance board interface part), and partly on the appliance board itself. Software components interact with it through the run-time support (the JVM). The TTCB component is trusted to the following extent: it is assumed to be not feasible to subvert the TTCB, but it may be possible to interfere in its interaction with software components through the JVM. Whilst we let a local host be compromised, we make sure that it does not undermine the distributed TTCB operation.

The TTCB would normally be built on dedicated hardware modules, with a dedicated network. However, we have also designed simpler configurations not requiring dedicated trusted hardware for the TTCB. The software-based solution consists of a small secure real-time kernel running on the bare machine hardware, inside which the TTCB is built, and on top of which the regular operating system runs (and all the rest of the host software) [19]. Note that the coverage expected of this configuration cannot be worse than security-hardened versions of known commercial operating systems. It might actually be better, since it only addresses the inner kernel and not the operating system as a whole. It may thus constitute a very attractive implementation principle, for MAFTIA and in general, for its cost/simplicity/resilience trade-off. The control channel can also assume several forms exhibiting different levels of timeliness and resilience, as detailed in [19]: it may or may not be based on a physically different network from the one supporting the *payload* channel; secure virtual private networks linking all TTCB modules together can be built over alternative networks, such as ISDN or GSM/UMTS.

3.2.4 Middleware

The distribution dimension impacts on the protocol design but not on the services provided by each host. These are constructed on the functionality provided by the several middleware modules, represented in Figure 1. These interactions occur through the run-time environment. The several profiles for building protocols, which will be detailed in the sections ahead, are achieved by composition of

the micro-protocols necessary to achieve the desired quality of service. The middleware hides these distinctions from the application programmer by providing uniform APIs that are parameterized with functional and non-functional guarantees. The design of these APIs is explained in more detail in [35].

As mentioned earlier, a middleware layer may host a trusted distributed component that overcomes the fault severity of lower layers and provides certain functions in a trustworthy way. These are in turn trusted by the layers above, in a recursive way. For example, a (distributed) transactional service trusts that a (distributed) atomic multicast component ensures the typical properties (agreement and total order), regardless of the fact that the underlying environment may suffer Byzantine malicious attacks.

Figure 2 details the middleware layers. We distinguish between site and participant parts, depending on whether the functionality provided is host-global or not, respectively. The site part has access to and depends on a physical networking infrastructure, not represented for simplicity, and multiplexes host-global services to any participant-level module. The participant part offers support to local participants (e.g., user applications) engaging in distributed computations. The lowest layer is the *Multipoint Network* module, *MN*, created over the physical infrastructure. Its main properties are the provision of multipoint addressing, basic secure channels, and management communications. The MN layer hides the particularities of the underlying network to which a given site is directly attached, and is as thin as the intrinsic properties of the former allow. It also provides a run-time (JVM and APPIA) compliant interface for the protocols to be used (e.g., IP, IPSEC, SNMP).

The *Communication Support Services* module, *CS*, implements basic cryptographic primitives, Byzantine agreement, group communication with several reliability and ordering guarantees, clock synchronization, and other core services. The CS module depends on the MN module to access the network. The *Activity Support Services* module, *AS*, implements building blocks that assist participant activity, such as replication management (e.g., state machine, voting), leader election, transactional management, authorization, key management, and so forth. It depends on the services provided by the CS module.

The block on the left of the figure implements failure detection and membership management. *Site failure detection* is in charge of assessing the connectivity and correctness of sites, whereas *participant failure detection* assesses the liveness of local participants, based on local information provided by sensors in the operating system and run-time support. *Membership* management, which depends on failure information, creates and modifies the membership (registered members) and the view (currently active, or non-failed, or trusted members), of sets of sites and of participant groups. Both the AS and CS modules depend on this information.

4 Intrusion Tolerance Strategies in MAFTIA

The goal of MAFTIA is to support the construction of dependable trustworthy applications, implemented by collections of components with varying degrees of trustworthiness. This is achieved by relying on distributed fault and intrusion-tolerance mechanisms. Given the variety of possible MAFTIA ap-

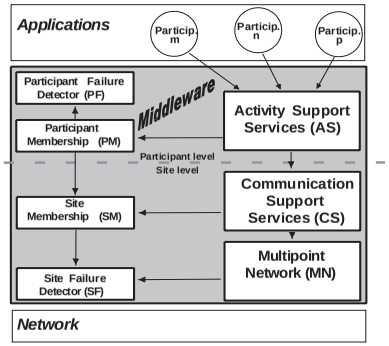


Figure 2. Detail of the MAFTIA middleware

plications, several different architectural strategies are pursued in order to achieve the above-mentioned goal. These strategies are applied at several levels of abstraction of the architecture, most importantly, in the implementation of the middleware and application services. In this section, we describe these strategies: fail-uncontrolled or arbitrary; fail-controlled with local trusted components; fail-controlled with distributed trusted components.

The conventions used for the figures in the following sections are as follows: grey means untrusted (the darker, the "less trusted"); white means trusted; the presence of a clock symbol means a synchronous environment; a crossed out clock symbol means an asynchronous environment; a warped clock symbol means a partially-synchronous environment; a key means a secure environment; dashed arrows means IPC or communication that can be interfered with; continuous arrows denote trusted paths of communication.

4.0.5 Fail-uncontrolled

The fail-uncontrolled or arbitrary failure strategy is based on the no-assumptions attitude discussed in Section 3. When very large coverage is sought of given mechanisms in MAFTIA, we resort to making no assumptions about time, following an asynchronous model, and we make essentially no assumptions about the faulty behavior of either the components or the environment. Of course, for the system as a whole to provide useful service, it is necessary that at least some of the components are correct. This approach is essentially parametric: it will remain correct if a sufficient number of correct participants exist, for any hypothesized number of faulty participants f.

Figure 3 shows the principle in simple terms. The hosts and the communication environment are not trusted, and are fully asynchronous. For a protocol to be able to provide correct service, it must cope

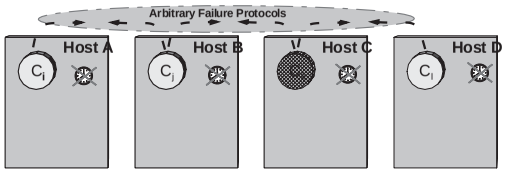


Figure 3. Fail-uncontrolled

with arbitrary failures of components and the environment. For example, component Ck is malicious, but this may be because the component itself or host C have been tampered with, or because an intruder in the communication system simulates that behavior.

Some protocols used by the MAFTIA middleware follow this strategy, in order to be resilient to arbitrary failure assumptions. They are of the probabilistic Byzantine class, and require a number of hosts $n > 3f$, for f faulty components. The MAFTIA middleware provides different qualities of service in this asynchronous profile (see Section 5), achieved by composition of several micro-protocols on top of basic binary Byzantine agreement, in order to achieve: reliable broadcast, atomic broadcast; multi-valued Byzantine agreement.

4.0.6 Fail-controlled with local trusted components

Figure 4 exemplifies a fail-controlled strategy. It consists of assuming that, as for the fail-uncontrolled strategy, hosts and communication environment are not trusted, and asynchronous. However, hosts have a local trusted component (LTC), which supports functions they can trust for certain steps of their operation. In MAFTIA, this strategy is implemented through a Java Card that equips some hosts. As such, we can construct protocols that cope with a hybrid of arbitrary and fail-silent behavior, depending on whether a component is interacting with the other components or with the local trusted component (LTC).

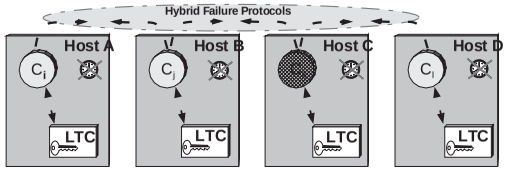


Figure 4. Fail-controlled with local trusted components

In the example, component C_k may be arbitrarily malicious, either because the component itself

or host C has been tampered with, or because an intruder in the communication system simulates that behavior. However, unlike the fail-uncontrolled strategy, the impact of this behavior on the other components (i.e., error propagation) may be limited, if the protocol makes components perform certain checks and validations with the LTC (for example, signature validation), which will prevent C_k from causing certain failures in the value domain (for example, forging). An additional proviso must be made: since the host environment is untrusted, IPC between a component and its LTC may be interfered with, though in a controlled way. For example, if host B is contaminated, component C_j may behave erroneously, but protocols can be designed in a way that prevents C_j from behaving in an arbitrary (e.g. Byzantine) way towards the other hosts.

This strategy is followed in the construction of the MAFTIA authorization service, described in Section 7. Components run distributed fault-tolerant authorization protocols based on capabilities that express the access control for objects. These protocols run among the authorization server replicas and the hosts running a MAFTIA application. Given the criticality of the authorization service, it is also worthwhile noting that the trust put on the Java Card LTC for this application is not absolute, in the sense that the higher-level protocols are ready to cope with the possibility of subversion of some Java Card modules and still ensure globally correct operation of the service. This is an excellent example of the innovative approach we take to trustworthy computing: components are trusted to the extent of their trustworthiness.

4.0.7 Fail-controlled with distributed trusted components

The "fail-controlled with distributed trusted components" strategy amplifies the scope of trustworthiness of the local component support, by making it distributed. As such, certain global actions can be trusted, despite a generally malicious communication environment. This strategy is implemented in MAFTIA through the TTCB (Trusted Timely Computing Base), which builds trust on global (distributed) time-related and security-related properties (such as global time, distributed durations, block agreement). One main impact of relying on the TTCB is that timed behavior can be supported globally in an intrusion-resilient way, as suggested by the warped clocks in Figure 5: the system is assumed to be *partially synchronous*, that is, anywhere in the interval ranging from time-free to fully synchronous, depending on the environment. This strategy assumes, as for the preceding strategies, that the hosts and communication environment are not trusted.

The distributed trusted component (DTC) is implemented by the local TTCBs interconnected by a control network. As with the "fail-controlled with local trusted components" strategy, in order for a protocol to be able to provide useful service, it has to cope with a hybrid of arbitrary and fail-silent behavior, depending on whether a component is interacting with the other components or with the TTCB. Consider the example of Figure 5, where again component C_k or host C may be arbitrarily malicious. Like the "fail-controlled with local trusted components" strategy, the impact of the faulty behavior of

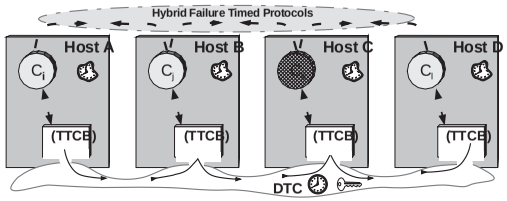


Figure 5. Fail-controlled with distributed trusted components

these components may be limited by enforcing certain validations with the local TTCB. However, the fact that the TTCBs are interconnected and can exchange information and perform agreement in a secure way — through the control channel — further limits the potential damage of malicious behavior: the DTC 'knows' directly what each of the components in different hosts 'say', unlike the solution with LTCs, where an LTC only 'knows' what a remote component 'says', through the local component. To achieve this, the TTCB allows the set-up of secure channels with any local component, and offers a low-level block consensus primitive. For example, components C_i through C_l could set up secure IPC with the TTCB, through which they would run such a consensus as part of the execution of some protocol.

The other relevant aspect of the TTCB strategy is time. The TTCB supports timed behavior in an intrusion-resilient way. As discussed in Section 3, timed systems are fragile in that timing assumptions can be manipulated by intruders. The TTCB supplies constructs that enable protocols to tolerate this class of intrusions. These are obviously related to the trusted time-related services briefly described earlier, namely absolute time, duration measurement and timing failure detection. As suggested in Figure 5, the TTCB DTC is a fully synchronous subsystem. It supplies its services to the payload system, which can have any degree of synchronism, as suggested by the warped clock. The TTCB does not make the payload system "more synchronous", but allows it to take advantage of its possible synchronism, in the presence of faults, both accidental and malicious. As such, the TTCB can assist an application running on the payload system to determine useful facts about time: for example, be sure it executed something on time; measure a duration; determine it was late doing something, etc. Then, the payload system, despite being imperfect (it suffers timing faults, some of which may result from attacks), can react (implement fault-tolerance mechanisms) based on reliable information about the presence or absence of errors (provided by the TTCB at its interface).

Depending on the type of application, it is not necessary that all sites have a local TTCB. Consider the development of a fault-tolerant TTP (Trusted Third Party) based on a group of replicas that collectively ensure the correct behavior of the TTP service vis-à-vis malicious faults. The nodes hosting these replicas have TTCBs that support the execution of the group communication and replica management protocols under a timed model.

Several of the MAFTIA middleware protocols follow the "fail-controlled with TTCB" strategy. These protocols are group-oriented, deterministic, and can provide timeliness guarantees. The MAFTIA middleware provides different qualities of service in this timed profile by composing several micro-protocols on top of basic unreliable multicast. For example, this is the way in which reliable multicast and atomic multicast protocols described in Section 6 are achieved.

5 Byzantine Agreement: the arbitrary approach

As discussed before, an established way for enhancing the fault tolerance of a server is to distribute it among a set of servers and to use replication algorithms for masking faulty servers. Thus, no single server has to be trusted completely and the overall system derives its integrity from a majority of correct servers.

In this section, we describe a configuration of the MAFTIA architecture for distributing trusted services among a set of servers that guarantees liveness and safety of the services despite some servers being under control of an attacker or failing in arbitrary malicious ways. In this configuration, the system model does not include timing assumptions and is characterized by a static set of servers with point-to-point communication and by the use of modern cryptographic techniques. Trusted applications are implemented by deterministic state machines replicated on all servers and initialized to the same state. Client requests are delivered by an atomic broadcast protocol that imposes a total order on all requests and guarantees that the servers perform the same sequence of operations; such an atomic broadcast can be built from a randomized protocol to solve Byzantine agreement. We use efficient and provably secure agreement and broadcast protocols that have recently been developed.

In the first part of this section, we provide a detailed discussion of these assumptions, compare them to related efforts from the literature, and argue why we believe that these choices are adequate for trusted applications in an Internet environment. In the second part, a brief overview of the architecture and protocols is given. Our main tool is a protocol for atomic broadcast, which builds on reliable broadcast and multi-valued Byzantine agreement in an asynchronous network.

5.1 Model

In our model, the system consists of a static set of n servers, of which up to t may fail in completely arbitrary ways, and an unknown number of possibly faulty clients. All parties are linked by asynchronous point-to-point communication channels. Without loss of generality we assume that all faulty parties are controlled by a single adversary, who also controls the communication links and the internal clocks of all servers. The adversary is an arbitrary but computationally bounded algorithm. Faulty parties are called *corrupted*, the remaining ones are called *honest*. Furthermore, there is a trusted dealer that generates and distributes secret values to all servers once and for all, when the system is initialized. The system can process a practically unlimited number of requests afterwards.

This model falls under the impossibility result of Fischer, Lynch, and Paterson [23] of reaching consensus by deterministic protocols. Many developers of practical systems seem to have avoided this model in the past for that reason and have built systems that are weaker than consensus and Byzantine agreement. However, Byzantine agreement can be solved by randomization in an expected constant number of rounds only [13]. Although the first randomized agreement protocols were more of theoretical interest, some practical protocols have been developed recently. For example, the randomized agreement protocol of [10] is based on modern, efficient cryptographic techniques with provable security and withstands the maximal possible corruption.

In our system, we use Byzantine agreement as a primitive for implementing atomic broadcast, which in turn guarantees a total ordering of all delivered messages. Atomic broadcast is equivalent to Byzantine agreement in our model and thus considerably more expensive than reliable broadcast, which only provides agreement of the delivered messages, but no ordering (see Section 5.2).

Below we elaborate on the three key features of our model: cryptography, asynchronous communication, and a static server set.

Cryptography. Cryptographic techniques such as public-key encryption schemes and digital signatures are crucial already for many existing secure services. For distributing a service, we need distributed variants of them from *threshold cryptography*.

Threshold cryptographic schemes are non-trivial extensions of the classical concept of secret sharing in cryptography. Secret sharing allows a group of n parties to share a secret such that t or fewer of them have no information about it, but $t+1$ or more can uniquely reconstruct it. However, one cannot simply share the secret key of a cryptosystem and reconstruct it for decrypting a message because as soon as a single corrupted party knows the key, the cryptosystem becomes completely insecure and unusable.

A *threshold public-key cryptosystem* looks similar to an ordinary public-key cryptosystem with distributed decryption. There is a single public key for encryption, but each party holds a *key share* for decryption (all keys were generated by a trusted dealer). A party may process a decryption request for a particular ciphertext and output a decryption share together with a proof of its validity. Given a ciphertext resulting from encrypting some message and more than t valid decryption shares for that ciphertext, it is easy to recover the message. A threshold cryptosystem must be secure against adaptive chosen-ciphertext attacks [50], which means that the adversary cannot obtain any information from a ciphertext unless at least one honest server has generated a decryption share.

In a *threshold signature scheme*, each party holds a *share* of the secret signing key and may generate shares of signatures on individual messages upon request. The validity of a signature share can be verified for each party. From $t+1$ valid signature shares, one can generate a digital signature on the message that can later be verified using the single, publicly known signature verification key. In a secure threshold signature scheme, it must be infeasible for the adversary to produce $t+1$ valid signature shares that cannot be combined to a valid signature and to output a valid signature on a message for which *no*

honest party generated a signature share.

Another important cryptographic algorithm is *threshold coin-tossing scheme*, which provides a source of unpredictable random bits that can be queried only by a distributed protocol. It is the key to circumventing the FLP impossibility result [23] and used by the randomized Byzantine agreement protocol.

Threshold-cryptographic protocols have been used for secure service replication before, e.g., by Reiter and Birman [46]. However, a major complication for adopting threshold cryptography to our asynchronous distributed system is that many early protocols are not robust and that most protocols rely heavily on synchronous broadcast channels. Only very recently, non-interactive schemes have been developed that satisfy the appropriate notions of security, such as the threshold cryptosystem of Shoup and Gennaro [50] and the threshold signature scheme of Shoup [49]. Both have non-interactive variants that integrate well into our asynchronous model.

No Timing Assumptions. We do not make any timing assumptions and work in a completely asynchronous model. Asynchronous protocols are attractive because the alternative is to specify timeout values, which is very difficult when protecting against arbitrary failures that may be caused by a malicious attacker.

It is usually much easier for an intruder to block communication with a server than to subvert it. Prudent security engineering also gives the adversary full access to all specifications, including timeouts, and excludes only cryptographic keys from her view. Such an adversary may simply delay the communication with a server longer than the timeout and the server appears faulty to the others.

Time-based failure detectors [16] can easily be fooled into making an unlimited number of wrong failure suspicions about honest parties like this. The problem arises because one crucial assumption underlying the failure detector approach, namely that the communication system is stable for some longer periods when the failure detector is accurate, does not hold against a malicious adversary. A clever adversary may subvert a server and make it appear working properly until the moment at which it deviates from the protocol — but then it may be too late.

Of course, an asynchronous model cannot guarantee a bound on the overall response time of an application. But the asynchronous model can be seen as an elegant way to abstract from time-dependent peculiarities of an environment for proving an algorithm correct such that it satisfies liveness and safety under *all* timing conditions. By making no assumption about time at all, the coverage of the timing assumption appears much bigger, i.e., it has the potential to be justified in a wider range of real-world environments. For our applications, which focus on the security of trusted services, the resulting lack of timeliness seems tolerable.

Static Server Set. Distributing a trusted service among a static set of servers leverages the trust in the availability and integrity of each individual server to the whole system. In our model, this set remains fixed during the whole lifetime of the system, despite observable corruptions. The reason is that there

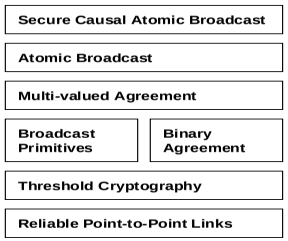


Figure 6. Asynchronous communications protocols for static groups.

are no protocols to replace corrupted servers in a secure distributed way, since all existing threshold-cryptographic protocols are based on fixed parameters (e.g., n and t) that must be known when the key shares are generated.

The alternative to a static server set is to remove apparently faulty servers from the system. This is the paradigm of view-based group communication systems in the crash-failure model [41]. They offer resilience against crash failures by eliminating non-responding servers from the current view and proceeding without them to the next view. Resurrected servers may join again in later views. But with the partial exception of Rampart [45], there is no group communication system that uses views and tolerates arbitrary failures (also Rampart cannot tolerate an attacker that has access to the failure detector).

The problem with Byzantine faults is that a corrupted server cannot be resurrected easily because the intruder may have seen all its cryptographic secrets. With the use of specialized "proactive" protocols [12], one could in principle achieve this by refreshing all key shares periodically. But such proactive cryptosystems for asynchronous networks have only recently been developed [8], after the formulation of this architecture, and further work would still be needed to build a fully asynchronous system with dynamic groups that tolerates Byzantine faults.

5.2 Secure Asynchronous Agreement and Broadcast Protocols

This section presents a short overview of the agreement and broadcast protocols used in this architecture configuration. Detailed descriptions can be found in related papers [10, 9, 11].

We need protocols for basic broadcasts (reliable and consistent broadcast), atomic broadcast, and secure causal atomic broadcast; they can be described and implemented in a modular way as follows, using multi-valued Byzantine agreement and randomized binary Byzantine agreement as primitives, as shown in Figure 6.

Byzantine agreement requires all parties to agree on a binary value that was proposed by an honest party. The protocol of Cachin et al. [10] follows the basic structure of Rabin's randomized protocol [42],

which is to check if the proposal value is unanimous and to adopt a random value, called a *common random coin*, if not. It terminates within an expected constant number of asynchronous rounds and uses a robust cryptographic threshold coin-tossing protocol, whose security is based on the so-called Diffie-Hellman problem. It requires a trusted dealer for setup, but can process an arbitrary number of independent agreements afterwards. Threshold signatures are further employed to decrease all messages to a constant size.

The other agreement primitive is *multi-valued Byzantine agreement*, which provides agreement on values from large domains. Multi-valued agreement requires a non-trivial extension of binary agreement. The difficulty in multi-valued Byzantine agreement is how to ensure the "validity" of the resulting value, which may come from a domain that has no a priori fixed size. Our approach to this is a new, "external" validity condition, using a global predicate with which every honest party can determine the validity of a proposed value. The protocol guarantees that the system may only decide for a value acceptable to honest parties. This rules out agreement protocols that decide on a value that no party proposed. Our implementation of multi-valued Byzantine agreement uses only a constant expected number of rounds; details can be found in [9].

A basic broadcast protocol in a distributed system with failures is *reliable broadcast*, which provides a way for a party to send a message to all other parties. Its specification requires that all honest parties deliver the same set of messages and that this set includes all messages broadcast by honest parties. However, it makes no assumptions if the sender of a message is corrupted and does not guarantee anything about the order in which messages are delivered. The reliable broadcast protocol of our architecture is an optimization of the elegant protocol by Bracha and Toueg [7]. We also use a variation of it, called *consistent broadcast*, which is advantageous in certain situations. It guarantees uniqueness of the delivered message, but relaxes the requirement that all honest parties actually deliver the message — a party may still learn about the existence of the message by other means and ask for it. Our implementation relies on non-interactive threshold signatures, which reduces the communication compared to previous implementations.

An *atomic broadcast* guarantees a total order on messages such that honest parties deliver all messages in the same order. Any implementation of atomic broadcast must implicitly reach agreement whether or not to deliver a message sent by a corrupted party and, intuitively, this is where Byzantine agreement is needed. The basic structure of our protocol follows the atomic broadcast protocol of Chandra and Toueg [16] for the crash-failure model: the parties proceed in global rounds and agree on a set of messages to deliver at the end of each round, using multi-valued agreement. For agreement every party proposes the messages that it wants to deliver in the current round. The agreement protocol decides on a list of messages and all messages in it are delivered according to a fixed order. The external validity condition ensures that all messages that are agreed-on list are appropriate for the current round. Details of this protocol are in [9].

A *secure causal atomic broadcast* is an atomic broadcast that also ensures a causal order among all

broadcast messages, as put forward by Reiter and Birman [46]. It can be implemented by combining an atomic broadcast protocol that tolerates a Byzantine adversary with a robust threshold cryptosystem. Encryption ensures that messages remain secret up to the moment at which they are guaranteed to be delivered. Thus, client requests to a trusted service using this broadcast remain confidential until they are scheduled and answered by the service. The threshold cryptosystem must be secure against adaptive chosen-ciphertext attacks to prevent the adversary from submitting any related message for delivery, which would violate causality in our context. Maintaining causality is crucial in the asynchronous environment for replicating services that involve confidential data. Our protocol for secure causal atomic broadcast follows the basic idea of Reiter and Birman's protocol. But because we use our atomic broadcast protocol and the non-interactive threshold cryptosystem of Shoup and Gennaro [50], we obtain the first provably secure implementation of secure causal atomic broadcast in an asynchronous network with Byzantine faults.

All our broadcast and agreement protocols work under the optimal assumption that $n > 3t$.

6 Reliable multicast: using trustworthy components

This section discusses MAFTIA architecture configurations following the strategy based on distributed trusted components (DTC). We solve a reliable multicast problem to exemplify the theory and the principles of construction of this kind of protocols.

The properties and correctness discussion of our protocols rely on the wormholes model, which postulates the existence of enhanced subsystems (wormholes) capable of providing a few simple privileged services to other components, with "good" properties otherwise not guaranteed by the "normal" weak environment in a distributed system [56]. For example, they can provide timely or secure functions in, respectively, asynchronous environments or systems with Byzantine failures. In MAFTIA the wormholes metaphor is materialized by the *Trusted Timely Computing Base (TTCB)* introduced in Section 3, a DTC providing a few timeliness- and security-related functions. Protocols built with a wormhole are run in a part of the system that might experience arbitrary delays or failures (asynchronous Byzantine environment). However, during their execution, they can call the wormhole services to perform correctly (small) crucial steps. In contrast to the rest of the system, the services only return trustworthy results. We say that such protocols are "wormhole-aware".

6.1 Model

Figure 7 shows a representation of the system in this configuration. Each node contains the typical software layers such as the operating system and runtime environments, and an extra component, the TTCB wormhole. The wormhole is distributed, with a local part in each node and a control channel. The local parts, or *local wormholes*, are computational components with activity, and the *control channel* is a private communication channel or network. The multicast protocol is executed by a group $P =$

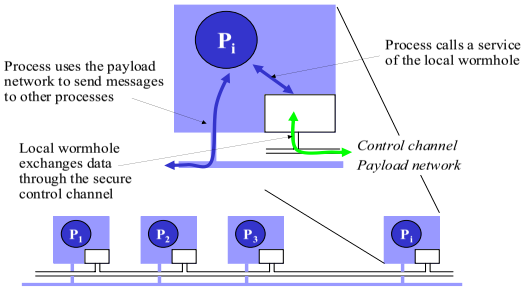


Figure 7. Architecture with a TTCB (payload subsystem is displayed in dark and the TTCB in white)

$\{p_1, ..., p_n\}$ of n processes. Processes run outside the wormhole and they communicate by sending messages through the payload network. At certain points of their execution, they can, however, request some services of the wormhole by calling its interface.

With the exception of the wormhole, the system is assumed to be asynchronous. Consequently, there can be no assumptions about the relative speed of processes, no bounds on message delivery delays (on the payload network), and no bounds on the invocations of wormhole services (since they are initiated and terminated in the asynchronous part of the system). The wormhole is assumed to be synchronous and capable of timely behavior. This means that once a request arrives at the wormhole interface, it will take a well defined interval until that answer is available at the same interface. In practical implementations these synchrony guarantees can be ensured because the wormhole has complete control over all resources in the node that are needed to perform its tasks, including the control channel (for details see [52, 14, 19]).

The payload part of the system, which includes the processes, can suffer from Byzantine faults. For instance, processes can stop working, skip some steps in the protocol, give invalid information to the wormhole or other processes, or collude with other malicious processes in an attempt to break the protocol. Byzantine faults can also affect the communication through the payload network and the service calls to the local wormhole. We assume that the payload network has an associated *omission degree* (Od), which implies that no more than Od messages are corrupted/lost in a reference interval of time. By making this assumption, a message only needs to be retransmitted $Od + 1$ times to ensure its reception in absence of attacks. In a Byzantine setting, however, for sufficiently strong adversaries one can envision attacks that will corrupt more than $Od + 1$ successive retransmissions. Whenever this happens, we take the approach of considering the receiver process as faulty (which is indeed the end result, since that process is unable to communicate). Nevertheless, the reader should notice that Od is just a parameter of the protocol: conservatively large values of Od basically simulate a reliable channel. Incidentally, if a process is systematically prevented from calling the local wormhole, then it will also be considered

as faulty.

The wormhole subsystem, which includes the control network, is assumed to fail only by crashing. Therefore, a local wormhole either provides its services as expected or it simply stops running. This assumption should hold even if malicious adversaries manage to attack and compromise a node with a local wormhole (for implementation details see [19]).

6.2 Wormhole services and interface

A wormhole provides a small number of services that can be accessed through calls to its local interface. Distinct protocols can utilize different services in different ways. However, in all cases, protocols are designed to run on the payload subsystem and use the wormhole infrequently, imposing a comparatively small load on it. The TTCB (Trusted Timely Computing Base), is just one example of wormhole, the one implemented in MAFTIA, whose most important services are briefly described ahead (a detailed description can be found in [35]).

The *Local Authentication service* (this is a component-oriented, rather than a high-level authentication service) makes the necessary initializations and authenticates the local wormhole component before the process. A timestamp with the current global time is returned by the *Timestamping service*. The *Trusted Block Agreement service* applies an agreement function to a set of values proposed by the processes and returns a value. By using different functions, this service can be configured to deliver results with diverse characteristics. Perhaps a good way to understand the service is through the description of its interface parameters:

tag,error ←TTCB_propose(eid, elist, tstart, decision, value)

value,prop-ok,prop-any,error ←TTCB_decide(eid, tag)

A process calls $TTCB\_propose$ to propose a value. The parameters have the meaning: eid is the unique identification of a process before the wormhole, obtained using the Local Authentication Service. $elist$ is a list with the $eids$ of the processes involved in the agreement. $tstart$ is a timestamp and corresponds to the latest instant when the agreement will start (it might be initiated earlier if all proposals arrive before $tstart$). A proposal made after $tstart$ is rejected and an error is returned. *decision* defines the agreement function (the TTCB offers a limited set). $value$ is the proposed value, and it can only have a small number of bytes (20 in the current implementation). The function returns an $error$ code and a unique identifier of this agreement, called the tag. Processes call *TTCB_decide* to get the result of the agreement. The result is a record with four fields: the decided *value*, a mask *prop-ok* with one bit set per each process that proposed the decided value, a mask *prop-any* with one bit set per process that proposed any value, and an *error* code.

6.3 Designing wormhole-aware protocols

We use a reliable multicast protocol as an example to illustrate the principles of building wormhole-aware protocols (more details can be found in [18]). A reliable multicast protocol guarantees that all correct processes deliver the same messages, and if a correct sender transmits a message then all correct processes deliver this message. No assurances, however, are provided about the order of message delivery. Formally, a reliable multicast protocol has the following properties [26] [4].

- *Validity:* If a correct process multicasts a message M, then some correct process in $group(M)$ eventually delivers M.
- *Agreement:* If a correct process delivers a message M, then all correct processes in $group(M)$ eventually deliver M.
- *Integrity:* For any message M, every correct process p delivers M at most once and only if p is in $group(M)$, and if $sender(M)$ is correct then M was previously multicast by $sender(M)$.

One achievement related to our model is that the protocol requires that, out of a total of n processes, no more than $f = n - 2$ processes are allowed to fail (the problem is of obviously little interest with less than two correct processes). The asynchronous Byzantine system model augmented with the TTCB wormhole allows us to lower the known limit of $f = (n - 1)/3$ processes [18].

Basic principles. Wormhole-aware protocols can be depicted as running on a dual space-time diagram (see Figure 8): the payload subsystem's, seen in evidence in the figure; and the TTCB subsystem's, whose timelines are collapsed in the thick gray bar for simplicity. A correct use of the wormholes principle mandates that most of the protocol execution takes place in the payload subsystem. The wormhole services are only invoked when there is an obvious trade-off between what is obtained from the wormhole service and the complexity/cost of implementing it in the payload subsystem (e.g., related with hard reliability, synchrony, or security requirements). This will become evident from the examples to come.

As usual, the protocols starts with the sender multicasting a message through the payload channel. Since the sender might be malicious or the network might be attacked or have omission failures, the protocol needs to ensure the reception of the message by all correct processes and the integrity of the message contents.

In asynchronous Byzantine environments this may entail some complexity and/or delay [44, 28]. The wormhole has thus an opportunity to come into play: the sender and all recipients send a hash of the message just sent/received, to the wormhole, which runs an agreement on the hashes in its protected environment, returning to all the sender's hash as result. If all goes well, processes see that: all proposed

[4] The predicate $sender(M)$ gives the message field with the sender, and $group(M)$ gives the "group" of processes involved, i.e., the sender and the recipients (note that we consider that the sender also delivers).

the same hash; and that the agreed hash corresponds to the message received. Thus, they terminate at the end of this phase, in an extremely fast manner.

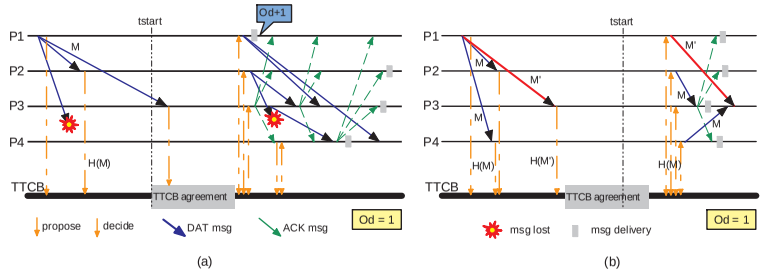


Figure 8. Reliable Multicast with a Wormhole: (a) omissions and delays; (b) malicious sender.

Detecting and recovering from errors. If this condition is not verified, then either one or more processes did not get the correct message or some of them were late when proposing to the wormhole (after $tstart$). Such a situation is depicted in Figure 8a, and it requires a second phase: P_3 is going to propose late and thus be rejected; P_4 suffers an omission. Detecting these errors is another difficult task in asynchronous Byzantine environments, and here we see another contribution of the wormhole: reliable error detection (performed concurrently with the agreement on the hash). Processes detect the errors immediately, because they spot "holes" in the *prop-ok* and *prop-any* masks (see Section 6.2), respectively for processes that proposed wrongly or did not propose at all.

In the second phase, processes try to remedy for the above-mentioned "holes", using classical retransmission and forwarding techniques. As shown in Figure 8a, processes retransmit the message until either all acknowledge the arrival of the message or the $Od + 1$ threshold is reached. The 'bounded omission degree' technique for synchronous environments [54] can be used here for termination in an asynchronous environment, due to the synchronous distributed error detection channel provided by the TTCB wormhole.

Each multicast is retransmitted at most $Od + 1$ times in order to resist a number of network failures ($Od = 1$ in the setting of the example). After a few forwardings and acknowledgements, we see that all processes end-up converging and deliver the message. In terms of protocol principle, the reader should see these operations as recovery measures that end-up asserting the missing bits in the masks. Figure 8b depicts the more serious situation where the sender is Byzantine: it sends a different message M' to P_3, who obviously proposes hash $H(M')$ and is deemed wrong. The remedy is used again: correct recipients forward M to P_3, who asserts its bit in its own *prop-ok* mask and acknowledges. The other recipients do the same in reaction to the acknowledgement, and all deliver.

Payload-wormhole synchronization. The role of $tstart$ deserves mention, since it can be generalized to other protocols as well. $tstart$ reveals an interesting way of performing synchronization between the asynchronous payload and the synchronous TTCB wormhole protocols.

To the latter, such a quantity is a real time instant or interval in the global timeline maintained by all wormhole modules. It allows wormhole protocols to perform essentially all timed operations. To the former, the time-free payload protocols, it is merely an integer quantity. These 'synchronizers' can be used as constants or variables, and establish a sound basis for decoupling the logic and timing aspects of the design of protocols [51]. Time-free modules can safely synchronize with each other by exchanging and/or agreeing on these values, adding quantities to them, and so forth, since all timing aspects (including timing failure, a.k.a timeout, detection) are dealt with in the wormhole. These values can be bound to real time when desired, by using the Timestamping service [53].

In this particular case, $tstart$ is set to the current time plus a delay. To prevent the service execution from being maliciously postponed, the Trusted Block Agreement service rejects proposals that arrive after $tstart$. Therefore, the selection of the $tstart$ value presents a tradeoff: if it is too large, the first phase may take longer than what is required; if the value is too small, a correct recipient may not receive the message before $tstart$ and the second phase will have to be executed unnecessarily (i.e., the opportunity to terminate the protocol early is lost). Note however that the value of $tstart$ only affects performance, and not safety or liveness. A good heuristics for selecting the delay is to make it proportional to an estimate of the message transmission time. Incidentally, in another work we have studied the dependable dynamic adaptation of timing parameters such as message transmission times, in systems using wormholes [15].

Acknowledgement protection. The acknowledgements, as the data messages, have to be protected from forgeries. To accomplish this task, we use a vector of Message Authentication Codes (MACs) through the payload, instead of relying on the services of the wormhole. This is a good example of a situation where the trade-off for using the wormhole does not occur: what is going to be used potentially massively (acknowledgements), and could be easily and efficiently achieved through the payload (MACs), should not go through the wormhole.

A MAC is a cryptographic checksum obtained with a hash function and a symmetric key [32]. Although MACs are not as powerful as signatures based on public-key cryptography, they are sufficient for our needs, and more importantly, they are several orders of magnitude faster to calculate. Since acknowledgements are multicasted to all processes, they do not take a single MAC but a vector of MACs with an entry per process.

7 Authorization

Authorization aims to ensure that only legitimate actions are carried out in the system, or, equivalently, to prevent illegitimate actions from being carried out. As such, authorization, and its implementation by access control mechanisms, participates in error detection (by detecting attempts to run illegitimate actions) and in error confinement (by preventing illegitimate actions), whether these errors are due to accidental faults or attacks. Authorization is thus of tremendous importance for Internet applications.

Currently, the most common authorization scheme used on the Internet is based on the client-server paradigm: a server satisfies or rejects client requests at its discretion, according to what it knows about the client (e.g., the identity claimed by the client, history of previous transactions, etc.). Unfortunately, the client-server model is not rich enough to cope with complex transactions involving more than two participants. For example, an electronic commerce transaction requires usually the cooperation of a customer, a merchant, a credit card company, a bank, a delivery company, etc. Each of these participants has different interests, and thus distrusts the other participants. Moreover, such a model is necessarily privacy intrusive, since it enables the server to record a lot of personal information about clients: identity, usual IP address, postal address, credit card number, purchase habits, etc.

MAFTIA proposes a new authorization scheme that can grant fair rights to each participant, while distributing to each one only the information strictly needed to execute its own task, i.e., a proof that the task has to be executed and the parameters needed for this execution, without unnecessary information such as participant identities. This scheme is based on two levels of access control:

An *authorization server* is in charge of granting or denying rights for transactions involving several participants; if a transaction is authorized, the authorization server distributes authorization proofs (i.e., capabilities) for all the elementary operations that are needed to carry it out.

On each participating host, a *reference monitor* is responsible for fine-grain authorization, i.e., for controlling the access to all local resources and objects according to the capabilities that accompany each request. To enforce hack-proofing of such reference monitors on off-the-shelf computers connected to the Internet, critical parts of the reference monitor are based on a MAFTIA trusted component, the Java Card Module, as described in Section 3.

7.1 Authorization Server

In [37], a generic authorization scheme had been proposed for distributed object systems. In this scheme, an application can be viewed at two levels of abstraction: composite operations and method executions. A composite operation corresponds to the coordinated execution of several object methods towards a common goal. For instance, printing file F3 on printer P4 is a composite operation involving the execution of a *printfile* method of the spooler object attached to P4, which itself has to request the execution of the *readfile* method of the file server object managing F3, etc. In the MAFTIA context, composite operations can be assimilated to transactions.

A request to run a transaction is authorized or denied by an authorization server, according to *symbolic rights* stored in an access control matrix managed by the authorization server. More details on how the authorization server checks if a transaction is to be granted or denied are given in [36] and [2]. If the request is authorized, capabilities are created by the authorization server for all the method executions needed to perform the transaction. These capabilities are simple method capabilities if they are used directly by the object requesting the execution of the transaction, i.e., used by this object to directly call another object's methods. Alternatively, the capabilities may be indirect capabilities or *vouchers*, if they cannot be used by the calling object but must be delegated to another object that will invoke other object methods to participate in the transaction. In fact, the notion of transaction is recursive, and a voucher can contain either a method capability or the right to execute a nested transaction.

This delegation scheme is more flexible than the usual "*proxy*" scheme by which an object transmits to another object some of its access rights for this delegated object to execute operations on behalf of the delegating object. Our scheme is also closer to the "*least privilege principle*", since it helps to reduce the privilege needed to perform delegated operations. For instance, if an object O is authorized to print a file, it has to delegate a *read-right* to the spooler object, for the latter to be able to read the file to be printed. To delegate this read-right with the proxy scheme, O must possess this read-right and could thus abuse this right by making copies of the file and distributing them. In this case, the read-right is a privilege much higher than a simple print-right. In our scheme, if O is authorized to print a file, O will receive a *voucher* for the spooler to read the file, and a capability to call the spooler. The voucher, by itself, cannot be used by O. With the capability, O can invoke the spooler and transmit the voucher to the spooler. The spooler can then use the voucher as a capability to read the file.

Since only transactions are managed by the authorization server, system security is relatively easy to manage: the users and the security administrators have just to assign the rights to execute predefined transactions, they do not have to consider all the elementary rights to invoke object methods. Moreover, since only one request has to be checked for each transaction, the communication overhead can be reduced.

The authorization server is a trusted third party (TTP), which could be a single point of failure, in case of both accidental failure or successful intrusion (including by a malicious administrator). To prevent this, with the MAFTIA authorization architecture, the authorization server is made fault- and intrusion-tolerant [2]: an authorization server is made of diversely-designed and implemented security sites, operated by independent persons. Faults and intrusions affecting security sites can be tolerated without degrading the service, as long as only a few security sites are affected.

In order to tolerate the failure of one or a small number of the sites composing the authorization server, two main protocols are used:

Mutual agreement: all non-faulty sites agree on the decision to grant or deny the authorization corresponding to a given request. This guarantees a correct decision as long as there is only a minority of faulty sites. In practice, the number f of faulty sites may have to be much less than half the total number

n of sites, depending on the fault assumptions. For instance, $(n \text{ ¿ } 3f)$ must be guaranteed if Byzantine faults are to be taken into account.

Threshold signature: the capabilities and vouchers are globally signed by the authorization server, using a threshold signature scheme. Each of the sites composing the authorization server generates a signature share (depending on its own private key share) so that if at least t valid signature shares are available (t being the threshold), it is possible to combine these shares to generate a unique signature that can be verified with a global public key. This guarantees that if a capability (or a voucher) has a correct signature, the corresponding operation is indeed authorized (the signed capability cannot be forged, even by a cooperation of f faulty sites, as long as f is strictly less than the threshold t).

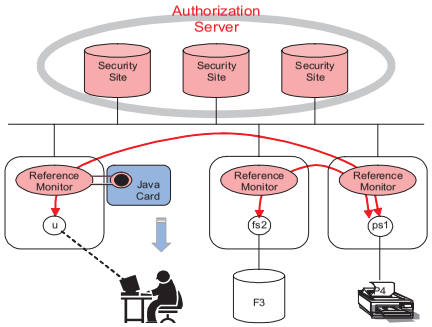


Figure 9. Authorization architecture.

These protocols are presented with more detail in [3]. The global architecture is given by Figure 9. The dialogue between a MAFTIA object and the authorization server is typically as follows (*see* Figure 10).

Object O asks the authorization server for the authorization to carry out an operation in the system. This operation may be the simple invocation of a particular method of a particular object O' or may be a transaction that requires the collaboration between several objects in the system.

In the first case, if object O is authorized to carry out the operation, it receives a capability, encrypted by the public key of the host where O' is located, and then signed using the threshold scheme described above. This capability will be presented and checked by the reference monitor located on the site of the invoked object O'.

In the second case, the user may receive several capabilities and vouchers. Capabilities are directly used by object O to invoke particular methods of particular objects, and are encrypted and signed as in the first case. Vouchers are not used by object O but are forwarded by object O to other objects that are involved in the execution of the transaction (e.g., a capability for O' to invoke a method m of an

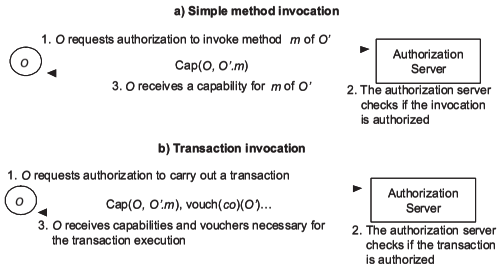


Figure 10. Protocol between a MAFTIA object and the authorization server.

object O'', as a part of the transaction). These vouchers will thus be transferred by object O to other objects, which will then execute their part of the transaction thanks to these vouchers. A voucher may be a capability (in which case they are encrypted and signed as above), or the right to execute a nested transaction (in which case the voucher is just signed). In the latter case, this voucher is a *token* that has to be presented to the authorization server to obtain directly all the authorization proofs needed to execute the nested transaction, without consultation of the access control matrix.

7.2 Reference Monitor

There is a reference monitor on each host participating in a MAFTIA-compliant application. The reference monitor is responsible for granting or denying local object method invocations, according to capabilities and vouchers distributed by the authorization server. In the context of wide-area networks (such as the Internet), the implementation of such a reference monitor is complicated since, due to the heterogeneity of connected hosts, it would be necessary to develop one version of the reference monitor for each kind of host. Moreover, since the hosts are not under the control of a global authority, there is no way to ensure that each host is running a genuine reference monitor, or the same version thereof. This is one reason we have chosen to implement them by using Java Cards.

We assume that any software, even that within an operating system or a JVM, can be copied and modified by a malicious user who possesses all privileges on a local host. In particular, on the Internet, any hacker can easily have these privileges on his own computer! The capability checks only provide assurance to *non-faulty* hosts, who can be sure that any remote request to execute a MAFTIA-application is genuine (if the capability is correct), and that a genuine MAFTIA request can only be executed on a host for which the capability is valid.

However, if a host is faulty or has been corrupted by a hacker, there is no assurance at all about the operations it executes locally. Nevertheless, we do trust the local Java Card Module to protect the

integrity of a corrupt host's private key. This means in particular that we exclude the possibility of a corrupt host being able to impersonate a non-faulty host since it is unable to sign messages correctly. Thus, the privileges gained by a hacker on a corrupt host give him no privilege outside that host unless the hacker is able to tamper with the local Java Card Module.

We consider the Java Card Module to be *sufficiently* tamperproof, as discussed in Section 3, in order to sufficiently delay the attacker's progress in corrupting further hosts. Global properties can thus be maintained by fault-tolerance mechanisms unless, say, more than f hosts are compromised in a given "window of vulnerability"[59]. The fact that the authorization scheme prevents a hacker from gaining privileges outside a corrupted host unless he successfully tampers with the Java Card Module means that the difficulty of violating such a global property is linear in f.

The capability checks carried out by the Java Card Module are based on strong cryptographic functions. Several cryptographic keys must be included in the Java Card:

PK_m, the MAFTIA public key. This key is associated to the MAFTIA private key SK_m, which is not stored in the Java Card. The Java object classes are signed off-line by this key SK_m, and this signature is checked at load time by the local JVM of the host[5], using PK_m stored in the Java Card.

SK_j, PK_j, a private/public key pair specific to the Java Card, thus specific to the host.

PK_{as}, the authorization server public key. This key is associated to all the private key shares of all the sites composing the authorization server.

Each capability is encrypted by the authorization server, using the public key PK_j of the site where the invoked object is located. Then the capability is signed by the authorization server with a threshold signature protocol. Consequently, the capability signature must first be verified using the authorization server's public key PK_{as}, and then decrypted (by the cryptographic functions of the Java Card) using the private key SK_j, which is stored only in the Java Card. Each access to a method of an object on a MAFTIA host is first intercepted by a *Dispatcher*, which is a Java object, located in the local JVM interfacing the Java Card. For each access to a local object method, the dispatcher checks if the invocation is carrying a capability, then sends this capability to the Java Card for verification. This verification corresponds to step 2 of Figure 11.

Other information can be stored in the Java Card, for instance for the authentication of the user owning the Java Card if the host is a personal workstation, or for the authentication of the administrator who has been assigned this Java Card if the host is a server. In the latter case, it may be possible to have several administrators for the same server, each administrator having his personal Java Card for this server, and all server administrator Java Cards sharing the same pair SK_j, PK_j. More details on the Java Card implementation of the reference monitor can be found in [21].

[5] Since version 1.2, the Java Development Kit includes software that allows classes to be signed and the signatures to be checked at load time.

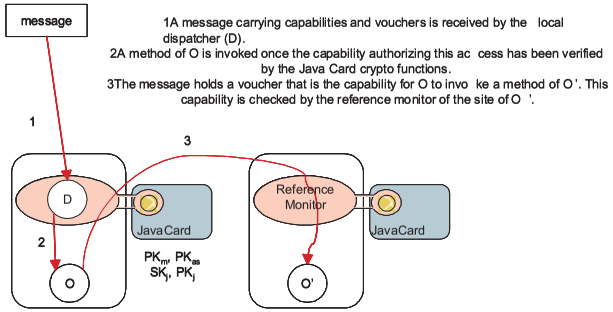


Figure 11. Example of a voucher corresponding to a capability.

8 Transactions: providing IT to applications

This section describes the MAFTIA transactional support service. The transactional support service is intended to support both applications built using the MAFTIA middleware and other activity support services, for example it can be used to guarantee the atomicity of updates to a replicated authorization server. The transaction support service appears to the user as a CORBA-style transaction service. Its intrusion tolerance is a non-functional property of the service implementation, transparent to the interface.

8.1 Overview

The MAFTIA transaction service provides a mechanism for implementing application-level intrusion tolerance and is itself intrusion-tolerant. To do this we apply the general MAFTIA architectural principle of distributing trust by replicating the servers implementing the transaction service and optionally the resource managers.

Our approach is to make use of standard group communication primitives, allow for heterogeneous resources, apply error compensation techniques to improve intrusion tolerance, to allow for multi-party transactions consider failure atomicity and allow recovery without reliance upon durable storage. This differentiates our work from approaches that make use of new or modified group communication primitives (for example, optimistic broadcast) [25, 47, 57]. Also, unlike other approaches, our focus is not on availability but on intrusion tolerance. This has resulted in us not being able to use techniques such as passive replication that are widely used by the database community. Passive replication is more efficient than active replication, and does not require deterministic replicas. However, the problem with adopting passive replication is its reliance on a leader-follower model. The updates occur at the leader and the followers are informed of the results. Whereas this is adequate in a crash-failure fault model, it is inappropriate for malicious faults, because the leader becomes a single point of failure: the leader can be corrupted and start sending corrupted updates to the leaders. By adopting an active replication approach we avoid this problem as there is no single point of failure and more that f members of the group must be corrupted before the group as a whole is compromised.

The approach that is closest to our is that of *GroupTransactions* [39] although our model of multi-party clients is different: our multiple parties are pre-existing and are not created within the context of a group transaction. Our model is also more general, since their system model assumes a LAN and does not explicitly consider malicious faults. However, they do address nested transactions whereas we only implement a flat transaction model.

8.2 Architecture

An overview of the transaction service architecture is shown in Figure 12. We have implemented the transaction service using the Appia framework [34]. Each major component and constituent Appia protocol layers are shown. As in a traditional distributed transaction service, the architecture is made up of clients, resource managers and transaction managers. Multiple clients interact with replicated transaction and resource managers. Our architecture differs from a traditional architecture in several ways: the transaction manager and resource manager components are replicated; each component uses intrusion-tolerant group communication protocols to provide intrusion tolerance; managers avoid the need for durable storage for recovery; and our service supports multi-party transactions.

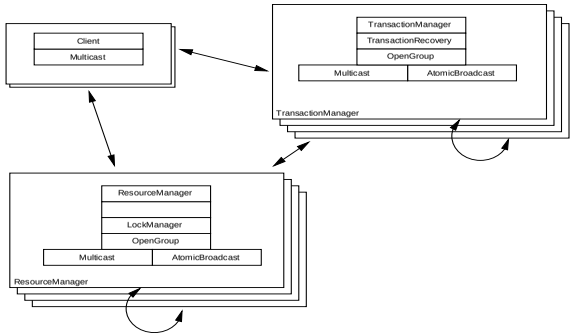


Figure 12. Overview of MAFTIA Transaction Service

Clients use the transaction manager to begin and end transactions and within the scope of a transaction the clients operate on resources via resource managers. There may be multiple clients participating in the same transaction or multiple clients participating in different transactions. The transaction manager

is primarily a protocol engine. It implements the two-phase commit protocol and recovery protocol. It also allows the creation of new transactions and the marking of transaction boundaries. In order to participate in transactions, resource managers are required to register themselves with the transaction manager. A resource manager is a wrapper for resources that allows the resource to participate in two-phase commit and recovery protocols coordinated by a transaction manager. The resource may or may not be persistent. In our implementation concurrency control is pessimistic but an optimistic scheme could also be implemented with minimal change to the interfaces of the resource manager.

Although appearing to clients as standard transaction and resource managers, the components of the MAFTIA transaction service are replicated. When a client makes a request it is processed by all the replicas and the client is delivered the result returned by the majority of replicas. To prevent an adversary manipulating the order of delivery of requests and therefore the outcome, we rely upon the MAFTIA intrusion-tolerant group communication service.

Assuming that failure can be reliably detected then a recovery mechanism is required to allow the resumption of transaction processing. Unlike in a traditional distributed transaction service, the local durable log cannot be used to recover because it may have been compromised. So, in the MAFTIA transaction service the recovery for transaction managers and resource managers relies upon a recovering replica querying the group to determine the state of the log.

We support multi-party transactions with a clear semantics on how clients share the decision for aborting or committing a transaction. In our model a single client begins a transaction, and passes the transaction identifier to other clients so that they can cooperate within the transaction scope too. Individual clients can unilaterally force a transaction abort but all clients must unanimously agree to attempt a transaction commit. These semantics are based upon those of Coordinated Atomic Actions (CAA) [58, 43] where participants must either agree on a normal or exceptional outcome, or abort the entire action. The general notion is to provide clear semantics for the termination of join action but to allow parties to share resources freely within the context of a transaction. Note that unlike CAA our model of multi-party transactions does not currently consider agreement upon exceptions.

8.3 Reliance on MAFTIA middleware

The MAFTIA transaction service both supports application-level intrusion tolerance by providing error confinement for multi-party interactions and is itself intrusion-tolerant because it provides correct service in the presence of compromised transaction and resource managers. Correct service is achieved through the application of the principle of error compensation that is implemented using active or "state machine" replication [40]. The transaction service is composed of replicated and diverse resource manager and transaction manager servers. We rely upon the MAFTIA middleware's communication services to implement the replication. Therefore, in order for the transaction service to tolerate intrusions, we need the communication services to be intrusion tolerant.

Two different strategies can be used to make the communication services intrusion tolerant. The "fail-uncontrolled" strategy can be used to provide fault-tolerant atomic broadcast for systems where Byzantine behavior by users is possible and we cannot make timing assumptions. The fault-tolerance provided by this strategy depends upon the use of time-free probabilistic Byzantine protocols. The "fail-controlled with distributed trusted components" strategy can be used to provide fault-tolerant atomic broadcast where a TTCB is present. The tamper-proof construction of the local TTCB and the control channel prevents the host engaging in Byzantine behavior or being vulnerable to timing attacks.

As discussed above, we replicate transaction managers and resource managers. These form server groups that are distributed across sites. Server groups are a set of n servers, of which up to f may fail in completely arbitrary ways. All members of the service group handle requests and the majority result is returned to the user of the service. This means that as long as no more than f servers fail, the overall service remains trustworthy. To allow voting on results the servers are assumed to be deterministic. In the arbitrary model, $3f + 1$ servers are sufficient to provide correct service in the face of f expected failures. In the TTCB model, depending upon the implementation of atomic broadcast that is used then as few as $2f+1$ servers may be sufficient. Note that it is assumed that the groups have static membership and sufficient diversity of implementation, platform etc. to give assurance that the servers do not share a common failure mode.

Recovery in our implementation does not depend upon local durable storage as an attacker may compromise this. Instead, a recovering group member will contact other group members to determine what its state should be. Such an approach assumes that there is some mechanism that ensures that a recovering member can be successfully reinitialized without compromising the security of the group. For example, when using the asynchronous timing assumptions then recovery may require the trusted dealer to redistribute keys so that the security of the group is maintained.

Interacting with the transaction managers and resource managers are an unknown number of possibly faulty clients. Clients are outside our control and can be implemented in any way. Therefore they can fail in arbitrary ways. Currently we do not make clients intrusion-tolerant or the transaction service tolerant of misbehaving clients. For example, clients may block the progress of transactions or access to resources managed by resource managers. We have avoided using timeouts to resolve these problems as they introduce a vulnerability that could be exploited by an attacker.

8.4 Implementation

As the Appia framework was the standard implementation framework for the MAFTIA middleware then this simplified integration with the MAFTIA communication services. Because both the arbitrary (*cf.* Section 5) and TTCB-based (*cf.* Section 6) communication services were implemented using the same framework, the transaction service can run under either configuration. Integration with other frameworks would simply require the substitution of the AtomicBroadcast layer.

As future work, the existing transaction service could be integrated with the authorization service. This would enhance intrusion-tolerance, for example, for example participation in transactions could be restricted to trusted clients. Enhancing our current use of a transaction identifier by treating it as a capability could do this. In this model, capabilities for participating in transactions and accessing resources would be issued by the distributed authorization server, and checked by the transaction and resource managers.

On another tone, the transaction service could use the TTCB functionality directly, as now done by the communication protocols. For example, the TTCB provides a basic service that allows for consensus on a limited amount of state. Invoking this service directly instead of via the Atomic Broadcast protocols could provide considerably better performance, at the cost of tying the transaction service implementation to a particular intrusion tolerance strategy.

9 Conclusion

We presented the approach taken in MAFTIA to architect and build intrusion-tolerant systems, i.e., systems that are assumed to remain to some extent faulty and/or vulnerable and subject to attacks that can be successful, the idea being to ensure that the overall system nevertheless remains secure and operational, using notions pertaining to the generic 'tolerance' paradigm.

We started by revising the basic dependability concepts under a security-related perspective, incorporating specific security properties, fault classifications, and security methods. Under the light of these revised concepts, the term *trustworthiness*, often preferred when the focus is on malicious activity, is essentially synonymous to dependability, and has a powerful and precise meaning: it points to generic properties and not just security; it has a well-defined relationship with the notion of *trust*. This relation supports an important design principle in MAFTIA: a *trusted component* has a set of properties that are trusted *to the extent of* the component's trustworthiness.

In the course of developing the MAFTIA architecture and intrusion-tolerant middleware, we were faced with a multitude of challenges that we shared with the reader, since they are common to any endeavor in distributed, malicious-fault tolerant architectures. As a result, we devised new architectural constructs and algorithmic strategies. We introduced architectural hybridization as a means to substantiate the notion of 'trustworthy trusted component' in a malicious-fault environment. We devised programming models based on the modular use of trusted components, taking advantage of their fault prevention potential to recursively assist and augment the power of fault-tolerance mechanisms. We developed protocols that reason in terms of the availability of such trusted components, to achieve efficient operation whilst preserving resilience. On the alternative track of fail-arbitrary asynchronous environments, satisfying safety under any conditions for highly critical security uses, we devised provably secure protocols employing efficient cryptographic techniques with randomization. We proposed a new authorization scheme that overcomes privilege amplification or privacy violation problems in multi-

participant transactions, based on innovative access control protocols. Finally, we introduced replication and transaction control mechanisms built on top of the mentioned protocols, in an illustration of the recursive, component-based overall strategy for intrusion tolerance in MAFTIA explained in the paper. The rationale behind these protocols, whose algorithmics is detailed in other publications, was presented as a proof of concept of the several strategies to achieve intrusion tolerance in and with the MAFTIA architecture and middleware.

Finally, the notion of handling a wide set of faults encompassing intentional and malicious faults in order to preserve system properties (security or other), if successfully achieved, as we hope to have demonstrated, has two striking effects: (a) it leads us to *x-tolerant system frameworks*, common system design principles where "any fault set" can be handled, instead of (as presently) changing framework depending on the fault model and application; (b) it presents a great advance on the ability to design accidental fault-tolerant systems in complex and unpredictable settings (presently a research subject).

Acknowledgements

MAFTIA was a project of the IST programme of the European Commission, whose participants were: FCUL University of Lisboa, IBM Zurich Research Labs, LAAS-CNRS, Qinetiq, University of Newcastle upon Tyne, Universitt des Saarlandes. We wish to warmly thank all the members of the project teams, whose contributions were invaluable for the collective success of the project. The achievements of MAFTIA are not limited to architectural work. The MAFTIA portal [6], where the reader can find all publications of the project, details other very successful work strands, not only the formal verification of some of the protocols discussed here, but also, for example, application work on intrusion detection.

References

[1] J.-C. Laprie A. Avizienis and B. Randell. Fundamental concepts of dependability. In *Proceedings of the IEEE Third Information Survivability Workshop*, Boston, MA, USA, October 2000. IEEE CS Press.

[2] N. Abghour, Y. Deswarte, V. Nicomette, and D. Powell. Specification of authorisation services. Maftia project ist 1999-11583, deliverable d27, January 2001.

[3] N. Abghour, Y. Deswarte, V. Nicomette, and D. Powell. Design of the local reference monitor. Maftia project ist 1999-11583, deliverable d6, April 2002.

[4] A. Adelsbach, D. Alessandri, C. Cachin, S. Creese, Y. Deswarte, K. Kursawe, J. C. Laprie, D. Powell, B. Randell, J. Riordan, P. Ryan, W. Simmonds, R. Stroud, P. Veríssimo, M. Waidner, and A. Wespi. *Conceptual Model and Architecture of MAFTIA. Project MAFTIA deliverable D21.* January 2002. http://www.research.ec.org/maftia/deliverables/D21.pdf.

[5] A. Adelsbach, D. Alessandri, C. Cachin, S. Creese, Y. Deswarte, K. Kursawe, J. C. Laprie, D. Powell, B. Randell, J. Riordan, P. Ryan, W. Simmonds, R. Stroud, P. Veríssimo, M. Waidner, and A. Wespi. *Conceptual Model and Architecture of MAFTIA. Project MAFTIA IST-1999-11583 deliverable D21.* January 2003. http://www.research.ec.org/maftia/deliverables/D21.pdf.

[6] MAFTIA portal: http://http://www.newcastle.research.ec.org/maftia.

[6] A. Avizienis, J.-C. Laprie, and B. Randell. Fundamental concepts of dependability. Research Report 01145 (Revision 1: December 2002), LAAS-CNRS, August 2001. UCLA CSD Report no. 010028; Newcastle University Report no. CS-TR-739.

[7] Gabriel Bracha and Sam Toueg. Asynchronous consensus and broadcast protocols. *Journal of the ACM*, 32(4):824–840, October 1985.

[8] Christian Cachin, Klaus Kursawe, Anna Lysyanskaya, and Reto Strobl. Asynchronous verifiable secret sharing and proactive cryptosystems. In *Proc. 9th ACM Conference on Computer and Communications Security (CCS)*, pages 88–97, 2002.

[9] Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. Secure and efficient asynchronous broadcast protocols (extended abstract). In Joe Kilian, editor, *Advances in Cryptology: CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 524–541. Springer, 2001.

[10] Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in Constantinople: Practical asynchronous Byzantine agreement using cryptography. In *Proc. 19th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 123–132, 2000.

[11] Christian Cachin and Jonathan A. Poritz. Secure intrusion-tolerant replication on the Internet. In *Proc. Intl. Conference on Dependable Systems and Networks (DSN-2002)*, pages 167–176, June 2002.

[12] Ran Canetti, Rosario Gennaro, Amir Herzberg, and Dalit Naor. Proactive security: Long-term protection against break-ins. *RSA Laboratories' CryptoBytes*, 3(1), 1997.

[13] Ran Canetti and Tal Rabin. Fast asynchronous Byzantine agreement with optimal resilience. In *Proc. 25th Annual ACM Symposium on Theory of Computing (STOC)*, pages 42–51, 1993. Updated version available from `http://www.research.ibm.com/security/`.

[14] A. Casimiro, P. Martins, and P. Veríssimo. How to build a Timely Computing Base using Real-Time Linux. In *Proceedings of the IEEE International Workshop on Factory Communication Systems*, pages 127–134, September 2000.

[15] A. Casimiro and P. Veríssimo. Using the Timely Computing Base for dependable QoS adaptation. In *Proceedings of the 20th IEEE Symposium on Reliable Distributed Systems*, October 2001.

[16] Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, 1996.

[17] CIDF. Common intrusion detection framework, 1999.

[18] M. Correia, L. C. Lung, N. F. Neves, and P. Veríssimo. Efficient Byzantine-resilient reliable multicast on a hybrid failure model. In *Proceedings of the 21st IEEE Symposium on Reliable Distributed Systems*, pages 2–11, October 2002.

[19] M. Correia, P. Veríssimo, and N. F. Neves. The design of a COTS real-time distributed security kernel. In *Proceedings of the Fourth European Dependable Computing Conference*, pages 234–252, October 2002.

[20] H. Debar, M. Dacier, and A. Wespi. Towards a taxonomy of intrusion-detection systems. *Computer Networks*, 31(8):805–822, 1999.

[21] Y. Deswarte, N. Abghour, V. Nicomette, and D. Powell. An internet authorization scheme using smart card-based security kernels. In I.Attali and T.Jensen, editors, *International Conference on Research in Smart Cards, e-Smart 2001*, Lecture Notes in Computer Science LNCS 2140, pages 71–82, Cannes, France, 2001. Springer-Verlag.

[22] J.E. Dobson and B. Randell. Building reliable secure systems out of unreliable insecure components. In *Conf. on Security and Privacy*, pages 187–193, Oakland, CA, USA, 1986. IEEE CS Press.

[23] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, April 1985.

[24] J. Fraga and D. Powell. A fault and intrusion-tolerant file system. In J.B. Grimson and H.-J. Kugler, editors, *IFIP 3rd Int. Conf. on Computer Security*, Computer Security, pages 203–218, Dublin, Ireland, 1985. Elsevier Science Publishers B.V. (North-Holland).

[25] Rachid Guerraoui and Andre Schiper. Transaction model vs. virtual synchrony model: Bridging the gap. In *Selected Papers from the International Workshop on Theory and Practice in Distributed Systems*, pages 121–132. Springer-Verlag, 1995.

[26] V. Hadzilacos and S. Toueg. A modular approach to fault-tolerant broadcasts and related problems. Technical Report TR94-1425, Cornell University, Department of Computer Science, May 1994.

[27] L.R. Halme and R.K. Bauer. Aint misbehaving: A taxonomy of anti-intrusion techniques, 2000.

[28] K. P. Kihlstrom, L. E. Moser, and P. M. Melliar-Smith. The SecureRing protocols for securing group communication. In *Proceedings of the 31st Annual Hawaii International Conference on System Sciences*, pages 317–326, January 1998.

[29] C.E. Landwehr, A.R. Bull, J.P. McDermott, and W.S. Choi. A taxonomy of computer program security flaws. *ACM Computing Surveys*, 26(3):211–254, 1994.

[30] J.-C. Laprie, editor. *Dependability: Basic Concepts and Terminology*, volume 5 of *Dependable Computing and Fault-Tolerance*. Springer-Verlag, Vienna, Austria.

[31] J.-C. Laprie. Dependable computing and fault tolerance: Concepts and terminology. In *15th IEEE Int. Symp. on Fault Tolerant Computing (FTCS-15)*, pages 2–11, Ann Arbor, MI, USA, 1985. IEEE CS Press.

[32] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[33] F. Meyer and D. Pradhan. Consensus with dual failure modes. In *Proceedings of the 17th IEEE International Symposium on Fault-Tolerant Computing*, pages 214–222, July 1987.

[34] H. Miranda, A. Pinto, and L. Rodrigues. Appia, a flexible protocol kernel supporting multiple coordinated channels. In *Proceedings of the 21st International Conference on Distributed Computing Systems*, pages 707–710, Phoenix, Arizona, April. IEEE.

[35] N. F. Neves and P. Veríssimo, editors. *Complete Specification of APIs and Protocols for the MAFTIA Middleware. Project MAFTIA deliverable D9.* July 2002. http://www.newcastle.research.ec.org/maftia/deliverables/D9.pdf.

[36] V. Nicomette and Y. Deswarte. Symbolic rights and vouchers for access control in distributed object systems. In J. Jaffar and R. H. C. Yap, editors, *Proc. 2nd Asian Computing Science Conference (ASIAN'96)*, LNCS n1179, pages 193–203, Singapore, 1996. Springer-Verlag.

[37] V. Nicomette and Y. Deswarte. An authorization scheme for distributed object systems. In *Proc. Int. Symposium on Security and Privacy*, pages 21–30, Oakland, CA, USA, 1997. IEEE Computer Society Press.

[38] NSA. Nsa glossary of terms used in security and intrusion detection, 1998.

[39] M. Patiño Martínez, R. Jiménez-Peris, and S. Arévalo. Group transactions: An integrated approach to transactions and group communication. In *Workshop on Concurrency in Dependable Computing (at 22nd International Conference on Application and Theory of Petri Nets and 2nd International Conference on Application of Concurrency to System Design*, pages 5–15, Newcastle upon Tyne, UK, 2001.

[40] D. Powell, D. Seaton, G. Bonn, P. Veríssimo, and F. Waeselynk. The Delta-4 approach to dependability in open distributed computing systems. In *Proceedings of the 18th IEEE International Symposium on Fault-Tolerant Computing*, June 1988.

[41] David Powell (Guest Ed.). Group communication. *Communications of the ACM*, 39(4):50–97, April 1996.

[42] Michael O. Rabin. Randomized Byzantine generals. In *Proc. 24th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 403–409, 1983.

[43] B. Randell, A. Romanovsky, R. J. Stroud, J. Xu, and A. F. Zorzo. Coordinated atomic actions: from concept to implementation. Technical Report TR 595, Department of Computing, University of Newcastle upon Tyne, 1997.

[44] M. Reiter. Secure agreement protocols: Reliable and atomic group multicast in Rampart. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pages 68–80, November 1994.

[45] Michael K. Reiter. Distributing trust with the Rampart toolkit. *Communications of the ACM*, 39(4):71–74, April 1996.

[46] Michael K. Reiter and Kenneth P. Birman. How to securely replicate services. *ACM Transactions on Programming Languages and Systems*, 16(3):986–1009, May 1994.

[47] Andre Schiper and Michel Raynal. From group communication to transactions in distributed systems. *Commun. ACM*, 39(4):84–87, 1996.

[48] Adi Shamir. How to share a secret. *ACM*, 22(11), November 1979.

[49] Victor Shoup. Practical threshold signatures. In Bart Preneel, editor, *Advances in Cryptology: EUROCRYPT 2000*, volume 1087 of *Lecture Notes in Computer Science*, pages 207–220. Springer, 2000.

[50] Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In Kaisa Nyberg, editor, *Advances in Cryptology: EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 1998.

[51] P. Veríssimo and A. Casimiro. The Timely Computing Base model and architecture. *IEEE Transactions on Computers*, 51(8):916–930, August 2002.

[52] P. Veríssimo, A. Casimiro, and C. Fetzer. The Timely Computing Base: Timely actions in the presence of uncertain timeliness. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 533–542, June 2000.

[53] P. Veríssimo, A. Casimiro, and C. Fetzer. The Timely Computing Base: Timely actions in the presence of uncertain timeliness. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 533–542, June 2000.

[54] P. Veríssimo and L. Rodrigues. *Distributed Systems for System Architects*. Kluwer Academic Publishers, 2001.

[55] P. E. Veríssimo, N. F. Neves, and M. P. Correia. Intrusion-tolerant architectures: Concepts and design. In R. Lemos, C. Gacek, and A. Romanovsky, editors, *Architecting Dependable Systems*, volume 2677 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[56] Paulo Veríssimo. Uncertainty and predictability: Can they be reconciled? In *Future Directions in Distributed Computing*, pages –. Springer-Verlag LNCS 2584, to appear 2003.

[57] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, and G. Alonso. Understanding replication in databases and distributed systems. In *Proceedings of the The 20th International Conference on Distributed Computing Systems (ICDCS 2000)*, page 464. IEEE Computer Society, 2000.

[58] J. Xu, B. Randell, A. Romanovsky, C. Rubira, R. Stroud, and Z. Wu. Fault tolerance in concurrent object-oriented software through coordinated error recovery. In *Proceedings of the Twenty-Fifth International Symposium on Fault-Tolerant Computing*, page 499. IEEE Computer Society, 1995.

[59] Lidong Zhou, Fred B. Schneider, and Robbert Van Renesse. Coca: A secure distributed online certification authority. *ACM Trans. Comput. Syst.*, 20(4):329–368, 2002.

« Prêt-à-voter » : un système de vote électronique pratique et vérifiable par les votants

Peter Ryan

Professeur à l'Université de Newcastle
Grande-Bretagne

Prêt à Voter

Peter Y A Ryan
University of Newcastle

Menu

- The problem.
- Voter-verifiability.
- Overview of Prêt à Voter "Classique".
- Some vulnerabilities with Prêt à Voter "Classique".
- Enhancements:
  - Distributed generation of ballot forms
  - On demand creation of ballot forms
  - Cut-and-choose revisited and post-auditing.
  - Re-encryption mixes.
  - Remote, coercion-resistant version.
  - Spooky voting at a distance (?).
- Conclusions.

The Problem

- From the dawn of democracy it was recognised that people would be tempted to try to corrupt the outcome of elections.
- Highly adversarial: system trying to cheat voters, voters trying to cheat the system, coercers trying to coerce voters, voters trying to fool coercers etc.
- The Ancient Greeks experimented with primitive technological solutions to try to shift the trust from people (officials) to mechanical devices.
- In the US they have been using technological devices for voting for over a century: e.g., level machines since 1887, punch cards, optical scans, touch screen etc. Prompted by high instance of fraud with paper ballots.
- All have problems, see "Steal this Vote" Andrew Gumbel.

"The Computer Ate my Vote"

- In the 2004 US presidential election, ~30% of the electorate used DRE, touch screen devices.
- Aside from the "thank you for your vote for Kerry, have a nice day" what assurance do they have that their vote will be accurately counted?
- What do you do if the vote recording and counting process is called into question?
- No paper trail to fall back on.
- Voter Verifiable Paper Audit Trail (VVPAT) and "Mercuri method". But paper trails are not infallible.

The challenge

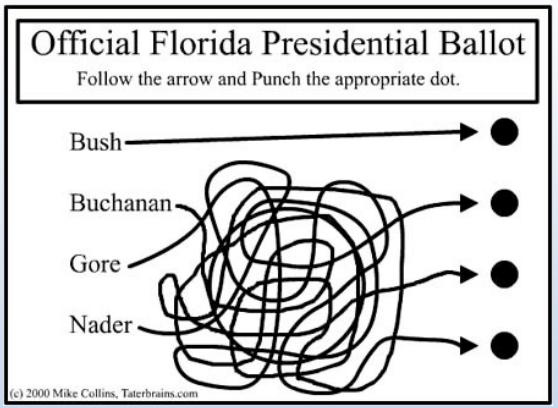
- Digital voting technologies hold out promise of accessible and efficient democracy.
- But there are dangers, witness the fiascos in the US (e.g., Johns Hopkins report, Rubin et al)
- Want high assurance that all votes are accurately recorded and counted-whilst maintaining ballot secrecy.
- The challenge is to reconcile these two conflicting requirements whilst minimising dependence on the components (booths, tellers etc.) of the scheme.
- Difficulty in validating an election: "correct" outcome not known due to secrecy requirement.
- Can be viewed as a special case of secure distributed computation, but need socio-technical approach.

Remote vs Supervised

- Important to draw a clear distinction between supervised and remote voting.
- In the former the voter casts their vote in enforced isolation, e.g., in a booth in a polling station.
- Remote voting, e.g., internet, postal etc. such isolation cannot be enforced.
- Hence dangers of coercion.
- The bulk of the talk will be about supervised, but we will venture into remote towards the end….

Hazards of e-voting!

Florida 2000



The spectrum of assurance

- Assurance can be derived from, at one end of the spectrum, (claims of) verification that the system will perform correctly.
- At the other end, run-time monitoring.
- Many voting systems lie at the verification end of this spectrum.
- Such assurance is very fragile: any failure of design, implementation, evaluation, deployment or maintenance can undermine it.
- Monitoring is much more robust but tends to conflict with secrecy requirements.
- Verify the election rather than the system.
- But note: detecting violations of secrecy is tricky.

Technical Requirements

- Key requirements:
  - Integrity/accuracy: count (sufficiently) accurately reflects votes cast.
  - Ballot secrecy: the way a voter cast their vote should only be known to the voter.
  - Voter anonymity: the fact that a given voter cast a vote should be secret.
  - Voter verifiability: the voter should be able to confirm that their vote is accurately included in the count and prove to a 3rd party if it is not (whilst not revealing their vote).
  - Universal verifiability: everyone should be able to verify the count.
  - Coercion resistance: there should be no way for the voter to prove to a coercer which way they voted.
  - Availability: all eligible voters should be able to cast their vote without let or hindrance throughout the voting period.
  - Ease of use, public trust, cost effective, scalable etc. etc…..

Accuracy

- Can recast this in terms standard protocol requirements:
- Think of the process of casting and counting a vote as the delivery of a message from the voter to their chosen candidate.
- Need to show:
  - Authentication
  - Assured delivery.

Ballot secrecy

- Ballot secrecy and voter anonymity seem to be used synonymously.
- In fact turn out to be distinct properties.
- Ballot secrecy:
  - Third party cannot establish which way a voter voted (even with voter collusion).
  - Not always feasible, e.g. if all votes go the same way. Probabilities can be assigned based on final counts.
  - Require that the adversary cannot do better than a probability based on final count?

vs. Voter anonymity

- Voter anonymity (d'après Steve Schneider):

$$\forall \pi: \Pi_V \bullet A \quad S \cong A \quad S[\pi]$$

  - Observer, with observational capabilities encoded in the abstraction A , cannot distinguish instantiations of the system with the voter's identities permuted.
- Note: works fine even if all votes go one way. Suggests that anonymity is more appropriate?
- Note: we are assuming that the fact of having voted not deemed confidential.
- Extend definition to permutations over all eligible voters.

Coercion resistance

- Intuitively: even where the coercer can demand that the voter "reveal" secret keys, the voter can fool the coercer.
- More formally, for all $C_2 \in C$, and any run of the protocol in which the voter votes for C_1 using secret(s) k_1, for all C_2 there exists k_2 such that another run that casts a vote for C_2 with k_2 that is indistinguishable to the coercer.
- Note:
  - Similarity to "plausible deniability" (=opacity?).
  - Similarity to resistance to guessing attacks.
  - Convergence of "formal" and crypto definitions of secrecy.

Assumptions

- For the purposes of the talk we will make many sweeping assumptions, e.g.,:
  - An accurate electoral register is maintained.
  - Mechanisms are in place to ensure that voters can be properly authenticated.
  - Mechanisms are in place to prevent double voting.
  - Existence of a secure Web Bulletin Board.
  - Crypto algorithms we use are secure.
  - Etc.

Cryptographic approaches

- Various approaches have been tried:
  - Blinded digital signatures (FOO).
  - Homomorphic tabulation (cf lever machines)
  - Mix net tabulation (cf ballot box).

Voter-verifiability in a nutshell

- Idea due to Benaloh, later Chaum and Neff.
- Voters are provided with an encrypted "receipt".
- Copies of the receipts are posted to a secure web bulletin board. Voters can verify that their (encrypted) receipt is correctly posted.
- A (universally) verifiable tabulation is performed on the receipts.
- Checks are performed at each stage to detect any attempt to decouple the encryption of the receipt from the decryption performed by the tellers.
- But, proofs provided to the voter of correct encryption of their vote must not be transferable.

Prêt à Voter

- Uses pre-prepared ballot forms that encode the vote in familiar form (an × against the chosen candidate).
- The candidate list is (independently) randomised for each ballot form.
- Information allowing the candidate list to be reconstructed is buried cryptographically in an "onion" on each ballot form.
- An excess number of forms are generated to allow for random auditing, before, during and after the election.

Example

- Each ballot form has a unique, secret, random seed *s*.
- For each form, a permutation of the candidate list is computed as a publicly known function of this seed.
- The seed information is buried cryptographically using public keys of a number of tellers in an "onion" printed on the form.
- The seed can only be extracted by the collective actions of tellers, or suitable subset if a threshold scheme is used.

Typical Ballot Sheet

Epicurus	
Democritus	
Aristotle	
Socrates	
Plato	
	$rJ9*mn4R&8

The voting “ceremony”

- Can be varied, but possible scenario:
  - Voter enters the polling station and takes a ballot form at random, sealed in an envelope (and/or scratch strip concealing the onion).
  - The voter goes to a booth, extracts the ballot form and makes their mark.
  - LH strip is destroyed/discarded.
  - The voter leaves the booth with the LH strip, the receipt, and registers with an official.
  - A digital copy, *(r, Onion),* of the receipt is made. The receipt is digitally signed and franked. Additionally, a paper audit copy is made.
  - The voter exits the polling station with their receipt.
  - Once the election is closed, digital copies of the receipts are posted to the Web Bulletin Board.
  - The voters can visit the WBB and confirm that their receipt appears correctly.

Voter marks their choice

Epicurus	
Democritus	✘
Aristotle	
Socrates	
Plato	
	$rJ9*mn4R&8

Voter’s Ballot Receipt

✘
$rJ9*mn4R&8

Remarks

- Note that the receipt reveals nothing about the vote:
  - The onion carries the crypto seed, encrypted with the teller’s public keys, that (a threshold subset of) the tellers use to reconstruct the permutation of the candidate list.
  - Without all of these secret keys (or an appropriate subset) the candidate list cannot be reconstructed and hence the vote value cannot be recovered.
- Vote is not directly encrypted, rather the frame of reference, i.e., the candidate list, is randomised and information defining the frame is encrypted.
- Vote casting can be in the presence of an official (à la Française).
- An encrypted paper audit trail can be incorporated.
- Works for ranked, STV etc.

Anonymisation and tabulation

- Once the election has closed all the receipts are posted to the WBB, a set of tellers perform a cascade of robust anonymising mixes on the receipts:
  - Receipts are decrypted by stages and undergo multiple secret shuffles. Intermediate stages are also posted to the WBB for audit.
  - Tellers transform the "r" index value. The final "r" values that emerge from the mixes give the original vote value in the canonical basis.
  - Any link between the original receipts and the decrypted values will be lost.

Seeds and onions

- Suppose that we have *k* tellers. Each teller will perform 2 mixes and has two RSA public key pairs. For each ballot form *2k* random germs are generated:

$$\varphi_i \in Z_N \text{ (some modest size N, e.g., } 2^{32})$$

- The seed value is taken to be the sequence of these germ φ values:

$$s := \langle \varphi_0, \varphi_1, \varphi_{2v}, \varphi_3, \ldots\ldots, \varphi_{2k-1} \rangle$$

Onion construction

- The germs are buried in the 2k layers of the onion:
- θ_0 is a random value, unique to each ballot form. Then:

$$\theta_{i+1} := \{\varphi_i, \theta_i\}_{PKTi}, \; i = 0, \ldots, 2k-1$$
$$\Theta := \theta_{2k}$$

- Thus, the onion Θ has a layered construction:

$$\Theta := \{\varphi_{2k-1}, \{\varphi_{2k-2}, \{\ldots\ldots, \{\varphi_2, \{\varphi_1, \{\varphi_0, \theta_0\}_{PKT\_0}\}_{PKT\_1}\}_{PKT\_2} \ldots\ldots\}_{PKT\_2k-2}\}_{PKT\_2k-2}\}_{PKT\_2k-1}$$

Candidate permutations

- These germs are used as arguments for an agreed, public function *f* from the seed space to the space of permutations for each teller mix:

$$\pi_i := f(\varphi_i), \; i=0 \text{ through } 2k-1$$

- The final candidate list permutation σ is computed as the product of the 2k permutations computed above applied to the basis ordering σ_0 to give the candidate order σ shown on the ballot form:

$$\sigma := \Pi_{i=0}^{2k-1} \pi_i \circ \sigma_0$$

Basis ordering σ_0

- We assume some canonical, basis ordering σ_0 from which all the permuted orderings on the ballot forms are derived by applications of the permutation functions derived from the hidden seed values:
- σ_0 :=

  Aristotle
  Democritus
  Epicurus
  Plato
  Socrates

Teller transformations

- Transformations on the ballot pairs:
- On each ballot pair (r_i, θ_i), the teller performs the transformation:

$$(r_i, \theta_i) \rightarrow (r_{i-1}, \theta_{i-1})$$

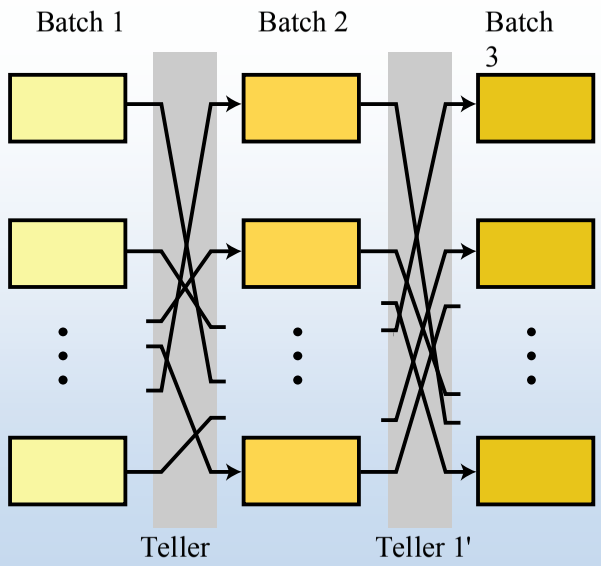
- Recall:

$$\{\theta_i\}_{SKTi-1} = \varphi_{i-1}, \theta_{i-1}$$

- And:

$$r_{i-1} = f^{-1}(\varphi_{i-1})\ (r_i)$$

- Thus, one layer of onion is stripped off and the revealed germ is used to compute the inverse of the *ith* permutation, which is applied to the index value.
- The final pair, (r_0, θ_0) comprises the index value that represents the vote value in the basis ordering σ_0 along with the inner onion value.



But s**t happens…

- All this is fine as long as we are happy to trust "The Authority", the tellers etc.
- But we want to trust nobody, just trust in cryptography.
- For the accuracy requirement:
  - Ballot forms may be incorrectly constructed, leading to incorrect encoding of the vote.
  - Ballot receipts could be corrupted before they are entered in the tabulation process.
  - Tellers may perform the decryption incorrectly.
- We now discuss the (fault) detection mechanisms.

Checking the ballot forms

- We need to check that the seed buried in the onion does correspond to the candidate permutation printed on the ballot form.
- Checks can be performed by auditors and the voters to catch such corruption:
  - Random audits of ballot forms performed before, during and after the election period by the Electoral Reform Soc etc.
  - Voters could also be invited to perform similar checks on randomly selected "dummy" forms, e.g., randomly select a pair of forms, one to check (and discard), one to cast their vote.

Auditing ballot forms

- To check the construction of the ballot forms the values on the form, onion and candidate ordering, can be reconstructed if the seed value is revealed.
- We can use the tellers in an on-demand mode to reveal the secret seed value buried in the onion. Avoids problems with storing and selectively revealing seeds.
- Note, for this checking process, the tellers are used in an on-demand basis before and during the election-quite different to the batch mode for the anonymising mix after the election has closed.

Ballot form checking modes

- This oracle teller mode suggests several ways for voters to check the well-formedness of ballot forms:

  1. Simple, single dummy vote
  2. Multiple or ranked dummy vote
  3. Given the onion value, the tellers return the candidate ordering

- Note: vulnerable to authority/tellers collusion attacks.
- The auditor checks are the more rigorous: not vulnerable to authority/teller collusions.

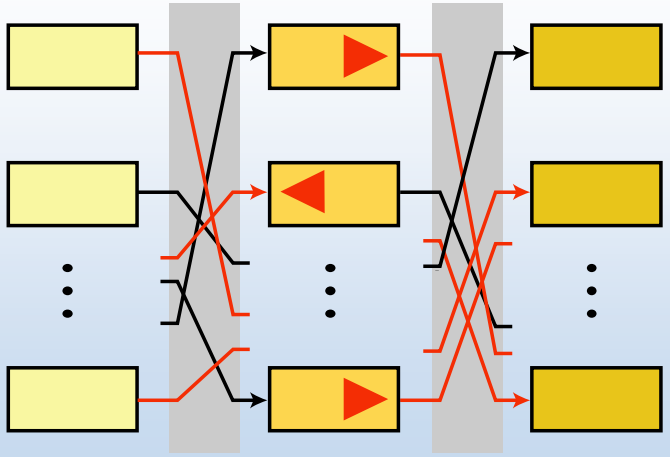
Recording and transmission

- To check that receipts are accurately recorded and input into the mix:
  - Voters can visit the WBB and check that their receipt appears correctly recorded.
  - Voter checks can be supplemented by independent audit authorities checking the WBB against the VEPAT style record of ballot receipts.

Auditing the tellers

- Partial Random Checking (PRC) of the teller transformations: auditor randomly selects half the of the links to be revealed and checked, but in such a way as not to reveal any links across the two transformations performed by the teller.
- Go down middle WBB column for each teller and randomly assign ► or ◄ to each pair.
- For a ►(◄), the tellers reveal the outgoing (incoming) link along with the associated germ value.
- Note: because no complete paths across a given teller's pair of mixes are revealed by the audit process, we can audit the tellers independently.

Auditing the tellers



Assurance arguments

- Assuming absence of collusion between auditors and The Authority and the tellers, the chance of p ballots being corrupted undetected falls off as:

$$(z)^p$$

- Where z is the proportion of forms randomly audited.
- Or

$$(1/2)^p$$

- For the tellers.
- For secrecy we rely on defence in depth: (almost) all the tellers would have to be compromised to compromise secrecy.

Advantages

- Voter experience simple and familiar.
- No need for voters to have personal keys or computing devices.
- Ballot form commitments and checks made before election opens ⇒ neater recovery strategies.
- Votes are not directly encrypted, just the frame of reference in which votes encoded. Hence:
  - The vote recording device doesn't get to learn the vote.
  - No need for ZK proofs of correctly formed encrypted receipts or cut-and-choose protocols. (but onus of proof shifts to the well-formedness of the ballot forms).
  - Avoids subliminal, kleptographic and side channels as well as social engineering attacks.

Vulnerabilities

1. Need to trust "The Authority" for secrecy (not for accuracy).
2. Need to protect ballot form information (chain of custody).
3. Chain voting.
4. Need to trust the auditors (absence of collusion with the tellers).
5. Need to trust tellers not to leak information (aside from audit info).
6. Subliminal, side and kleptographic channels.
7. "Social engineering" attacks.
8. Enforcing the destruction of LH strips.
9. Separation of teller modes, i.e., ensure that each ballot form is processed only once.
10. Need to constrain the WBB audits, i.e., reveal only L or R links.
11. Ballot stuffing.
12. DoS attacks.

Counter-measures

- 1-3 will be countered by the distributed, encrypted construction of "proto"-ballot forms.
- 4: can be countered by audit selection using a prior agreed crypto hash, Fiat-Shamir style.
- Re-encryption mixes help here too, see later: allows rerunning and independent re-auditing of the mixes.

Chain Voting

- Chain voting is a known style of attack that can be effective against some conventional paper ballot schemes. In this attack, the coercer smuggles an unused ballot form out of the polling station and marks his preferred candidate. The voter is told that they will be rewarded if they emerge with a fresh, unmarked form. This can then be marked again and passed to the next voter.
- Systems with pre-printed ballot forms, in particular Prêt à Voter, are vulnerable, especially where the forms are a controlled resource.
- The French voting system avoids this.
- Prêt à Voter mark II, will also counter it due to the on-demand creation of the ballot forms.

Subliminal and side channels

- Many crypto schemes, e.g., Chaum, Neff etc are potentially vulnerable to subliminal, side and kleptographic channels.
- Voter's choice is communicated in the booth to the encrypting device. Hence the device might leak information via random (Neff) of semantic (Chaum) or side channels.
- In Prêt à Voter, non-determinism is resolved before voter choices are revealed or association between ballot forms and voters is established.
- And voter choice is not communicated to the device.
- => information flow analysis.

Kleptographic channels

- These occur where a crypto device may select crypto variables in such a way to leak information to a colluding party.
- Prêt à Voter "Classique" is vulnerable: The Authority might choose seed values in such a way that a certain keyed hash of the onion value leaks information about the candidate list to a colluding entity (who shared the hash key).
- Note: Authority behaviour looks innocent.
- Distributed generation of ballot forms will counter this: no single entity determines the crypto variables.

Social engineering attacks

- Cryptographic voting schemes, e.g., Chaum and Neff (VoteHere), frequently involve moderately complex, multi-step protocols between the voters and the devices.
- Opens up possibilities for a malicious device to fool the voter about the protocol sequence, e.g., turning a cut-and-choose into a choose-and-cut.
- Prêt à Voter Classique seems fairly immune due to extremely simple protocol sequence.

Destruction of LH column

- For coercion resistance it is essential that voters not be able to exit the polling station with the LH strip.
  - Procedural: officials oversee destruction of LH strips.
  - Mechanical: device that automatically strips off the LH strip and discards it.
  - Decoy strips: plentiful supply of alternative LH strips provided in the booth.
  - Scratch strips: onion under the strip (in 2D bar code?) candidate list overprinted: revealing the onion destroys the list.
  - Disc ballots!? Ballot "forms" take the form of a pair of discs sealed together. After selection they are separated. Axial symmetry ensures that the original configuration is lost.
  - Quantum!? Ballot "forms" using entangled q-bits. Measurement to reveal candidate lists collapses the wave functions.

Ballot stuffing

- Having the voters check for the appearance of their receipt on the WBB doesn't detect ballot stuffing: in which the authorities add spurious receipts.
- Counter-measures:
  - Check numbers of votes cast again number posted.
  - The VEPAT helps here.
  - Incorporate voter signatures?

Denial of Service

- Tricky in general.
- VEPAT helps.
- Re-encryption mixes help: can bin faulty mix tellers and rerun mixes and audits if necessary.

Enhancements to Prêt à Voter 2005

- Distributed generation of ballot forms.
- Concealment of onion/candidate list associations.
- On demand printing of ballot forms.
- Re-encryption mixes.

Distributed creation of ballot forms

- We would like to set things up so that:
  - Only the voter gets to see the onion/candidate list association for their ballot form in the isolation of the booth.
  - No single entity knows or controls the seed entropy.
  - And enable on-demand creation and printing of ballot forms.
- This can be achieved with a “pre-mix” that (roughly) mirrors the tabulation mixes.
- Mixing is done under an extra layer of encryption that can be stripped off at the last moment.
- Can be adapted for remote variants.

ElGamal

- Let α be a generator of cyclic group, e.g., Z_p^*, p a large prime. Choose k ($2 \le k \le p-2$) and let $\beta = \alpha^k \pmod p$.
- p, α and β made public, k kept secret.
- Encryption of m in {0, …, p-1}:

$$(\alpha^x, \beta^x.m) =: (y_1, y_2)$$

- For random x in {0,…,p-1}.
- Decryption:

$$m = y_2 / y_1^k$$

- Re-encryption: choose random y in {0,…,p-1} and form:

$$(\alpha^x.\alpha^y, \beta^x.\beta^y.m) = (\alpha^{x+y}, \beta^{x+y}.m)$$

- Note: same as directly encrypting m with randomisation $x+y$.
- Note: re-encryption does not require knowledge of a key.

Distributed generation of proto ballot forms

- An initial clerk generates a set of "entangled" pairs of ElGamal "onions": pairs have the same initial plaintext seed s_i^0 :

$$\{(\alpha^x, \beta_R{}^x.s_i^0), (\alpha^y, \beta_T{}^y.s_i^0)\}$$

- The first onion is encrypted under the Registrar's (threshold) PK β_R, the second under the Teller's (threshold) PK's β_T .
- These pairs are put through a set of q-1 re-encryption/transformation mixes by independent clerks:

$$\{(\alpha^{x'}, \beta_R{}^{x'}.s_i^{h+1}), (\alpha^{y'}, \beta_T{}^{y'}.s_i^{h+1})\}$$

$$s_i^{h+1} = s_i^h \times \hat{s}_i^h \pmod p$$

- i.e. a re-encryption and injection of fresh entropy $\hat{s}_i^h$ to the seed value s.

Distributed generation

- After q of mixes:

$$((\alpha^{x'}, \beta_R{}^{x'}.s^*)), (\alpha^{y'}, \beta_T{}^{y'}.s^*))$$

$$s^* := s_i^q$$

- Thus, these pairs go through re-encryption mixes and each of the *q* clerks contributes to the final seed value.
- These pre-mixes could be audited.
- These pairs can now be distributed in this form with the candidate list encrypted.
- Proto-ballot forms with the two onions printed on them:

Proto-ballot forms

784663023897	887250055758

Revealing the ballot form

- The booth device can then decrypt the LH onion, say, to give the candidate permutation σ:

$$\{\sigma, (\alpha^y, \beta_T{}^y.s)\}$$

- We might have arranged for the LH onions to be encrypted under the booth key.
- Alternatively the booth device might transmit the LH onion to the registrars that perform a threshold decryption and return the seed to the booth.

Remote version

- Can be adapted to remote versions (e.g., use the Cornell protocol to convert the LH onion to be encrypted under the voter V_i's PK)

$$((\alpha^x, \beta_{Vi}{}^x . s^*)), (\alpha^y, \beta_T{}^y . s^*))$$

- A similar construction is possible for the original RSA onions (might have to be careful about possible matching of terms between mix and pre-mix where deterministic primitives are used).

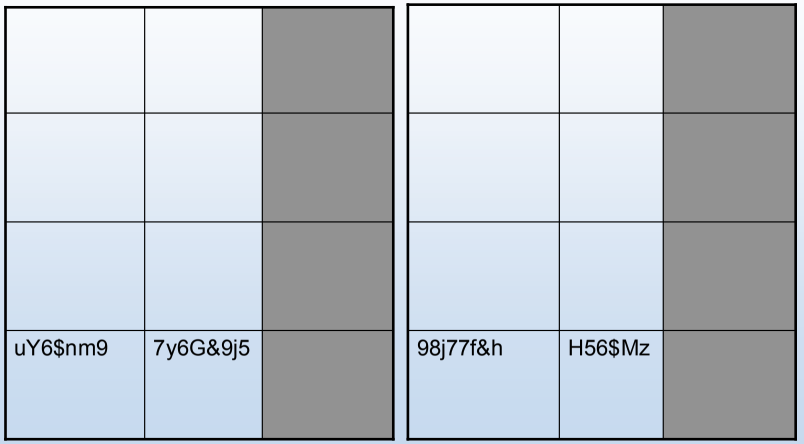
On-demand printing

- So, we can minimise chain voting and chain of custody problems by arranging the print ballot forms at the last moment, i.e., in the booth.
- Note subtle distinction between creating and printing.
- Use fresh, additional sources of entropy, e.g., fibres in the paper, the voters themselves etc.
- The problem now is that we can't pre-audit pre-committed ballot forms.

Cut-and-choose revisited

- A possible solution is to re-introduce cut-and-choose element to the voter protocol along with post-auditing.
- This might be implemented using double-sided proto-ballot forms:

2 sided proto-ballot forms



Decrypted 2 sided ballot forms

Thales		
Plato		
Socrates		
uY6$nm9	7y6G&9j5	

Plato		
Thales		
Socrates		
98j77f&h	H56$Mz	

Vote selection

- Forms can be decrypted and printed in the booth.
- The two sides show independent PaV ballot forms.
- Voter makes an arbitrary choice as to which side to use.
- They mark their cross (or ranking etc) against the candidate of choice as before.
- The other side is left blank.
- The LH column of the chosen side is destroyed, and with it the greyed, RH column of the other side:

Vote selection

Thales		
Plato	X	
Socrates		
uY6$nm9	7y6G&9j5	

Plato		
Thales		
Socrates		
98j77f&h	H56$Mz	

Receipt

X	
7y6G&9j5	

Plato	
Thales	
Socrates	
98j77f&h	H56$Mz

Discussion

- The unused side can now be checked for well-formedness (at the time of casting and later in the WBB).
- This avoids some of the vulnerabilities of PaV Classic but at the cost of re-introducing some complexity in the voting protocol.
- Note: still no need for the voter to communicate their vote to the device, hence no subliminal/side channels etc.
- Distributed creation of seeds counters kleptographic attacks.
- Note: symmetry between the two sides, hence no psychological bias.

Post-auditing

- The flip sides of receipts could be checked at the time of casting.
- In addition, all the info would be posted to the WBB. The flip, unused sides would be checked for well-formedness.
- Seeds revealed for the unused sides.
- Need mechanisms to prevent leakage of seeds for used sides, e.g., authorisation code in LH column?

Discussion

- This solution avoids a lot of the vulnerabilities of PaV Classic, e.g., no need to trust "The Authority", but makes the protocol more complex for the voter. And may re-introduce the possibility of "social-engineering" attacks (Karlov et al).

Re-encryption mixes

- Prêt à Voter "Classic" uses Chaumian (decryption) mixes. Now we explore adaptation to re-encryption mixes.
- Advantages of re-encryption:
  - Tellers inject fresh entropy at each stage, hence onion size doesn't grow with number of tellers and germ size.
  - Less dependence on availability of tellers: a faulty mix teller can just be binned and replaced.
  - Full mixing over the ElGamal group.
  - Clean separation of mixing and decryption stages.
  - Mixes and audits can be rerun afresh.
- Downsides:
  - Need shuffle commitments.
  - Tricky to mesh with Prêt à Voter's special encoding of votes.

Re-encryption mixes

- Prêt à Voter's rather special representation of the vote in the receipts makes it tricky to mesh with re-encryption mixes. Some possible approaches:
  1. Leave r, index terms unchanged through the mixes.
  2. Follow re-encryption mixes with Chaumian decryption mixes.
  3. Absorb the r into the onion value.
  4. transform both r and θ terms leaving vote value invariant
  5. Add teller transforms to the index values, storing the entropy in an extra (pre-generated and audited) "onion" value.
  6. Use crypto-homomorphism approaches.

Re-encryption mix/tabulation

- The special form of Prêt à Voter receipts, index valid plus ElGamal "onion", makes re-encryption mix/tabulation slightly tricky.
- Alternative:
  - For simplicity consider just random cyclic shifts of the candidate list and single choice voting.
  - Let s be the candidate list offset. For some suitable γ, encrypt γ, to form the onion:
    $$(\alpha^x, \beta^x . \gamma^{-s})$$
  - A receipt pair can now be transformed into a pure ElGamal form:
    $$(r, \alpha^x, \beta^x . \gamma^{-s}) \rightarrow (\alpha^x, \beta^x . \gamma^{r-s})$$
  - Note: the value *r-s* indicates the vote value in the base ordering.
  - This ElGamal term can be put through a conventional re-encryption mix and the final (threshold) decryption yields γ^{r-s}.

Decryption

- Final decryption will yield:
  $$\gamma^{r-s} \pmod p$$
- The vote value is *r-s (mod n)*
- Which is nice, except that to get *r-s* we need to find a discrete log.
- If we generated the seeds s randomly from Z^*_p, s would range over {0,…,p-1}. So extracting *r-s* would be intractable.
- To avoid this whilst maintaining coercion resistance, we adapt the construction of the proto-ballot forms:
- At each stage of the pre-mix the clerk draws the seed entropy to a binomial distribution around 0 with σ.
- If there are q clerks, the final seed values will be drawn from a binomial distribution mean 0, standard deviation = $\sigma\sqrt{q}$.
- Set up a look-up table for the logs out to some multiple of $\sigma\sqrt{q}$. Occasionally we may have to search out beyond this.

Modified proto-ballot forms

- Clerk C_0 generates a set of "entangled" pairs of ElGamal "onions". A pair has the same initial plaintext seed s_i^0 :
  $$\{(\alpha^x, \beta_R{}^x . \gamma^{-si0}), (\alpha^y, \beta_T{}^y . \gamma^{-si0})\}$$
- The first onion is encrypted under the Registrar's (threshold) PK β_R, the second under the Teller's (threshold) PK's β_T .
- These pairs are put through a set of q-1 re-encryption/transformation mixes by independent clerks:
  $$\{(\alpha^{x'}, \beta_R{}^{x'} . \gamma^{-si,\, h+1}), (\alpha^{y'}, \beta_T{}^{y'} . \gamma^{-sh+1,\, h+1})\}$$
  $$s_i^{h+1} = s_i^h + \hat{s}_i^h \pmod p$$
- At each stage, the fresh entropy $\hat{s}_i^h$ is drawn from a binomial mean 0, standard deviation σ to the seed value s_i.
- The final have a binomial distribution centred around 0 with standard deviation $\sigma\sqrt{q}$.

Discussion

- Plausible deniability is retained (no outer values).
- Need more elaborate coding for full permutations.
- Could use separate onions for each cell.
- Is there a nice crypto primitive or trick to sidestep this?
- Note: for STV, ranked etc, we can mix the ballot cells separately.
- May work more elegantly with Paillier encryption thanks to homomorphic properties:

$$\{m_1\}_K \times \{m_2\}_K = \{m_1+m_2\}_K$$

Auditing re-encryption mixes

- We can use the Partial Random Checking approach again here, except that now the re-randomisation factor is revealed for an audited link rather that the germ value.
- Actually some subtleties due to the extra freedom the tellers have in the re-randomisation.
- Use shuffle commitments.
- Other approaches can also be applied now that we are dealing with ElGamal terms, e.g., the Neff approach.

Remote, coercion-resistant Prêt à Voter

- Joint work with Michael Clarkson and Andrew Myers, Cornell.
- Still speculative.

Coercion resistance

- The voter should have no way to prove to the coercer which way they voted, even if they are prepared to cooperate with the coercer.
- Coercer can observe virtually all steps of the protocol, the WBB and even demand the voter to reveal keys, choose randomness etc, but the voter should be able to lie undetected.
- Need to postulate at least a window in which the voter can interact with the voting system unobserved.
- And assume availability of anonymous channels.
- Need to assume that the voter's private (authentication) key remains secret at least during registration phase?

Remote Prêt à Voter

- Naïve approach: casting vote by just submitting an onion and index value. Open to coercion.
- More sophisticated, coercion resistant version (à la Juels et al and Clarkson, Myers): supply voters with a proto-ballot form and a "capability".
- Capabilities constructed like onions but with "valid" flag at the centre.
- Casting: voter casts a triple:

  (index, onion, capability)
- Coerced voter can corrupt their capability. Invalidity only revealed after the anonymising mixes.
- Designated verifier proof (DVP) to convince the voter, and only the voter, of the validity of their capability.

Capabilities

- Construction similar to ElGamal "onions", but with plaintext: registrar signature on a "valid" string along with a nonce:

  $m_i := (Sig_{Reg}(valid, N_i))$

  $Capability_i := (\alpha^x, \beta^x.m_i)$
- Delivered to the voters along with a non-interactive, Designated Verifier Proof (DVP) of validity.
- Or perhaps provide voters with a capability in person at a registration centre (with interactive proof).
- Designed to go through the mix and be decrypted along with the associated ballot: validity only revealed after anonymisation.
- Need to ensure that each voter gets exactly one valid capability.

Capabilities

- Again, we would like to construct these capabilities in such a way that no single entity knows their values or the association with voter ids.
- And without having to trust anyone as to their correct construction.
- Ideally, we would also like to have the voters participate in their construction to counter official ballot stuffing, whilst preserving anonymity.
- At the moment it is not clear how to achieve all this without some level of trust in a (distributed) registrar and some secure (e.g., anonymous) channel.
- Need to assume voters have private/public key pairs.
- List of valid nonces committed ahead of time?

Capabilities

- Possible approach: have a number of registrars generate fragments of the capability under a layer of encryption along with DVP proofs.
- Use suitable algebraic features to enable to voter to assemble these into a full capability in such a way that the sub DVPs entail the proof of the full capability.

"Onion" distribution

- Could be based on the distributed "onions" described earlier. Distribute two (or more) pairs.
- Need to finalise details of the distribution protocol.
- At what point does the system commit to the ballot forms?
- Voter could contribute their own entropy: e.g., re-encrypt the onion.

Quantum Prêt à Voter

- Even more speculative…
- Start with supervised?
- Use random, entangled quantum onions?
- Distributed creation? Mirror pre-mix?
- One part with voter, the other with the voting service.
- Measurement collapses these to give (linked) classical values.
- Decryption protocol to change LH quonion into the candidate list.
- Each voter gets n(>1) such quonions and does a cut and choose.
- Can we arrange for the action of marking the ballot to destroy the candidate list information?

Quantum opportunities

- Encrypted frame of reference = photon polarisation?
- Erasure of information, in particular, candidate list.
- Enforcing audit constraints: reveal only one of two "conjugate" values.
- On-demand creation or revealing of crypto information.
- Secure sources of entropy.
- Quantum tabulation?
- Unconditional privacy!?

Conclusions

- Voting systems a dynamic area of research.
- Lots of open questions!
- Highly socio-technical in nature.

Open questions

- Exact status of impossibility results: unconditional secrecy and accuracy cannot be achieved simultaneously.
- Unconditional privacy (Jeroen's scheme)?

Future work

- On the current model:
  - Determine exact requirements.
  - Formal analysis and proofs.
  - Construct threat and trust models.
  - Investigate error handling and recovery strategies.
  - Develop a full, socio-technical systems analysis.
  - Develop prototypes and run trials, e.g., e-voting games!
  - Investigate public understanding, acceptance and trust.

Future work

- Beyond the current scheme:
  - Finalise remote, coercion resistant version (using "capabilities").
  - Re-encryption mixes for general electoral methods.
  - Use of Paillier and other primitives?
  - Establish minimal assumptions.
  - Alternative sources of seed entropy: Voters, optical fibres in the paper, quantum…?
  - Alternative robust mixes.
  - Quantum variants.
  - Unconditional schemes?

References

- David Chaum, Secret-Ballot receipts: True Voter-Verifiable Elections, IEEE Security and Privacy Journal, 2(1): 38-47, Jan/Feb 2004.
- P Y A Ryan, "E-voting", presentation to the Caltech/MIT workshop on voting technology, MIT Boston 1-2 October 2004.
- P Y A Ryan, "A Variant of the Chaum Voter-verifiable Election scheme", WITS, 10-11 January 2005 Long Beach Ca.
- D Chaum, P Y A Ryan, S A Schneider, "A Practical, Voter-Verifiable Election Scheme", Newcastle TR 880 December 2004, Proceedings ESORICS 2005, LNCS 3679.
- B Randell, P Y A Ryan, "Trust and Voting Technology", NCL CS Tech Report 911, June 2005, to appear IEEE Security and Privacy Magazine.
- P Y A Ryan, T Peacock, "Prêt à Voter, A Systems Perspective", NCL CS Tech Report 929, September 2005, submitted to ESORICS 2006.
- P Y A Ryan and Steve A Schneider, "Prêt à Voter with re-encryption mixes", Newcastle CS TR 956, April 2006, submitted to ESORICS 2006.
- Clarkson and Myers, "Coercion-resistant Remote Voting using Decryption Mixes", at FEE 2005. http://www.win.tue.nl/~berry/fee2005/
- P Y A Ryan, "Spooky voting at a distance" CalTech Workshop on Classical and Quantum Information Assurance, December 2005, http://www.cpi.caltech.edu/quantum-security/

References II

- M. Jakobsson and A. Juels and Ronald Rivest, "Making Mix Nets Robust for Electronic Voting by Randomized Partial Checking", USENIX Security Symposium 2002, pp 339-353“.
- C. Karlof and N. Sastry and D. Wagner, "Cryptographic Voting Protocols: A Systems Perspective“, USENIX Security Symposium", LCNS 3444, pp 186-200“, Springer-Verlag 2005.
- A. Neff, "A verifiable secret shuffle and its application to e-voting", Conference on Computer and Communications Security, ACM, pp 116-125, 2001
- A. Neff, "Practical high certainty intent verification for encrypted votes“, http://www.votehere.net/documentation/vhti, 2004
- A. Neff, "Verifiable mixing(shuffling) of El-Gamal pairs", http://www.votehere.net/documentation/vhti, 2004
- P.Y.A. Ryan and S. A. Schneider, "Prêt à Voter with Re-encryption Mixes", Newcastle Tech Report CS-TR-956 April 2006

Announcement

Workshop On Trustworthy Elections
(WOTE 2006)
Robinson College, Cambridge, United Kingdom
June 29 - June 30, 2006

http://www.win.tue.nl/~berry/wote2006/

Announcement and Call for Contributions

Held in conjunction with the:
6th Workshop on Privacy Enhancing Technologies,
Robinson College, Cambridge, United Kingdom
June 28 - June 30, 2006
(http://petworkshop.org/2006/).

A Practical Voter-Verifiable Election Scheme

David Chaum[1], Peter Y A Ryan[2], Steve Schneider[3]

[1] Votegrity
[2] School of Computing Science, University of Newcastle
[3] Department of Computing, University of Surrey

Abstract. We present an election scheme designed to allow voters to verify that their vote is accurately included in the count. The scheme provides a high degree of transparency whilst ensuring the secrecy of votes. Assurance is derived from close auditing of all the steps of the vote recording and counting process with minimal dependence on the system components. Thus, assurance arises from verification of the election rather than having to place trust in the correct behaviour of components of the voting system. The scheme also seeks to make the voter interface as familiar as possible.

1 Introduction

Since the dawn of democracy, it has been recognised that the process of recording and counting votes could be the target of attempts at corruption. The Ancient Greeks investigated the use of (primitive) technological devices to provide trustworthy voting systems and avoid the need to trust voting officials [1]. The challenge is to provide voters with complete confidence that their vote will be accurately recorded and counted whilst at the same time guaranteeing the secrecy of their vote.

Most traditional approaches to this problem involve placing significant trust in the technology, mechanisms or processes used to process votes. Thus, for the traditional paper ballot, the handling of the ballot boxes and counting process must be trusted, i.e., the boxes must not be lost or manipulated and that the counting process is accurate. Various observers are introduced to the process which helps to spread the dependence on the technology but does not eliminate it.

With many of the touch screen devices widely used in the recent US presidential elections, the voter at best gets some form of acknowledgement of the way she casts her vote. After that, she can only trust in the assurances of the manufacturers and certifiers that her vote will be accurately included in the final tally.

By contrast, in [3], Chaum presents a digital voting scheme that enables voter verification, i.e., provides each voter with a means to assure themselves that her vote has been accurately included in the vote tally. This scheme combines a number of cryptographic techniques and primitives to provide a high degree of transparency whilst at the same time preserving ballot secrecy. Rather than having to place trust in the components to perform correctly, steps of the vote recording and tallying process are closely monitored to detect any malfunction or corruption.

The key elements in voter-verification are:

- when a voter casts her vote in a booth, she gets a receipt showing her vote in encrypted form.
- a voter confirms in the booth that her intended vote is correctly encoded in the receipt. The vote cannot be read subsequently outside the booth.
- a number of tellers perform anonymising mixes and decryption on the batch of encrypted ballot receipts. The decrypted votes emerge at the end of this process, with all links between the original receipts and the final decrypted values lost in the multiple mixes. Intermediate steps of the tellers processing are posted to a bulletin board, which might be published via the web for example.
- random checks are performed on all steps of the process to ensure that, with high probability, any attempt to corrupt vote capture and counting will be detected.

The point of the encrypted receipt is to provide the voter with a means to check that her ballot is entered into the tallying process and, if her receipt has not been included, to prove this to a third party. The fact that her vote is in encrypted form ensures that there is no way for a third party to know which way she voted. A voter can visit the bulletin board and check that her (encrypted) ballot receipt has been correctly posted. The tellers process these posted receipts and there are mechanisms in place to ensure that all posted receipts are entered into the tallying process.

The anonymising mixes performed by the tellers ensure that there is no link between the encrypted ballot receipt and the decrypted version that is finally output by the tallying process.

The design philosophy is to minimise trust in components. The approach is to strive for maximal transparency of the whole vote casting, recording and counting process, consistent with maintaining ballot secrecy. Thus, the integrity of the ballot forms and the correctness of the tellers' transformations are closely audited. The encryption of the voter's choice on the receipt is performed in the booth, is transparent, and does not depend on the intercession of any hardware or software devices that might be susceptible to failure or corruption.

2 Prêt à Voter

The original scheme of [3] uses visual cryptography to encrypt the receipts and perform the decryption in the booth. The scheme presented here uses a more conventional representation of the vote, i.e., ballot forms with the candidates or voting options listed in one column, and the voter choices entered in an adjacent column. As a result, the scheme is easier to understand and implement.

An earlier paper, [7], introduced the idea of encoding the vote in terms of two aligned columns, one carrying the candidate or option list in randomised order (independent for each ballot form) whilst the other strip carries the voter choice. In this version, the voter was invited to choose which of the left and right columns to retain as the receipt. This introduced a certain asymmetry with both cryptographic and psychological implications.

In this paper we introduce some further innovations: we use ballot forms that are generated and printed in advance. As before, these have two columns, one of which shows the candidate list in scrambled order. Now however, rather than choosing between columns as previously, the voter will always discard the left hand column containing the candidate list, and submit the right hand column containing the marked vote. This avoids the asymmetry in the choice between left and right columns of the previous scheme.

A further innovation is to use the tellers in an oracle mode to enable the checks on the well-formedness of the ballot forms. This is in addition to the previous use of the tellers to perform the anonymising mix during the tallying phase. Besides allowing independent auditing authorities to perform random checks, this also opens up the possibility of novel checking modes, including enabling the voters to cast a dummy vote and have the tellers return the decryption to them as a check on the construction of the ballot forms.

The scheme presented here provides a number of appealing innovations, notably:

- Voters should find the vote casting process entirely familiar.
- Cryptographic commitments are generated before voter choices are known.
- Voter checks on the correct construction of the ballot forms are supplemented by random audits. Thus, voters are able to contribute to the verification of the vote capture process but the assurance of the scheme is not dependent on the voters being sufficiently diligent.
- Checks on the correct construction of the ballot forms are performed before votes are cast, thus simplifying the recovery strategies.
- The vote recording devices in the voting booths do not learn the voters' choices. This neatly avoids any threats of such devices leaking the voters' choices.
- The scheme is conceptually much simpler than others that have been proposed, thus easing its implementation and increasing the chances of voter acceptance.
- The current scheme shows considerable flexibility, suggesting that it could readily be adapted to different electoral requirements.

3 The Election Setup

A number of tellers are appointed. Each is assigned or creates *two* secret/public key pairs. The use of two keys per teller is a technical convenience arising from the audit process that will become clear later. These public keys are publicised and certified.

An authority creates a large number of ballot forms, significantly more than required for the electorate. These will have a familiar appearance: a left hand column listing the candidates or options and a right hand column into which the voter can insert her selection. This might just be an X in one cell for a single choice election or a ranking for a Single Transferable Vote (STV) system. Thus, for a four candidate race, a typical ballot form might look like:

Nihilist	
Buddhist	
Anarchist	
Alchemist	
	$7rJ94K$

However, the order in which the candidates are listed will be randomised for each ballot form, that is, for any given ballot, the candidate order shown should be unpredictable. The random looking value at the bottom of the right hand column (which we call an 'onion' for reasons that will become apparent in Section 5) contains the information from which the candidate ordering can be reconstructed, buried cryptographically under the public keys of the tellers. The precise construction of the onions will be described in Section 5.2.

The exact details of the voting procedure can be varied according to the nature of the election and according to the perceived nature of threats to which the system is exposed. For simplicity of presentation we outline one simple procedure. Other procedures are possible and indeed one of the advantages of this scheme is that it appears to be significantly more flexible than previous variants.

4 An Example

The scheme is best introduced by way of a simple example. We will give a more formal and general description later. Suppose for simplicity that we are dealing with a simple election system in which each voter selects exactly one candidate and the winner will be the candidate who garners the most votes. This allows us to present the example using simple cyclic shifts of the candidate ordering. Generalisations to deal with options to select more than one candidate or to rank them, etc. are straightforward and discussed later. Clearly, a "none of the above" option could also be included.

4.1 Processing votes

Suppose that there are four candidates and these are given a base ordering:

Anarchist
Alchemist
Nihilist
Buddhist

Since we are considering only cyclic shifts in this example, there are four possible candidate lists, corresponding to the four possible offsets, 0 to 3, from the base candidate list. The generation of the random offsets and cryptographic values will be described in detail later.

For convenience of the mathematical manipulations, we also adopt a canonical numbering convention for the candidates from 0 to 3 as indicated. Thus a vote for Anarchist will be represented as 0, for Alchemist as 1 etc. This numerical representation is purely for the machine manipulations and need not trouble the voter.

Consider the following ballot form:

Buddhist	
Anarchist	
Alchemist	
Nihilist	
	Qqkr3c

This has an offset of 1. Thus the onion—*Qqkr3c*—encodes the value 1. Suppose the system is to process a vote for Nihilist. This would be represented by a mark in the Nihilist box:

Buddhist	
Anarchist	
Alchemist	
Nihilist	X
	Qqkr3c

Once the voter has marked her choice, the left hand column that shows the candidate ordering is detached and destroyed, to leave a ballot receipt of the form:

X
Qqkr3c

Such right hand strips showing the position of an X and an onion value constitute the ballot receipts.

This is now fed into the voting device, presumably an optical reader, which transmits the information on the strip, the position of the X (as a numerical value 0, 1, 2 or 3) and the value of the onion, to the tellers. The tellers use their secret keys to perform the decryption of the onion (see later), and generate the decrypted vote value corresponding to the vote in the base ordering. In this case the decryption process yields the offset 1, so the vote value is the position of the vote (3) with the appropriate offset removed, yielding candidate $3 - 1 = 2$:

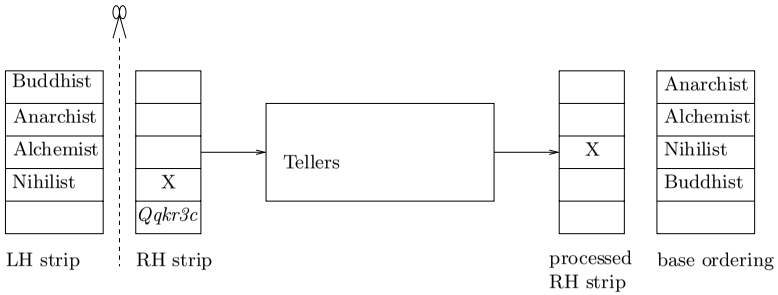


Fig. 1. Processing a vote

Nihilist. This process is illustrated in Figure 1. A more detailed description will be provided later.

4.2 Casting the vote

Our voter, Anne, first authenticates herself and registers at the polling station. She is invited to select, at random, a ballot form. She now enters a booth with her ballot form and marks her X in the usual way. Suppose that she decides to vote for the "Buddhist" candidate:

Nihilist	
Buddhist	X
Anarchist	
Alchemist	
	*e*1*rg*38

She now removes the left hand strip (for shredding), and feeds the right hand strip into the voting device. This checks that the ballot strip is unused and reads the position of Anne's X, and the value of the onion. The device marks the strip as having been used to cast a vote and returns it to Anne for her to retain as the ballot receipt.

X
*e*1*rg*38

Note that the vote recording device does not learn which way Anne voted. Its role is merely to read the information on Anne's receipt and relay it to the the tellers via the bulletin board. This is a significant advantage of this scheme over

many other schemes where the voting device necessarily learns the voter's choice, raising the possibility that the device could somehow leak this information.

The device transmits its digital record of the receipt to a central server for subsequent posting to the bulletin board once the election has closed. Anne will later be able to visit the bulletin board and confirm that her receipt is correctly posted and hence that it is correctly entered into the tallying process. The tallying process is deliberately constructed to hide the link between specific ballot receipts and the resulting decrypted votes, in order to provide voter anonymity. Thus Anne cannot directly link her input vote strip to any specific resulting vote, and so she cannot directly verify that her vote has been correctly decrypted. However, the fact that the votes are all correctly processed can be checked to a high degree of confidence, which provides Anne with the assurance that her vote will be decrypted correctly.

Observe that Anne's receipt alone does not reveal which way she voted. Unless the tellers are involved, this can only be determined if the left hand strip (now destroyed), that carries the candidate ordering, is aligned against it. Only the totality of the tellers, acting in consort, using their collection of secret keys are able to extract the seed information and so reconstruct the candidate ordering for that ballot form.

5 Construction of the Ballot Forms

The above description should have provided the reader with the key intuitions. We now give some of the mathematical details.

5.1 Construction of the Cryptographic Seeds and Offsets

For each ballot form, the authority will generate a unique, random seed. Suppose that there are k tellers (numbered 0 to $k-1$), then this seed will be made up of a sequence of $2k$ values that we will call the germs:

$$seed := g_0, g_1, g_2 \ldots g_{2k-1}$$

Each of these germs should be drawn from some modest size field, perhaps 2^{32}. Thus, for $k = 3$ say, the seed values will then range over 2^{192}. These numbers can be adjusted to achieve whatever cryptographic strength is required.

The offset for the candidate list is now calculated from these germ values as follows. First a publicly known cryptographic hash function is applied to each of the germs and the result taken modulo v, where v is the size of the candidate list:

$$d_i := hash(g_i) \pmod{v} \quad i = 0, 1, 2, ..., 2k-1$$

The cyclic offset θ that will be applied to the candidate list on this form is now computed as the $(mod\ \ v)$ sum of these values:

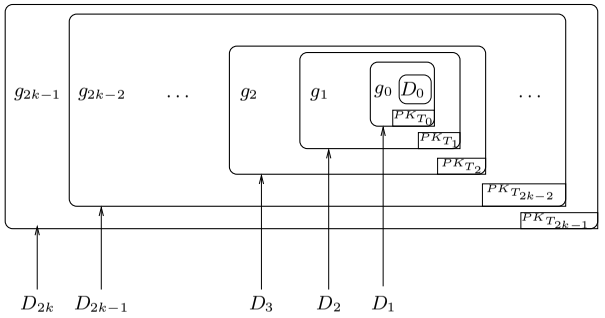


Fig. 2. An onion

$$\theta := (\textstyle\sum_{i=0}^{2k-1} d_i) mod\ \ v$$

5.2 Construction of the Onions

In order to facilitate auditing of the tellers whilst preserving anonymity of the voters (see [3] or [2] for more details), each teller performs two Chaum mixes and, accordingly, has two independent secret/public key pairs assigned to it. Teller i will have public keys $PK_{T_{2i}}$ and $PK_{T_{2i+1}}$, and corresponding secret keys. The onion is formed by nested encryption of the germs under these public keys, and is given by:

$$\{g_{2k-1}, \{g_{2k-2}, \{\ldots, \{g_1, \{g_0, D_0\}_{PK_{T_0}}\}_{PK_{T_1}} \cdots\}_{PK_{T_{2k-3}}}\}_{PK_{T_{2k-2}}}\}_{PK_{T_{2k-1}}}$$

We introduce a little more notation to denote the intermediate layers of the onions. D_0 is a random, nonce-like value, unique to each onion. The subsequent layers are defined as follows:

$$D_{i+1} := \{g_i, D_i\}_{PK_{T_i}}$$
$$Onion := D_{2k}$$

Where i ranges over $\{0, 1, ..., 2k-1\}$. The construction of an onion is pictured in Figure 2.

6 The Role of the Tellers

The primary role of the tellers is to perform an anonymising mix and decryption on the batch of encrypted ballot receipts posted to the bulletin board. This

ensures that the decrypted votes that emerge at the end of mix cannot be linked back to the encrypted receipts that are input to the process. Aside from some minor differences, the role of the tellers and the auditors are essentially as in the Chaum original. For completeness we give a brief overview here. More detailed descriptions can be found in [3] or [2].

The first, left hand column, of the bulletin board shows the receipts in exactly the same form as the printed receipts held by the voters. A voter can check this column to verify that her receipt has been accurately posted. An easy way to do this would be to search on the string representing the onion value and check that the X appears in the correct box, i.e., as shown on the voter's receipt.

The information in the first, left hand column of the bulletin board is then passed to the first teller, $Teller_{k-1}$, for processing. There is no shuffling of the information when it is passed to the teller. The position of the X on the voting slip is encoded as an integer r, and the correctness of this encoding can be simply and publicly verified.

The tellers will subsequently manipulate the numerical representations of the receipts, i.e., pairs of the form (r_i, D_i), where r_i is between 0 nd $v-1$, and D_i is an ith level onion. The initial value of r_{2k} is the encoding of the position of the X as originally placed by Anne on her receipt.

Each column (apart from the first, which contains the actual receipts) shows only the simplified, digital representation: a pair (r_{2k}, D_{2k}) consisting of a value r from Z_v and the value D of the onion layer.

Each teller accepts an input column of votes (r, D) from the previous teller, and then carries out two manipulations, to produce a middle column of votes and an output column of votes. The output column produced by the teller is then passed to the next teller in the chain.

Thus for each of the (r_{2i}, D_{2i}) pairs in the batch in the input column, $Teller_{i-1}$ will:

- apply its first secret key, $SK_{T_{2i-1}}$ to strip off the outer layer of the onion D_{2i} to reveal the enclosed germ g_{2i-1} and the enclosed onion D_{2i-1}.

$$g_{2i-1}, D_{2i-1} = \{D_{2i}\}_{SK_{T_{2i-1}}}$$

- apply the hash function to the germ value and take the result (mod v) to recover d_{2i-1}:

$$d_{2i-1} = hash(g_{2i-1}) \pmod{v}$$

- subtract d_{2i-1} from r_{2i} (mod v) to obtain a new r value r_{2i-1}:

$$r_{2i-1} = r_{2i} - d_{2i-1} \pmod{v}$$

- form the new pair (r_{2i-1}, D_{2i-1})

Having completed these transformations on all the pairs in the initial batch as posted in its input column, the teller applies a secret shuffle to the resulting, transformed pairs and posts the resulting (transformed and shuffled) pairs to its middle column on the bulletin board.

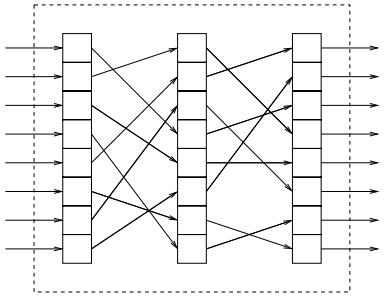


Fig. 3. A teller

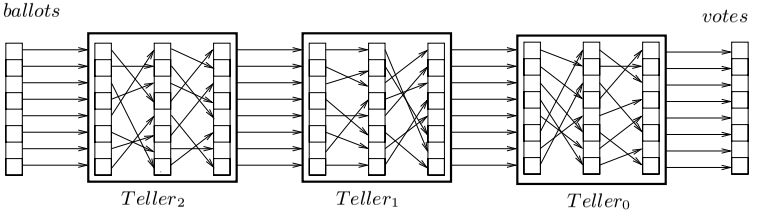


Fig. 4. Three tellers anonymising mix

$Teller_{i-1}$ now repeats this process on the contents of the middle column using its second secret key, $SK_{T_{2i-2}}$ to obtain a new set of (r_{2i-2}, D_{2i-2}) pairs. It will apply a second secret shuffle, independent of the previous one, to this batch of new pairs. The resulting transformed and shuffled (r_{2i-2}, D_{2i-2}) pairs are now posted to the output column on the bulletin board, and passed on to the next teller, $Teller_{i-2}$. This process is illustrated in Figure 3.

This process is repeated by all the tellers in sequence, as illustrated in Figure 4 for a sequence of three tellers. The value of any of the intermediate r values is thus given by:

$$r_{2k-i} = r_{2k} - \Sigma_{j=1}^{i} d_{2k-i} \pmod{v}$$

When the last teller performs the final transformation it outputs a batch of pairs which comprise a final r value, r_0, and the inner onion value D_0. The final r_0 values are the values of the original votes in the canonical, base ordering. Figure 5 illustrates the effect of the process on a single vote.

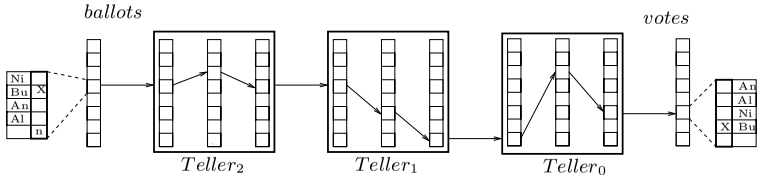


Fig. 5. A vote processed by three tellers

To see this, observe that the candidate list on each form is shifted by the $(mod\ \ v)$ sum of the d values, i.e., θ. Thus the initial r value is the candidate value plus θ modulo v. For each ballot pair, the tellers will have subtracted out the d values from the initial r value, thus cancelling the original shift of the candidate list and so recovering the original candidate value. Thus:

$$r_0 = r_{2k} - \Sigma_{j=1}^{2k} d_{2k-i} \ \ (mod\ \ v) = r_{2k} - \theta \ \ (mod\ \ v)$$

Consider the example of Anne's vote again (illustrated in Figure 5). The form she used to cast her vote had an offset of 2 and her X was in the second box, value 1. Hence the initial value of r_{2k} was 1 in her case. The tellers will in effect compute:

$$r_0 = r_{2k} - \sum_{j=1}^{2k} d_i (mod\ \ 4) = 1 - 2 (mod\ \ 4) = 3$$

Thus the final r value $r_0 = 3$ does indeed translate to a vote for "Buddhist" in the base ordering. The encryption of the vote can thus be thought of as a (co-variant) transformation of the frame of reference, decryption to the corresponding (contra-variant) transformation.

The overall effect then, is to have posted on the bulletin board, in the left hand column, the batch of initial receipts as posted by the voting devices. In the right hand column we will have the fully decrypted votes. In between there will be a set of columns with the intermediate, partially decrypted (r, D) pairs. Each column will be some secret permutation and decryption of the previous one, and the permutation will not be published. This is illustrated in Figure 6. Note that the decryptions at each mix stage prevent the permutation being reconstructed by simple matching of onions or r values.

The purpose of using the hash of the germ values buried in the onion layers to transform the r values is to foil guessing attacks on the mixes. Without these hashes it would be possible to guess links through the mixes and check the guess by performing the appropriate computations (with the knowledge of the tellers' public keys). With the hash functions, these checks would require the computation of pre-images of the hashes, thus rendering such attacks intractable.

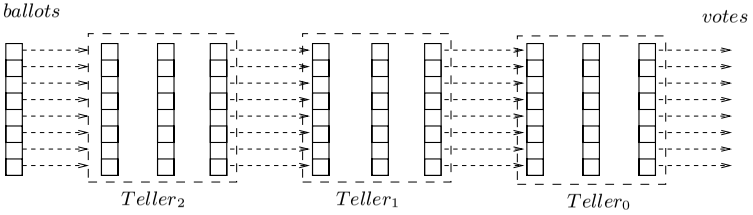


Fig. 6. Information posted by the sequence of three tellers

We will see later that, for audited links the tellers are required to reveal not only the link but also the associated germ. The computations performed by the auditors are thus perfectly tractable.

Assuming that all the tellers perform their transformations correctly, there will be a one-to-one correspondence between the elements of each column and the next. The exact correspondence, namely which (r, D) pair in one column corresponds to which pair in the next column, will be hidden and known only to the teller who performed the transformation between those columns. Thus, the receipts will have undergone multiple, secret shuffles between the first column as posted by the voting devices and the final decrypted column. This ensures that no voter can be linked to her vote, so ensuring ballot secrecy.

The fact that several tellers are used gives several layers of defence with respect to voter privacy: even if several of the tellers, but not all, are compromised, the linkage of a voter with her vote will remain secret.

The decrypted votes are posted in the final column so the overall count can be verified by anyone.

7 Auditing the Process

The description so far has assumed that all the steps of the vote casting, recording and counting are performed correctly, to specification. In fact, we want to avoid having to place such trust in the components of the scheme: the authority that generates the ballot forms, the device that records and transmits the receipt values and the tellers that perform the mixes and decryptions. In this section we identify the failure modes and corresponding counter-measures.

We assume for the purposes of this paper that measures are taken to prevent failures of the surrounding system, for example, in the maintenance of the electoral role, voter authentication etc. Here we concentrate on the failure modes of the cryptographic core of the scheme. With respect to the accuracy requirement there are three failure modes of the technical core of the scheme:

– Incorrectly constructed ballot forms, i.e., forms for which the cryptographic seed information buried in the onion does not correspond to the candidate order printed on the form.

– Incorrect recording of the values on the receipts and/or transmission to the Bulletin Board for tabulation.
– Errors or corruption in the transformations by the tellers on the ballot pairs.

Any of the failure modes could lead to vote values being incorrectly decrypted, i.e., resulting in decrypted vote values different from those intended by the voters. We now detail the checking processes that are employed to detect, with high probability, any such failures. We start with role of the authority tasked with generating the ballot forms.

7.1 Checking on the authority

Suppose that a suitable authority has generated and distributed a large number of printed ballot forms to the polling stations. Independent auditors will be appointed whose task is to subject a random sampling of these ballots to well-formedness checks. These checks are designed to establish that the seeds buried cryptographically in the onions correctly correspond to the candidate list that appears on the form, given the declared public keys of the tellers. The auditors might also be tasked with checking the quality of the entropy used in the creation of the ballot forms.

Further random audits could also be performed during the election. Indeed, once the election has closed, left-over forms could also be routinely audited as well.

In addition to the checks performed by the auditors, mechanisms can also be provided to enable the voters to perform checks on the integrity of the ballot forms of their own, as detailed shortly. Thus, the voters are empowered to contribute to the verification of the election. First we describe the auditor checks, then we describe those that could be made available to the voters.

Auditing the Ballot Forms A set of independent auditing authorities are appointed. These should be chosen in such a way as to minimise the chance of collusion. They might, for example, be drawn from civil liberties groups, the political parties etc. Each would be invited to make a random sampling of, say, 5% of the ballot forms generated by the authority.

To check the construction of the forms, some access to the cryptographic seeds is required. This could be achieved by requiring the authority to store the seeds along with their association with the onion values on the forms. However, the storing and selective release of such crypto material is potentially rather delicate and fragile. A novel and more elegant and robust approach is to use the tellers to strip off the layers of encryption for forms selected for audit and reveal the seed material.

Once the seed material for a ballot form selected for audit has been revealed, the form's integrity can be verified by recomputing the offset and onion value. If these match those printed on the form then it is safe to conclude that the form was indeed correctly constructed. Note that these calculations can be performed and verified by anyone, since the public keys of the tellers and the crypto hash functions are all public knowledge, More precisely, to check a ballot form, the following actions are performed:

– the auditor sends a digital copy of the onion on the form to the tellers.
– the tellers strip off the layers of encryption using their private keys to reveal the germs.
– the sequence of germ values are returned to the auditor.
– given the germ values, and knowing the public keys of the tellers, the auditors are able to reconstruct the value of the onion and can check that this agrees with the value printed on the form.
– they now recompute the offset value as the $(mod\ \ v)$ sum of the hashes of the germs.
– they can now check that the offset applied to the candidate list shown on the form agrees with the value obtained above.

If these checks are successful, it is safe to conclude that the ballot form in question was correctly constructed. Checked ballot forms, for which the seed has been revealed, are then discarded. If a random sampling of a significant proportion of forms all pass the checks, then it is safe to conclude that all the forms are correctly formed. The statistical calculations of the levels of confidence afforded by such random sampling are straightforward and, of course, the sampling rates can be adjusted to achieve whatever confidence levels are required.

Note further, that the algorithms for these checks are publicly known, so in principle, anyone could construct such a checker and make it freely available. Similarly anyone could examine such a checker to establish that it was performing correctly. Note also that any interested party could volunteer to perform some of the auditing. Thus, for example, the Electoral Reform Society could act as auditors. Representatives of the political parties could act as auditors. Furthermore, any results produced by an auditor can be double checked by independent parties.

Voter Checks on Ballot Form Integrity In addition to the integrity checks performed by the auditors described above, the scheme also allows for checks on ballot form integrity to be performed by the voters themselves. This empowers the voters to contribute to the dependability of the election outcome, a sort of dependability for the people, by the people!.

The technique of using the tellers as an oracle during the voting phase suggests a number of alternative modes for checking the integrity of the ballot forms. These do not involve the revealing of the seed information.

1. Single dummy vote.
2. Multiple or ranked dummy vote.
3. Given the onion value, the tellers return the candidate ordering.

In the first, the voter would cast a dummy vote in exactly the same way that she will later cast her real vote in the booth, except that in this case the dummy

vote would probably be cast in the presence of voting officials. Thus, she could put a cross against a random selection and send the receipt off to the tellers. They would decrypt the onion and return what they believe was the vote cast. If the onion was correctly constructed, this should of course agree with the dummy vote selected.

This has interesting psychological implications: assuming that the check succeeds, it should provide the voter with some assurance that when she comes to cast her real vote, it will also be correctly counted. On the other hand it might undermine her confidence that the secrecy of her vote will be assured.

Such a single dummy vote provides a rather weak check on the ballot form construction, probing only part of the construction. The second mode seeks to rectify this: by allowing the voter to cast several dummy votes, either in series or in parallel by making a ranking selection. In the latter case, given the receipt, the tellers should return what they believe to be the candidate ranking chosen by the voter. This provides a more complete check on the construction of the ballot form. Both of these suffer the drawback that the voter is expected to make random choices in the presence of officials.

The third mode is perhaps the most satisfactory. It provides a complete check on the ballot form but does not require the voter to make any random selections. Here, given only the onion value, the tellers should return what they believe to be the candidate ordering as shown on the ballot form.

We note that, in contrast to the auditor checking mode, these three modes are vulnerable to collusion attacks. If the authority that generated the forms is in collusion with one of the tellers there is the possibility of corrupting forms without detection by these modes. For example, the authority could flip a pair of candidates on the ballot forms. The colluding teller performs the corresponding flip during the checking phase, but not during the tallying phase.

The auditor checking mode is not vulnerable to such collusions and so is more rigorous. It therefore appears to be more suitable for the auditing authorities. It could also be made available to voters, but it seems less intuitive and so perhaps less reassuring to the voters. The psychological aspects of these checking modes from a voter perspective will be investigated in future work.

Thus, a possible voting procedure might be to allow a voter when she registers at the polling station to select a pair of ballot forms at random and nominate one for checking. This could then be checked in the presence of officials using, say, the third mode described above. Assuming that the check goes through okay, the checked form is discarded and the voter can proceed to the booth with her "real" ballot form. If any check fails, she should notify an official who should then investigate and diagnose the source of the error. We will discuss the error handling and recovery strategies later.

As noted earlier, care has to be taken in assessing the assurance provided by the voter checks as these are vulnerable to collusion attacks. Various countermeasures could be adopted to limit the likelihood of such collusions going undetected. One possibility is to use an l out of k threshold scheme for the onion encryptions. The l cardinality subsets of the k tellers could then be chosen randomly for each dummy voting request. If the colluding tellers were omitted when a corrupted dummy vote was decrypted, an error would be flagged. In any case, the random checks by the auditors would catch such manipulated ballot forms as these are not vulnerable to such collusion attacks.

The tellers might return incorrect germ values but this will of course throw up a mismatch between the recomputed onion value and the value on the form. It might be that a teller malfunctions, or is loaded with the wrong keys. In this case the checks serve a useful role in debugging such configuration errors.

Note that the encryptions are all bijective, hence the germ values are uniquely determined by the onion value. The tellers cannot therefore find alternative germ values that would give the same onion value but a different offset.

Together, these checks ensure that if a malicious or corrupted authority tried to corrupt votes by providing a candidate ordering that does not correspond to the seed information buried in the onion, they stand a high chance of being detected. The chance of corruption going undetected falls off exponentially with the number of ballots they try to corrupt.

We stress that all the checks detailed here serve purely to probe the well-formedness of the ballot forms, i.e., serve to detect any failure of the candidate orderings on the forms to correspond to the information buried in the onions. These checks do not provide any detection of corruption during the tallying phase. A form that is correctly constructed in this sense will correctly capture the voter's intention. Of course, this does not of itself ensure that the vote will ultimately be correctly decrypted. For this we need additional mechanisms to ensure that all ballot receipts will be correctly recorded, transmitted and decrypted. These we address next.

8 Checking on the vote recording devices

We need to ensure that ballot receipts are faithfully recorded, transmitted and entered into the tallying process. This is where the bulletin board comes into play. Once voting has closed, all ballot receipts are posted to the bulletin board. The material posted to the bulletin board will be publicly available in read-only mode. Thus any voter can visit the board and confirm that her receipt appears correctly in the input column.

If her receipt does not appear, or appears in corrupted form (in particular, if the position of the X is incorrect), this should be reported. The voter has her receipt to prove to an official that her receipt does not appear correctly. In practice all ballot forms would be printed with anti-counterfeiting measures and would have been stamped and digitally signed by the device in the booth when the vote was cast to prevent attempts to fake receipts.

Assuming that voters are reasonably diligent in performing these checks, any failures to faithfully post receipts to the bulletin board, and hence to enter them into the tallying, should be detected. Precautions would also be needed to prevent anyone inserting additional, invalid receipts. One simple precaution would be to ensure that the number of posted receipts matched the number of

cast ballots. The digital signatures applied by the voting devices could also be used to help prevent fake ballots being introduced.

A further possible enhancement is for the device in the booth to produce a paper copy of the ballot receipt. This copy is posted into a locked and sealed audit box (perhaps after being viewed under glass and confirmed by the voter in the manner of the 'Mercuri method' [5]). Now, independent auditors can perform checks of the correspondence between published receipts and the paper audit trails stored in the audit boxes. This serves to supplement the checks performed by a voter on the appearance of her receipt in the published list. This last enhancement has similarities to the Voter Verifiable Paper Audit Trail (VVPAT [5]) and has the advantage that the checks on ballot receipts on the bulletin board performed by the voters are supplemented by auditor checks. The assurance of the scheme is thus less dependent on the diligence of the voters in checking the appearance of their receipts in the published list.

9 Checking on the Tellers

The checks described above should ensure that voters' intentions are correctly encrypted in the ballot receipts and that all receipts are correctly entered in the tabulation process. Now we must ensure that all the receipts are accurately decrypted. For this, we must ensure that all the transformations performed on the receipts by the tellers during the anonymising mixes are correct.

As in the original Chaum scheme, the auditing of the tellers is based on the notion of partial random checking proposed in [4]. This takes place after the teller processing has finished, and is applied to the information committed to by the tellers on the bulletin board.

For each teller an auditing authority goes down the middle column and randomly assigns R or L to each (r, D) pair. For pairs assigned an R, the auditor requires the teller to reveal the outgoing link (to the right) to the corresponding pair in the next column along with the corresponding germ value. For all pairs assigned an L, the auditor requires the teller to reveal the incoming link (from the left) along with the germ value.

This way of selecting links ensures that, for any given teller, no complete route across the two shuffles performed by that teller are revealed by the audit process. Hence no ballot receipt can be traced across the two mixes performed by any given teller. Each ballot transformation has a 50/50 chance of being audited.

This is illustrated in Figure 7, with the selected links included. The remaining links are not revealed.

For each teller the auditor performs such a random audit. Given the property that there are no full links revealed across any teller's mixes, the L/R selection can be made quite independently for each teller. This is the rationale for making each teller perform two mixes.

Suppose that, for a revealed link, the pair has been transformed thus:

$$r_i, D_i \longrightarrow r_{i-1}, D_{i-1}$$

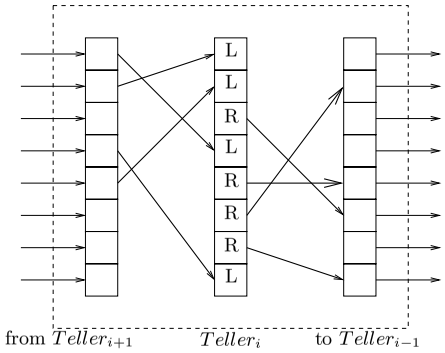


Fig. 7. Auditing $Teller_i$

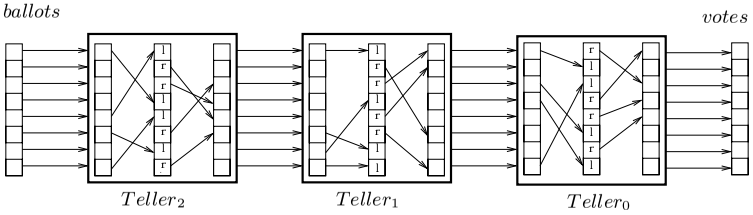


Fig. 8. Auditing the three tellers

Knowing this and the corresponding germ value g_{i-1} (which the teller is required to provide for each revealed link), it can be checked that the following hold:

$$D_i = \{g_{i-1}, D_{i-1}\}_{PK_{T_{i-1}}}$$

and

$$r_{i-1} = r_i - hash(g_{i-1}) (mod \ \ v)$$

If these equalities hold on a link we can conclude that the teller executed the correct transformation on this ballot pair. Some additional reasoning is required to show that it is not possible for a teller to perform a corrupted mix and be able to reveal false links in such a way as to pass any audit.

Figure 8 illustrates the audit across the sequence of three tellers.

10 Error Handling and Recovery Strategies

So far we have only described the checks that can be performed. A full description of the scheme requires detailing error handling and recovery modes. Due to lack of space we will not attempt to give an exhaustive description here.

Let us just consider the error handling strategy for a failed voter check. The first step for the official is to confirm that there is a real disagreement. Anne will have both parts of the dummy ballot form so she can prove which way she cast her dummy vote and she has the printout for the tellers. The official can thus establish that the problem is genuine and not just a case of voter error.

If the problem is real, the official should now run a further, auditor check: use the tellers as an oracle to extract the seed value and use this value to reconstruct the onion value and candidate list offset. If these values agree with those shown on the ballot, then it is fair to conclude that the form was correctly constructed by the authority. The error must then lie with the decryption of the vote performed by the tellers.

If this check fails, it can mean one of two things: the form was incorrectly constructed by the authority, or the form was perhaps actually correctly formed but the seed value returned by the tellers is incorrect.

Clearly, errors have to be diagnosed and collated. Strategies for dealing with patterns of errors must be specified. Thus, if a significant number of ballot forms were found to be malformed, doubt would be cast on the integrity of the authority charged with generating the forms. Note another pleasing feature of the scheme: any significant corruption on the part of the authority generating the ballot forms would almost certainly be detected by random audits before the election opens. Hence, this authority could be replaced before the election even starts.

A full description of error handling and recovery strategies is the topic of current research.

11 Generalising ballots

This paper has so far considered ballots that allow a vote against a single candidate. More generally, elections may allow votes or preferences to be cast against a number of candidates. In this case a right hand strip may contain a number of X's, or perhaps a list of numbers against candidates.

In this case, in order to avoid leaking information about votes, it is necessary to allow any permutation of the candidate list on the left hand strip, rather than just a cyclic permutation.

In order to achieve this, the germs could be used as keys for a cryptographic permutation function. The overall permutation applied to the candidate list as shown on the ballot form would then be a composition of the $2k$ separate permutations obtained from the $2k$ germs.

We use a publicly known hash function h that maps germs to permutations, so that $p_i = h(g_i)$ is a permutation of names on ballots. The overall permutation is given by the composition of the permutations for all the germs:

$$\pi = p_{2k-1} \circ p_{2k-1} \circ \ldots \circ p_0$$

(where $f \circ g(x) = f(g(x))$). If the base candidate ordering is $base$, then the candidate list on the ballot is given by $\pi(base)$. Thus a corresponding vote r on the right hand strip corresponds to a vote of $\pi^{-1}(r)$ against the base ordering.

The steps in the tellers take (r_{i+1}, D_{i+1}) to (r_i, D_i), where each step reverses one permutation comprising π. Here, the r values will encode either a ranking or an element of the power set of candidates as appropriate. The onion is unpeeled as previously to extract the associated seed g_i and the inner onion D_i. In this case the computation of r_i is given by:

$$r_i := (h(g_i))^{-1}(r_{i+1}) \qquad = \qquad p_i^{-1}(r_{i+1})$$

Given that the initial vote r provided to the tellers is r_{2k}, we obtain that

$$r_i = (p_i^{-1} \circ p_{i+1}^{-1} \circ \ldots \circ p_{2k-1}^{-1})(r_{2k})$$

and thus the final vote r_0 posted by $Teller_0$ is $\pi^{-1}(r)$, which is indeed the vote cast.

12 Related work and conclusions

A large number of cryptographic voting schemes have been proposed over the past 20 years or so. These use a variety of cryptographic techniques, ranging from blind signatures to cryptographic homomorphisms etc. The idea of providing the voter with an encrypted receipt goes back to the original scheme proposed by Chaum. Another scheme, that also uses encrypted receipts and has similar goals, is the VoteHere scheme of Adler and Neff, [6]. The cryptographic primitives used there are quite different from those of this paper and appear to be significantly more complex.

We have presented a new voter-verifiable election scheme based on the original Chaum scheme. This variant preserves the essential features of the original whilst sidestepping the complexity of the visual cryptography of the original. The presentation of the encoding on the vote is quite intuitive and familiar. A pleasing spin-off is that the randomisation of the candidate order counters any tendency to bias the voter choice that might arise from a fixed order.

The new scheme provides some interesting advantages over previous variants:

- The format of the ballot forms and the process of casting a vote is quite familiar.
- The cryptographic commitments are generated before the voter choices are revealed, even before the election period starts.
- The vote recording devices do not learn the voter choices. This avoids the possibility of such devices leaking this information.

- Voters get to perform their own checks on the correct construction of their dummy ballot forms. This should help instil confidence that their real votes will ultimately be correctly decrypted during the tallying process.
- The checking performed by the voters is supplemented by audits performed by various auditing agencies.
- The problem of storing and selectively revealing seed information is solved by the novel use of the tellers during the voting period as oracles to reveal the seeds for ballot forms used for auditing.
- Voters get to run their checks before casting their vote. This avoids some of the messiness in the recovery mechanisms of earlier variants when a voter discovers a mal-formed receipt after casting their vote.
- The initial auditing phase performed on the ballot forms forms should serve to weed out any corrupt authority even before the election opens.

Precautions need to be taken to prevent double voting. In particular, care needs to be taken to ensure that ballot forms used for checking cannot be reused to cast real votes. These details of such mechanisms will be discussed in a future paper.

Similarly, precautions are need to clearly separate the two functions of the tellers: the on-demand ballot form integrity checking function and the anonymising mix function. In particular it is essential to ensure that no ballot form that has been used to cast a "real" vote can be subsequently used in a checking mode. Various procedures can be envisaged to prevent this: appropriately marking a receipt that has been used to cast a vote and ensuring that it cannot be reused for either dummy or real voting. It would be satisfying to develop cryptographic mechanisms to enforce this.

For the purposes of illustration we have described how the scheme can be used for a single vote system, i.e., in which voters get to choose just one of a set of options or candidates. Where a voter can rank the candidates in order of preference (or indeed where she can vote for more than one candidate), full permutations in place of the simple cyclic shifts presented here. In practice, full permutations would probably be used even for single selection elections.

13 Future Directions

The destruction of the left hand strips of the ballot forms is essential to prevent both coercion and vote buying. An issue that requires careful consideration then is how to best enforce the destruction and ensure that it is not possible for the voter to exit the booth with both parts of the ballot form. Mechanical devices that enforce the destruction when the vote is cast are a possibility. Another interesting possibility is to ensure that plenty of dummy left hand strips are available in the booth, rather than trying to enforce destruction of this strip. If a voter is threatened with coercion she can simply select an appropriate strip that will keep the coercer happy.

Another issue is that, as presented, the scheme entails the authority knowing the association of all onions and candidate lists. Thus, if the authority were compromised, it could jeopardise the secrecy of the election. Various measures can be envisaged to counter or at least minimise this risk. Ballot forms could be generated in some distributed fashion using various sources of entropy. Alternatively, ballot forms could be generated and printed on demand. An intriguing possibility is to use entropy derived from the paper used to print the forms, for example using optical fibres stirred into the paper during manufacture. Ballot forms could be supplied in sealed envelopes to prevent the information being garnered in transit. The problem remains that there is still a point at which the onion and candidate list must be presented to the voter.

For the three voter checking modes, the germ values do not have to be revealed. This suggests the possibility of reusing a "dummy" ballot form to cast a real vote. This has the advantage that the form used for the real vote will itself have been tested. Ballot forms could come equipped with two onion values, both of which should yield the candidate ordering shown. One could be used for checking, the other to cast the real vote. This possibility may however open up vulnerabilities and would need to be subjected to careful analysis. This is the subject of current research.

This scheme would appear to be readily adapted to remote voting. The simplest adaption is to distribute ballot forms by post. Votes could then be cast by providing the onion value along with suitable indicators of the voter selection in the right hand column. Alternatively, protocols could be used for on-line, authenticated distribution of the crypto material. Of course, the threat of coercion that plagues remote voting systems rears its head again, but there may be ways to offset this.

These avenues are the subject of current research.

14 Acknowledgements

The authors would like to thank Ben Adida, Jeremy Bryans, Roberto Delicata, Jeroen van der Graaf, Michael Jackson, Cliff Jones, Steve Kremer, Aad van Moorsel, Thea Peacock, Rene Peralta, Brian Randell, Ron Rivest, Mark Ryan, Fred Schneider, Robert Stroud, and Poorvi Vora for many helpful discussions.

This work was partially funded by the EPSRC DIRC project, www.dirc.org.uk.

References

1. Robert S. Brumbaugh. *Ancient Greek Gadgets and Machines*. Thomas Y. Crowell, 1966.
2. Jeremy W. Bryans and Peter Y. A. Ryan. A Dependability Analysis of the Chaum Voting Scheme. Technical Report CS-TR-809, University of Newcastle, 2003.
3. David Chaum. Secret-Ballot Receipts: True Voter-Verifiable Elections. *IEEE Security and Privacy*, 2(1):38–47, Jan/Feb 2004.
4. M. Jakobsson, M. Juels, and R. Rivest. Making Mix Nets Robust for Electronic Voting by Randomised Partial Checking. In *USENIX'02*, 2002.
5. R. Mercuri. A better ballot box? *IEEE Spectrum*, 39(10), 2002.

6. C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *ACM-CCS*, 2001.
7. Peter Y. A. Ryan. A Variant of the Chaum Voter-Verifiable Scheme. Technical Report CS-TR 864, University of Newcastle, 2004. Also in WITS 2005: Workshop on Issues in the Theory of Security.

A System-based Analysis of Prêt à Voter

Peter Y. A. Ryan and Thea Peacock

School of Computing Science, University of Newcastle,
Newcastle upon Tyne, NE1 7RU, United Kingdom

Abstract. It is widely recognised that the security of even the best-designed technical systems can be undermined by system-based weaknesses: implementation flaws, environmental factors that violate (often implicit) assumptions, faulty procedures and human fallibility. This is especially true of cryptographic voting systems, which have, typically, a large user base and are used infrequently.
In the spirit of the this observation, Karlof et al [10] presented a systems perspective analysis of the Chaum [4] and Neff [17] schemes. By stepping outside the purely technical, protocol specifications, they identify a number of potential vulnerabilities of these schemes. In this paper, we perform a similar analysis of the Prêt à Voter scheme [5].
Firstly, we examine the extent to which the vulnerabilities identified in [10] apply to Prêt à Voter. We then describe some further vulnerabilities and threats not identified in [10]. Some of these, such as chain voting attacks, do not apply to the Chaum or Neff schemes, but potentially apply to Prêt à Voter. Where appropriate, we propose enhancements and counter-measures.
Our analysis shows that Prêt à Voter is remarkably robust against a large class of system-based vulnerabilities, including those described in [10].

1 Introduction

Voting systems are the bedrock of democratic societies, and date back several millennia. While many different mechanisms for voting have been proposed [9], they usually share a similar set of goals such as accuracy, ballot secrecy, verifiability and coercion-resistance [12], [6].

In an attempt to improve the accessability and efficiency of the election process, democracies have experimented with various automated voting systems that increase speed and accuracy of ballot counting. Arguably, some of these new mechanisms offer greater secrecy, and in the case of remote systems, increased voter participation. However, these attempts have been fraught with problems, many of which are due to reliance on computer hardware and software performing as intended or claimed. (See for example [2], [1], [11], [13]).

Recent proposals for cryptographic voting systems hold out the promise of resolving many of these problems by introducing transparency and verifiability. Notable examples are the Chaum [4] and Neff [16], [17] schemes and Prêt à Voter [5]. These strive to provide assurance of secrecy and accuracy without any reliance on the underlying technical system (software, hardware etc.). Instead, the assurance is derived from a the properties of the cryptography and a high degree of transparency in the vote recording and counting stages.

While it is important to analyse the core protocol in order to assess its security [3], [12], it is also essential to consider its interaction with the surrounding system, e.g. computer hardware and software, voters and voting officials. For instance, collusion between parties can make the attacks more difficult to detect and resolve. This point was argued by Ryan [20], and more recently, demonstrated by Karlof et al [10] in their examination of the Chaum and Neff schemes from the systems perspective.

In this paper, we continue this theme with an analysis of Prêt à Voter, and in doing so, find that it is remarkably robust against the vulnerabilities described in [10]. In addition, we identify some new vulnerabilities, and offer mitigation strategies, particularly where they may apply to Prêt à Voter.

The structure of the paper is as follows. In Section 2 we briefly describe the Chaum and Neff schemes. In Section 3, we recall the potential weaknesses identified in [10]. In Section 4, we present an outline of the Prêt à Voter scheme. Section 5 examines the extent to which the attacks of [10] also apply to Prêt à Voter. Following this, in Section 6, we identify further possible attacks and suggest mitigations. In Section 7 some other vulnerabilities of Prêt à Voter are described, along with some counter-measures. Finally, we summarise and conclude in Section 8.

2 The Chaum and Neff Voting Schemes

Although the mechanisms differ in several ways, Prêt à Voter and the Chaum and Neff schemes all provide voter verifiability. More precisely, after casting her vote, the voter is provided with a physical receipt with the vote value cryptographically concealed, so ensuring secrecy. The voters can later check that their vote has been correctly posted to the *Web Bulletin Board* (WBB). Various checking mechanisms are deployed to detect any failure or corruption in either the encryption or decryption of the receipts, so ensuring accuracy.

Due to space constraints, we only summarise the key features of the Chaum and Neff schemes, and refer the reader to [5], [3] , [17], [18] for details, or [12], [22] for overviews.

In the booth, the voter interacts with a machine, during which the voter's choice is communicated to the machine and encoded in a receipt. The voter is given the opportunity to verifying that their vote is correctly encoded. This is achieved by way of a cut-and-choose protocol, in which the device commits to two or more encryptions of the vote, the voter chooses one to retain. The other encryptions are then opened, verified and discarded. A digital copy of the chosen receipt is retained by the device and, once the voting period has ended, is posted to a WBB . The voter retains a hard copy of their encrypted ballot receipt, which she can subsequently check on the WBB to make sure that the receipt has been correctly recorded.

The receipts are shuffled and decrypted in a series of anonymising mixes to ensure that no link remains between the encrypted ballot receipts and the final decrypted vote values. The results of each stage of the mix process are posted to the WBB.

Aside from the details of the cryptographic primitives involved and the mix processes, the essential difference between the two schemes is in the format of the receipt.

In the Neff scheme, the receipt consists of a matrix of ciphertexts of the digits '0' or '1'. A chosen candidate is distinguished by the arrangement of digits in the corresponding row.

In the Chaum scheme, the voter's choice is represented using visual cryptography [15] to generate a 2-D ballot image showing the candidate name, which is split between two encrypted, transparent layers. The ballot image is visible when the two sheets are accurately overlaid but, when separated, each sheet is a pattern of random pixels. The voter selects one layer to retain as a receipt, so without knowledge of the crypto keys, the receipt does not reveal the voter's choice.

3 Cryptographic Voting Protocols: Chinks in the Armour

The vulnerabilities considered in [10] fall into four main categories: those due to subliminal channels, "social engineering" style attacks against the cryptographic protocols, denial of service attacks and implementation flaws. In this section, we briefly recall each of these vulnerabilities and how they may apply to the Chaum and Neff schemes.

3.1 Subliminal Channels

Subliminal channels provide a means for a malicious agent to transmit information over a channel in a way that is hidden from the legitimate users of the channel. They can arise whenever there are alternative valid encodings of the "intended" information. Additional information can be encoded in suitable choices between these alternatives. Public access to the WBB makes this a particularly virulent threat for voter-verifiable schemes. There are two classes of subliminal channel identified in [10]: *random* and *semantic*.

The Neff scheme makes use of randomised crypto variables in the creation of the ballot receipt giving rise to a possible *random* subliminal channel. By judicious selection of random values, the voter's choice could be encoded in the encrypted receipt [10]. Any agent with knowledge of the coding strategy for this subliminal channel could then gather this information by observing the posted receipts.

Random channels do not occur in the Chaum scheme, as it uses deterministic algorithms. However, *semantic* subliminal channels, can occur if there are alternative valid representations of the ballot image. Certain information could then be conveyed by altering the ballot image, far example by adjusting the font size or positioning of the image. Note that, in contrast to the random channel of the Neff scheme, this channel emerges after the anonymising mixes, when the decrypted ballot images emerge. Thus a malicious device would presumably encode information about the voter identity rather than the vote value.

Mitigation Counter-measures for random subliminal channels are tricky, since the randomness may be essential for security properties. We refer the reader to [10] or [22] for details. A standard counter-measure, noted in Karlof et al, is to require the use of pre-determined randomness. In effect, the non-determinism is resolved before the voter choices are communicated to the system. The difficulty with this approach is ensuring that the devices adhere to this pre-determined entropy. To achieve this, Karlof et al suggest the use of trusted hardware [10], but this is contrary to the principle of transparency and minimal dependence which the Chaum, Neff and Prêt à Voter (see later) schemes aim to achieve.

Similarly, with the Chaum scheme, one could enforce a standard format for each the ballot image for each candidate option. This also runs into the difficulty of monitoring and enforcing adherence. It is also difficult to square with the possibility of "write-ins" that the Chaum scheme enables.

3.2 Social Engineering Attacks

Both the Chaum and Neff schemes require non-trivial, multi-step interactions between the voter and the machine. As both schemes involve cut and choose protocols, the sequence of steps in the protocol is highly significant. These are designed is to detect any attempt to construct receipts that do not encode the voter's true choice. We will refer to the choice the voter makes in a cut-and choose step as the protocol choice to distinguish this from the voter's candidate choice.

By re-ordering the steps in the protocol, or introducing extra ones, the machine could learn the voter's protocol choice before it has to commit to the receipt encoding [10]. In this case the device can corrupt the vote value with impunity. Voters may not notice or appreciate the significance of such changes in the protocol execution.

Similarly, the machine could feign an error and re-boot after it learns the voter's protocol choice. Relying on the voter making the same choice in the second round of the protocol, the machine then constructs a receipt for a different candidate. If the voter changes her mind, the machine re-boots again until the voter gets it "right" [10].

Another possible vulnerability of the Chaum scheme, not identified in [10], but mentioned in [20], is as follows. The machine attempts to corrupt the vote value by incorrectly constructing one of the layers. If the voter chooses the other layer for retention, the corruption will go undetected. However, if the voter chooses the corrupted layer (which would lead to detection), the machine could try to fool the voter by printing 'destroy' instead of 'retain' on the layer that the voter chose as the receipt. If the voter fails to notice this, or simply ignores

it, the corruption will again pass undetected. This is another way that a corrupt machine could undermine the cut and choose element of the protocol. Even if the voter does notice the switch and is confident that they are right, it may be difficult to demonstrate this to a voting official.

Mitigation We refer the reader to [10] or [22] for suggested mitigations. The obvious approach is to try to ensure that voters understand the procedures and the appreciate their motivation. This is similar to the notion of instilling a "security culture" in an organisation. In practice of course, this may not really be feasible. In the context of an election system, where the set of users can be vast and usage infrequent, the possibilities for voter eduction is limited.

However, we suggest that a better solution is to make the voter experience much simpler. This is a characteristic of the Prêt à Voter scheme as we describe later.

3.3 Denial of Service

There are several ways in which DoS attacks could disrupt or invalidate an election. See [10] for a discussion. For all such attacks, adequate error-handling and recovery strategies need to be in place. In addition, some form of back-up is desirable, e.g. a *voter-verifiable paper audit trail* (VVPAT) [14]. We return to these attacks and counter-measures later, when we examine the robustness of Prêt à Voter.

4 The Prêt à Voter Scheme

We now present an overview of the Prêt à Voter scheme. For full details see [5]. Prêt à Voter is inspired by the Chaum scheme, but replaces the visual cryptographic techniques with a conceptually and technologically simpler mechanism to represent the encrypted vote value in the ballot receipt. In the polling station, voters select a ballot form at random, an example of which is shown below.

Democritus	
Plato	
Socrates	
Thales	
	$7rJ94K$

In the booth, the voter makes her selection in the usual way by, for example, placing a cross in the RH column against the candidate of choice. She now separates the right hand (RH) and left hand (LH) sides, and the latter, which carries the candidate list, is discarded to leave the ballot receipt. Supposing a vote for Plato, the receipt would appear thus:

X
$7rJ94K$

She then leaves the booth and authenticates herself with to an official, who scores off her name in the register. The receipt is placed under an optical reader, or similar device, to record the cryptographic value at the bottom of the strip, and the numerical representation of the cell into which the cross has been entered. The voter retains the hard copy of the RH strip as her receipt. The ballot forms would incorporate various anti-counterfeiting devices, and the receipt would be digitally signed when the vote is cast.

Possession of a receipt might appear to open up the possibility of coercion or vote-buying. However, the candidate lists on the ballot forms are randomised, so the order in which the candidates are shown is unpredictable. Clearly, as long as the LH strip is removed, the RH strip alone does not indicate which way the vote was cast.

The cryptographic value printed on the bottom of the receipt, the 'onion', is the key to extraction of the vote. Buried cryptographically in this value, is the seed information needed to reconstruct the candidate list. This information is encrypted under the secret keys of a number of tellers. Thus, only the tellers acting in consort are able to reconstruct the candidate order and so interpret the vote value encoded on the receipt.

Once the election has closed, all the receipts are transmitted to a central tabulation server which posts them to a secure WBB. This is an append-only, publicly visible facility. Only the tabulation server can write to this and, once written, anything posted to it will remain unchanged. Voters can visit this WBB and confirm that their receipt appears correctly.

After a suitable period, the tellers take over and perform a robust, anonymising, decryption mix on the batch of posted receipts. Various approaches to auditing the mixes can be used to ensure that the tellers perform the decryptions correctly. Details of this can be found in [5].

Another place where the accuracy of the scheme could be undermined is in the vote capture stage, and the encoding of votes in the receipts. In the case of Prêt à Voter, this could occur if the ballot forms are incorrectly constructed, i.e., the candidate list shown does not correspond to the crypto seeds buried in the onion value. Various mechanisms can be deployed to detect and deter such corruption. The approach suggested in [5] is to perform a random pre-audit of the ballot forms. In other words, both independent third parties and the voter can check the well-formedness of the ballot forms. The checks performed in each case, however, differ. The former involves extracting the crypto seeds, re-computing the onions and checking that they correspond to to the candidate permutation.

Later in this paper, we discuss an alternative approach using on-demand creation and printing of forms along with a cut-and-choose mechanism and post-auditing.

For full details of the mechanisms used to detect any malfunction or misbehaviour by the devices or processes that comprise the scheme, see [5].

5 Systems-based Analysis of Prêt à Voter

In this section we examine the extent to which the vulnerabilities identified in [10] apply to Prêt à Voter.

5.1 Subliminal channels

Subliminal channels, certainly of the type identified by Karlof et al, are not a problem for Prêt à Voter. This is due mainly to the rather special and, to our knowledge unique, way that votes are encoded in Prêt à Voter. Most cryptographic voting schemes require the voter to supply her vote choice to the device, which then produces a (verifiable) encryption. In the case of Prêt à Voter, the voter's choice is encoded in a randomised frame of reference. It is the information that allows this frame of reference to be recovered, the "seed" value, that is encrypted. This can be done ahead of time without needing to know the vote value. In Prêt à Voter, the ballot forms are generated in advance and allocated randomly to voters. Thus, the cryptographic commitments are made before any linkage to voter identities or vote choices are established.

Regarding semantic subliminal channels, suitable implementation should ensure that, for a given ballot form and vote choice, the digital representation in a Prêt à Voter receipt is unique, i.e., the onion value and the index value indicating the chosen cell on the LH column. Hence, there should be no possibility of a semantic subliminal channel.

Like the Chaum scheme, Prêt à Voter "Classic", [5], uses deterministic cryptographic algorithms. In fact, all the tellers' operations can be made deterministic: encryption and decryption are deterministic and the tellers can be required to post batches of transformed ballot receipts in lexical order at each stage, thus eliminating random subliminal channels. It is not clear whether this is really necessary, as the tellers do not have access to any privacy sensitive information.

5.2 Social Engineering Attacks

In Prêt à Voter, the voter does not engage in a multi-step protocol with the device so there is little scope for any social engineering style attacks.

It is worth noting that in Prêt à Voter, the analogue of the 'cut and choose' element of the Chaum or Neff schemes is performed by independent auditing authorities by checking a random selection of the ballot forms. Thus the voters are not required to perform a cut-and-choose. The authority commits to the crypto material on the ballot forms ahead of the election. A random selection of these are checked by the auditors for well-formedness. This can be done before, during and after (on unused, left-over forms) the election period. Assuming that they pass the checks, audited forms are destroyed. Forms that fail the checks would be retained for forensic purposes.

Prêt à Voter also allows the possibility of voters performing random checks on the ballot forms in addition to checks performed by auditors. This is more in the spirit of arranging for the trust to reside solely in the voters themselves. Care has to be taken, however, to avoid introducing other vulnerabilities, such as the possibility of checking ballot forms that have been used to cast votes.

5.3 Denial of Service

Whilst these schemes succeed in removing the need to trust the devices and tellers for the accuracy requirement, we may still be dependent on them to some extent for availability. Unless suitable measures are taken, the failure of a teller for example, could at least hold up and in the worst case block the tabulation. Corruption of the digital copies of receipts would call the election into doubt. Thus measures must be taken to make the scheme robust against (manifest) failure or corruption of devices. In other words, we have endured that, with high probability, any (significant) failures or corruption will be detected, but we still have to address the isssue of error handling and recovery.

A possible enhancement of Prêt à Voter, detailed in [23], is to replace the decryption mixes with re-encryption mixes. This has a number of advantages, one being that recovery from DoS failures is much easier. There are a number of reasons for this:

- The mix tellers do not need secret keys, they simply re-randomise the encryption. A failed mix teller can therefore simply be replaced without having to surgically extract keys.
- The mix and audit can be independently re-run. With (deterministic) decryption mixes, the selection of links for audits cannot be independently selected on the re-run of the mix without compromising secrecy.

The use of threshold encryption schemes would also help to foil DoS attacks by ensuring the failure of a proportion of the (decryption) tellers could be tolerated.

In [19], it is suggested that a VVPAT [14]-style mechanism be incorporated. As the device scans the voter's receipt, it generates an extra copy. Once this copy has been verified by the voter (and possibly an official), it is entered into a sealed audit box. This provides a physical back-up of receipts cast, should recovery mechanisms need to be invoked.

We emphasise that this is actually quite different to the conventional notion of VVPAT that generates a paper audit trail of unencrypted ballot slips. The conventional form of VVPAT suffers from a number of privacy problems. For example, if the trail retains the order of votes cast, receipts could be linked to voters by comparing the order in which votes are cast with that of voters entering

the booth. Also, any mismatch between the voter's choice and the paper audit record can be difficult to resolve without compromising the voter's privacy.

Where encrypted receipts are recorded, these problems disappear. Note that, due to the encrypted form of the receipts, it is possible for monitors to verify a vote as it is cast, in addition to the voter checking. Any discrepancies can be resolved without loss of voter privacy. We will henceforth refer to such a mechanism as *Verified Paper Audit Trail* (VPAT).

5.4 Discarded Receipts

As with the Chaum and Neff schemes, carelessly discarded receipts could be a problem in Prêt à Voter, as this could indicate which receipts will not be checked by voters on the WBB [10]. Malicious parties could delete or alter the corresponding receipts, confident that the voters will not check them on the WBB.

Aside from voter education [10], a possible mitigation is, again, to invoke a VPAT mechanism, as described above. Independent observers could check the correspondence between the VPAT and the contents of the WBB. This has the added advantage in that less reliance need be placed on the voter's diligence in checking their receipts on the WBB.

5.5 Invalid Digital Signatures

In the Chaum scheme, digital signatures act as a counter-measure against faked receipts being used to discredit election integrity. However, a device that falsified signatures could be used to discredit voters, leaving them without a way to prove a dishonest system [10].

Voters should thus be provided with devices capable of veifying the digital signatures. Such devices could be provided by various independent organisations, such as the Electoral Commission, etc. Similar measures could be utilised for Prêt à Voter.

Given that encrypted receipts can be cast in the presence of officials and other observers, we have the possibility of checking digital signatures at the time of casting and applying physical authentication mechanisms, such as franking, to the receipt.

5.6 Insecure Web Bulletin Board

Like the Chaum and Neff schemes and indeed many cryptographic voting schemes, Prêt à Voter relies on a secure WBB, which. In a possible attack, the WBB arranges for the voter to see a correct record of her ballot receipt, which, in collusion with the mix-net, has been deleted or altered. As a result, the voter could mistakenly believe that her vote has been accurately counted [10].

However, we note that suggested mitigations, such as robust data storage and allowing only authorised write access to the WBB [10] are still vulnerable to corruption. The challenge is provide a trusted path from the WBB to the voter. The problem of implementing secure web bulletin boards is the subject of ongoing research in the cryptographic voting community.

6 Further Vulnerabilities

In this section, we discuss other possible vulnerabilities not identified in [10], and suggest appropriate mitigation strategies.

6.1 Doll Matching Attack

This vulnerability is specific to the Chaum scheme and is not identified in [10]. Each of the layers that combine to reveal the ballot image have to carry a pair of "dolls", anologues of our "onions", one for each layer. It is essential that these are identical between the layers. The voter is required to check that values on the two layers match. Failure to do this could allow the device to construct fake receipts without detection. It might seem difficult to produce a fake doll that alters the vote value whilst differing from the real doll in a way that is visually almost imperceptible. This would be analogous to finding (approximate) collisions of a cryptographic hash function. Nevertheless, this should still be considered a potential vulnerability.

A counter-measure is to ensure that visual matching of the dolls is as easy to perform as possible. Thus, for example, the dolls might be encoded aligned bar codes on the two layers. Any mismatch would then show up as a misalignment of the bars.

6.2 Undermining Public Confidence in the Secrecy of Encrypted Receipts

Another potential attack against schemes employing encrypted receipts, is as follows. The Mafia claim to have a way of extracting a vote from the encrypted receipt. If a sufficient number of voters were convinced by such a claim, and so influenced to alter their vote, it may be possible to undermine the outcome of the election. For instance, a coercer might urge voters to submit their receipts, claiming that the 'correctness' of the choice would be checked. Some reward, such as entry into a lottery, would be offered for receipts that passed the supposed checks.

Countering such a psychological attack, other than by voter education, could be difficult.

6.3 Side-channel Attacks

Karlof et al do not discuss the possibility of the vote-capture devices leaking the voters choices via side-channels, e.g., hidden wires, wireless-enabled devices, etc.

This may be because they are regarded as inevitable. In fact, in the case of Prêt à Voter, such channels are not, in fact, a problem.

As explained earlier, the voter's choice does not need to be communicated to the device in order to obtain the encoding of this choice in the receipt. Hence, even if such channels existed, they could not be exploited to leak information from the booth device.

6.4 Kelptographic Channel Attacks

On the other hand, the current version of Prêt à Voter is still vulnerable to another form of subliminal channel: a Kleptographic channel. These were orginally described by Young and Yung [24] and their relevance to voting schemes described by Zagorski et al [8]. In the case of Prêt à Voter, the idea is that the authority creating the ballot forms would carefully select the seed values in such a way as to encode information about the candidate list in the onion values. This encoding would use some secret key shared with a colluding third party. Thus, seeds would be chosen so that a certain keyed hash applied to the onion value would carry information about the corresponding candidate order.

Clearly this would require a significant amount of searching in the seed space and computation, but the resulting selection of seeds would appear random to anyone not knowing the coding strategy and secret hash key.

It is fairly straightforward to eliminate this kind of attack by arranging for the seeds to be created in a distributed fashion by several entities in such a way that no single entity can control or know the resulting seed values. Ryan et al [23] describes such a mechanism, in which several trustees create proto-ballot forms in a kind of pre-mix. The resulting proto-ballot forms have two distinct onions encrypting the same seed value. The seed value, and hence candidate list, can be extracted from one of these on demand to reveal the full ballot form. Full details can be found in [23].

7 Prêt à Voter Specific Vulnerabilities

Our analysis of Prêt à Voter revealed other possible vulnerabilities that are specific to the scheme. We discuss them and suggest possible mitigations.

7.1 Chain Voting

Chain voting is a well known style of attack that can be effective against some conventional paper ballot schemes. In this attack, the coercer smuggles an unused ballot form out of the polling station and marks his preferred candidate. The voter is told that they will be rewarded if they emerge with a fresh, unmarked form. This can then be marked again and passed to the next voter.

Particularly vulnerable are election systems in which the ballot forms are a controlled resource: on registration, voters are given one ballot form and they are observed to cast this before existing the polling station. The voter is then under duress to cast the form marked by the coercer and to retain the fresh form that they are provided with when they register. The procedures arguably make it harder for the coercer to initialise the attack. However, a determined attacker would certainly find a way, for example by bribing an official, or placing a fake form in the ballot box. Once the attack is initialised, the procedure works in the coercer's favour.

A possible counter-measure is to make ballot forms freely available in the polling stations, as, for example, in French elections. Voter identity is checked when casting a vote rather than at the time of collecting a ballot form. Thus, as it is not certain that a marked form was actually used to cast a vote, the motivation for the attack is undermined.

Neither the Chaum nor the Neff schemes, in which the ballot forms and receipts are generated on demand in the booth, are vulnerable to this style of attack. Prêt à Voter is, however, potentially vulnerable, as the ballot forms are pre-printed. The counter-measure above does not work as cast receipts are posted to a publicly-verifiable WBB.

7.2 Mitigation

Observe that at the time of making her candidate choice, it is only necessary for the voter to see the candidate list . A possible counter-measure therefore is to conceal the onion by a 'scratch strip', similar to that used in lottery tickets. The procedure could then be for the voter to register and collect a fresh ballot form, with scratch strip intact. The voter goes to the booth, marks her selection, then detaches and destroys the LH strip. She exits the booth and takes her receipt to an official who checks her identity and that the scratch strip is intact. The voter, or an official, now removes the strip and records the receipt as previously described.

Steps would need to be taken to ensure that the scratch strips cannot be scanned with some device to read the concealed onions. This has reportedly been done using the laser photo-acoustic effect. See [7] for details. Although expense and technical know-how would be required on the part of the attacker, it should still be considered a possible threat.

A rather different counter-measure is to return to an on-demand creation of ballot forms, e.g. printing in the booths. This avoids chain voting and certain chain of custody issues but at the cost of having to re-introduce the voter involvement in the 'cut and choose' along with post-auditing to the protocol. The trade-offs involved in this are investigated in [21].

7.3 Authority knowledge

In the current version of Prêt à Voter, the authority has knowledge of ballot form information, i.e. the crypto seeds used to generate the candidate offsets, hence the onions, and, in particular, the association between these values. This means that the authority has to be trusted not to leak this information. Even if the authority is entirely trustworthy, there is always a danger of this information

being leaked during distribution or storage of the ballot forms, i.e., chain of custody issues.

To counter this, we arrange for the ballot form material to be constructed in a distributed fashion, in such a way as to ensure that no single entity knows the association of onions and candidate lists. As noted previously, up until the time of casting a vote, it is not necessary for the voter to see the onion. One could thus devise a scheme in which the onion and candidate list are never exposed simultaneously.

We now outline a simple scheme for distributed construction of scratch card style ballot forms. It is based on onions encrypted using El Gamal, or a similar randomised encryption algorithm. The ballot form material is generated by a number of ballot clerks. The first clerk generates a large quantity of onions, which then enter a re-encryption pre-mix involving the other clerks. The last clerk collects the resulting permuted, random onions and, for each one, produces two re-encryptions. These paired onions are now printed onto ballot forms, one at the bottom of the LH column, the other on the bottom of the RH column. A proto-ballot form is shown below:

	7rJ94K
jH89Gq	

These two onions should correspond to the same candidate list. The RH onion is now concealed with a scratch strip.

These are now shuffled and passed to a final clerk who then dispatches the visible, LH onions to the tellers. The tellers send back the corresponding candidate list which is now printed in the LH column and the LH onion is removed. This results in ballot forms similar to those proposed in Section 4, but with the onion value concealed by a scratch strip.

Note that no single entity now knows the association of onion and scratch strip. Strictly speaking, the last two clerks acting in collusion could form the association, but the scheme can be elaborated to raise the collusion threshold. Interestingly, we note also that, even though these two clerks in collusion know the onion/candidate list association, they cannot prove this knowledge to a third party as they do not know the necessary crypto seeds used to compute the onion values. Ballot forms can be randomly audited as before.

Alternatively, the pre-mix approach of [23] alluded to earlier as a counter to kelptographic attacks, would also be effective here to eliminate the authority knowledge problem.

7.4 Enforcing the Destruction of the Left-hand Strips

After the voter's selection has been marked on the ballot form, the left-hand strip must be destroyed. Failure to do so would allow the voter to use it as proof of her vote to a third party. Clearly, this would lay the system open to coercion.

Several ways of enforcing this are possible. The voter could be required to destroy the left-hand strip in the presence of an official, preferably in some mechanical shredding device. This could be done at the time of casting the ballot form, as suggested above. However, care would have to be taken to ensure that the official is not able to record the association of the receipt and candidate list.

Another possibility is to have devices in the booth that would automatically cut off and destroy the LH strip and then pass the receipt into a scanner. This would make the voter's interaction simpler, but such devices would have to be carefully evaluated, and run counter to the "trust nobody and nothing" philosophy of voter-verifiable schemes.

Another possibility is to make 'decoy' left-hand strips freely available in the booths, so the voter cannot convince the coercer that the one she emerges with is genuine.

7.5 Confusion of Teller Modes

As previously mentioned, the tellers perform an anonymising decryption mix on the receipts posted to the WBB. However, they also have a role in checking the construction of ballot forms, both by auditors and, potentially, voters [5]. For ballot forms selected for audit, the onions are sent to the tellers, who return the corresponding seed values. The auditors then re-compute the onion values and candidate offsets, and check that they are correct. In voter checking, the tellers return the candidate ordering corresponding to the onion value sent by the voter.

The checked forms should then be discarded. If the audited forms were later used to cast a vote, there could be a threat to ballot secrecy. Conversely, it should not be possible to run a check on a form that has been used to cast a vote.

To mitigate this, ballot forms could be checked by voters in the presence of an official, who then ensures that used forms are discarded. Forms could be invalidated once used, for example, using the described scratch strip mechanism. An authentication code could be overprinted on the scratch strip that would be necessary to enable the checking mode. Revealing the onion would entail removing the scratch strip and the code along with it, ensuring that the form could not be reused later.

8 Conclusions

In this paper, we have performed a systems-based analysis of the Prêt à Voter scheme. In particular, we have extended the analysis of Karlof et al with some

further systems based vulnerabilities not identified in [10], and considered the applicability of these vulnerabilities to the Prêt à Voter scheme.

Prêt à Voter has proved to be remarkably resilient to these vulnerabilities, many of which stem from voter interaction with devices and generation of entropy at the time of voting. However, Prêt à Voter, is potentially prey to chain voting attacks, which do not apply to schemes in which the crypto material is generated on demand. In fact, as we have discussed, this attack is particularly virulent in the context of voter-verifiable schemes with pre-prepared ballot forms and Web Bulletin Boards. Wherever such threats apply to Prêt à Voter, counter-measures have been suggested.

As with any secure system, voting schemes require great care in their design and evaluation, not only of the cryptographic core, but also of the surrounding system. Analysis from a system perspective has provided valuable insight into the way forward for Prêt à Voter, and some enhancements have been suggested. It has also underlined the need for adequate error-handling and recovery strategies, and that a VPAT style mechanism would be highly desirable.

The analysis presented here, as with the analysis of Karlof et al, does not of course constitute an exhaustive, systematic, identification of all the system-based threats to Prêt à Voter. Arguably, complete coverage for such an analysis could never be guaranteed given the open-ended nature of systems. However, we feel that this analysis constitutes a useful first step towards a more systematic analysis technique for voting systems. Here, we explore the space of possible failure modes and adversary capabilities, which will enable us to build a threat model.

We already have the start of a taxonomy of attacks, i.e., classification into subliminal channels, side-channels, kleptographic channels, social engineering threats, etc. It seems likely that a form of design-level information flow analysis should help guide further analysis. This will be pursued in future research.

Finally, we conclude that, provided that, in addition to a formal analysis of the core, technical system, a systems perspective is taken into consideration during the design and evaluation phases, there is every reason to suppose that cryptographic schemes of this kind can provide trustworthy, verifiable elections.

9 Acknowledgements

The authors would like to thank Michael Clarkson, Michael Jackson, Steve Kremer, Andrey Povyakalo, Mark Ryan and Luca Vigano for fruitful discussions and DSTL and the EPSRC DIRC project for partial funding of this work.

References

1. The trouble with technology. *The Economist*, Sept 16 2004.
2. J. Bannet, W. Price, A. Rudys, J. Singer, and D. Wallach. Hack-a-vote: Security issues with electronic voting systems. *IEEE Security and Privacy*, 2(1), Jan/Feb 2004.
3. J. Bryans and P.Y.A. Ryan. A dependability analysis of the chaum voting scheme. Technical Report CS-TR-809, University of Newcastle upon Tyne, 2003.
4. D. Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, 2(1):38–47, January-February 2004.
5. D. Chaum, P.Y.A. Ryan, and S. Schneider. A practical, voter-verifiable election scheme. In *European Symposium on Research in Computer Security*, number 3679 in Lecture Notes in Computer Science. Springer-Verlag, 2005.
6. A. Fujioka and T. Okamoto andK. Ohta. A practical secret voting scheme for large scale elections. In *Workshop on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, pages 244–251. ACM, 1992.
7. E. Gerck. Instant lottery cards too, re: reading pins in 'secure' mailers without opening them, 2005. http://www.mail-archive.com/cryptography
8. M. Gogolewski, M. Klonowski, P. Kubiak, M. Kutylowski, A. Lauks, and F. Zagorski. Kleptographic attacks on e-election schemes with receipts, 2006. http://www.nesc.ac.uk/talks/639/Day2/workshop-slides2.pdf.
9. D. W. Jones. A brief illustrated history of voting, 2003. http://www.cs.uiowa.edo/ jones/voting/pictures.
10. C. Karlof, N. Sastry, and D. Wagner. Cryptographic voting protocols: A systems perspective. In *USENIX Security Symposium*, number 3444 in Lecture Notes in Computer Science, pages 186–200. Springer-Verlag, 2005.
11. T. Kohno, A. Stubblefield, A. D. Rubin, and D. S. Wallach. Analysis of an electronic voting system. In *Symposium on Security and Privacy*. IEEE, 2004.
12. S. Kremer and M. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In *European Symposium on Programming*, number 3444 in Lecture Notes in Computer Science, pages 186–200. Springer-Verlag, 2005.
13. T. W. Lauer. The risk of e-voting. *Electronic Journal of e-Government*, 2(3), Dec 2004.
14. R. Mercuri. A better ballot box? *IEEE Spectrum Online*, October 2002.
15. M. Naor and A. Shamir. Visual cryptography. In *Advances in Cryptology*, number 950 in Lecture Notes in Computer Science, pages 1–12. Springer-Verlag, 1994.
16. A. Neff. A verifiable secret shuffle and its application to e-voting. In *Conference on Computer and Communications Security*, pages 116–125. ACM, 2001.
17. A. Neff. Practical high certainty intent verification for encrypted votes, 2004. http://www.votehere.net/documentation/vhti.
18. A. Neff. Verifiable mixing(shuffling) of el-gamal pairs, 2004. http://www.votehere.net/documentation/vhti.
19. B. Randell and P.Y.A. Ryan. Voting technologies and trust. *IEEE Security & Privacy*, 2005. To appear.
20. P.Y.A. Ryan. Towards a dependability case for the chaum voting scheme, 2004. DIMACS Workshop on Electronic Voting – Theory and Practice.
21. P.Y.A. Ryan. Putting the human back in voting protocols. In *Fourteenth International Workshop on Security Protocols*, Lecture Notes in Computer Science. Springer-Verlag, 2006. To appear.
22. P.Y.A. Ryan and T. Peacock. Prêt à voter: a systems perspective. Technical Report CS-TR-929, University of Newcastle upon Tyne, 2005.
23. P.Y.A. Ryan and S. A. Schneider. Prêt à voter with re-encryption mixes. Technical Report CS-TR-956, University of Newcastle upon Tyne, 2006.
24. A. Young and M. Yung. The dark side of black-box cryptography, or: Should we trust capstone? In Neal Koblitz, editor, *Crypto '96*. Springer-Verlag, 1996.

Mesures de performances des systèmes embarqués du type Java Card

Virtualisation de système et sécurité

Pierre Paradinas

Professeur au CNAM
France

La carte à microprocesseur
De la carte à mémoire à Java Card
Un exemple de système embarqué

Pierre.Paradinas @ cnam.fr
Cnam/Cedric
Systèmes Enfouis et Embarqués (SEE)

Plan Java Card

- La carte à "mémoire" histoire d'une technologie
- La carte à microprocesseur
  - Architecture matériel
  - Normes et standards
  - Domaine d'applications
  - **Java Card**
  - Autre modèle .Net
  - Nouveaux modèles applicatifs
- Enjeux et perspectives de l'industrie

Java Card - Introduction

- Besoin marché vers des systèmes "programmables"
  - Recherche d'un meilleur T2M
  - Recherche de solution "évolutive" (dépasser la ROM)
- Avant d'arriver à cela les applications sont :
  - Longues à développer (cycle de deux ans !)
  - Statiques dans leurs ensembles de fonctions (ROM)
- Des tentatives...
- 1ère version : octobre 1996, démarrage et produit réel en 1998, une réalité industrielle à partir de 2000. En 2004, le nombre de Java Card vendu atteindra le milliard.

Etapes du développement de l'industrie

- La carte à microprocesseur et les grandes étapes de développement de la technologie (marché et technique) :
  - Les pionniers (1975-1985) : de(s) idées aux premiers produits, les bases technologiques sont établies,
  - 1985-1995 : les marchés et les déploiement importants sont là (CB & GSM), la technologie du départ est améliorée et renforcée, des limites apparaissent comme le besoin de flexibilité qui annonce Java Card,
  - 1995-2005 : explosion du marché (EMV, ID), qui s'accompagne d'un nouveau paradigme les cartes évolutives basées sur Java Card.
  - 2005-?? : la carte devient un élément du "réseau".

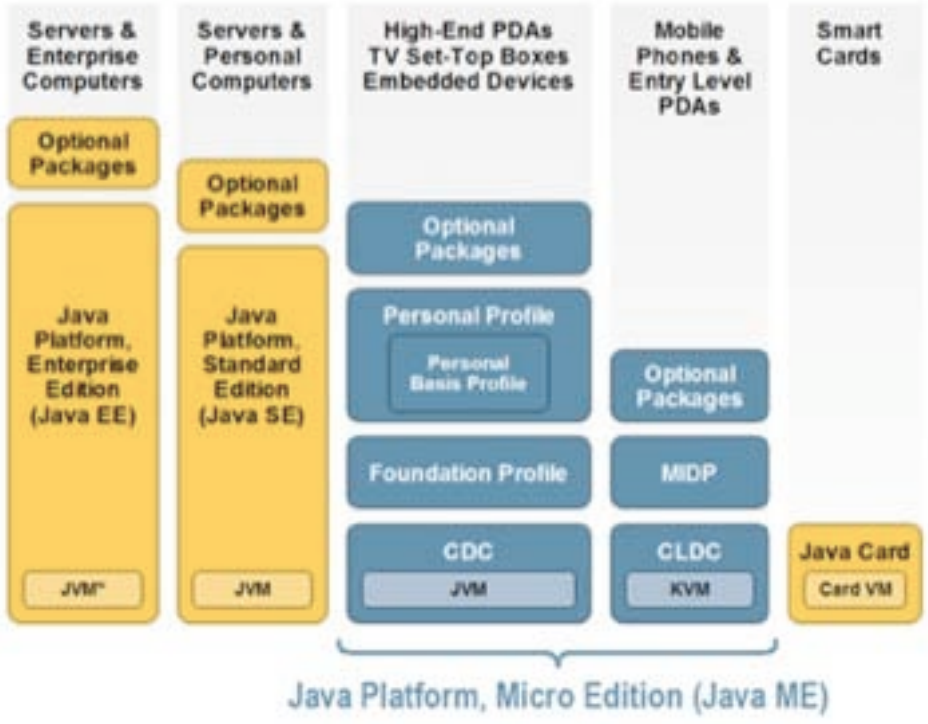
L'arrivée de la Java Card

- Novembre 1996, la première proposition d'utilisation de Java pour les cartes est faite par une équipe de Schlumberger (Austin),
  - Proposition de Java Card API permettant la programmation en Java de/dans la carte
  - C'est la Java Card 1.0
- Bull, Gemplus et Schlumberger créent le Java Card Forum
  - Le JCF discute et propose à Sun des spécifications.
- Novembre 1997, publication de la spécification Java Card 2.0
  - Gemplus démontre en octobre/novembre CASCADE, le premier chip RISC 32 bit (ARM 7) avec de la mémoire Flash, "une/son" implémentation de la Java Card 2.0 et des DMIs (Direct Method Invocation)...

Les évolutions de la 2.x

- La version 2.0 de la Java Card Specification introduit :
  - Un environnement d'exécution
  - La possibilité d'écrire des applets avec une approche OO,
    - même si le format de chargement n'est pas encore spécifié...
- Mars 1999, la version 2.1 qui contient 3 parties est publiée :
  - Java Card API Specification
  - Java Card Runtime Environment Specification
  - Java Card Virtual Machine Specification

Un élément de la technologie Java



A propos du modèle de licence...

- La spécification est disponible sur :
  - http://java.sun.com/products/javacard/
- Vendre des cartes (avec ou sans logo) et afficher une compatibilité avec la technologie impliquent d'être licencié **Java Card Technology**;
  - ce qui donne accès à :
    - Une implémentation de référence
    - La suite de test de compatibilité
    - Du support spécifique

A propos du modèle de licence...

- JAVA AUTHORIZED LICENSEES OF JAVA CARD TECHNOLOGY
  - The companies listed below have licensed Java Card technology from Sun Microsystems. Only Java Card licensees can ship products that bear the "Java Powered" logo and claim compatibility with the Java Card Platform specification and Java Card TCK.
    - ARM, Aspects, Axalto, CCL/ITRI, Fujitsu, Gemplus, Giesecke & Devrient GmbH, GoldKey Technology, HiSmarTech, I'M Technologies Ltd., IBM, incard, KEBTechnology, Logos Smart Card, Microelectrónica Española, Oberthur Card Systems, ORGA Kartensysteme, SAGEM, Sermepa, Setec, Sharp, Smart Card Laboratory Inc., SSP Solutions, Inc., STMicroelectronics, Toppan Printing, Trusted Logic.
      - D'après http://java.sun.com/products/javacard/licensees.html (le 18/11/05)

Avantages de la technologie

- Interoperabilité des applets
- Sécurité (sécurité du langage, évaluation,...)
- Multi-application
- Dynamique
- Ouverte et compatibilité

Le Java Card Forum

- Association regroupant les fabricants de silicium, les encarteurs et des clients
  - Promouvoir la solution de la Java Card
  - Définir des choix technologiques puis les proposer à Sun qui en fait le "standard". Processus différent du JCP (Java Community Process : http://jcp.org).

http://
www.javacardforum.org

Introduction à la technologie Java Card

- Prise en compte des architectures matérielles des cartes dont les tailles sont très réduites : moins de 1Ko RAM, 24-48 Ko de ROM et 8 à 16 Ko NVM (EEPROM).
- Pour "mettre la technologie Java dans une carte", les choix suivants sont faits :
  - Réduire les fonctionnalités du langage
  - "Distribuer le modèle" la VM entre le "in-card" et le "off-card"
- 3 parties :
  - Java Card API Specification
  - Java Card Runtime Environment Specification
  - Java Card Virtual Machine Specification

Java Card Langage et Java Langage

Caractéristiques supportées	Caractéristiques non supportées
booléen, octet, court	long double, flottant
tableau à une dimension	caractères et chaînes
paquetage Java, classes, interface et exceptions	tableau à plusieurs dimensions
héritage, méthode virtuelle, surcharge et création d'objet	chargement dynamique de classe
mot clé `int` et entier 32-bit sont optionnel	gestionnaire de sécurité (security manager)
	pas de ramasse miettes
	threads
	sérialisation

Java Card API 2.1

- 3 paquetages de référence
  - java.lang
  - javacard.framework
  - javacard.security
- Une extension
  - javacardx.crypto (liée à des problèmes d'exportation de produit)

java.lang

- Strict sous ensemble de `java.lang`
- La classe `objet` est la racine de la hiérachie de classe Java Card
- Classes supportées : `Objet` and `Throwable`
- La classe `Objet` a qu'un constructeur et par défault est la méthode `equals`

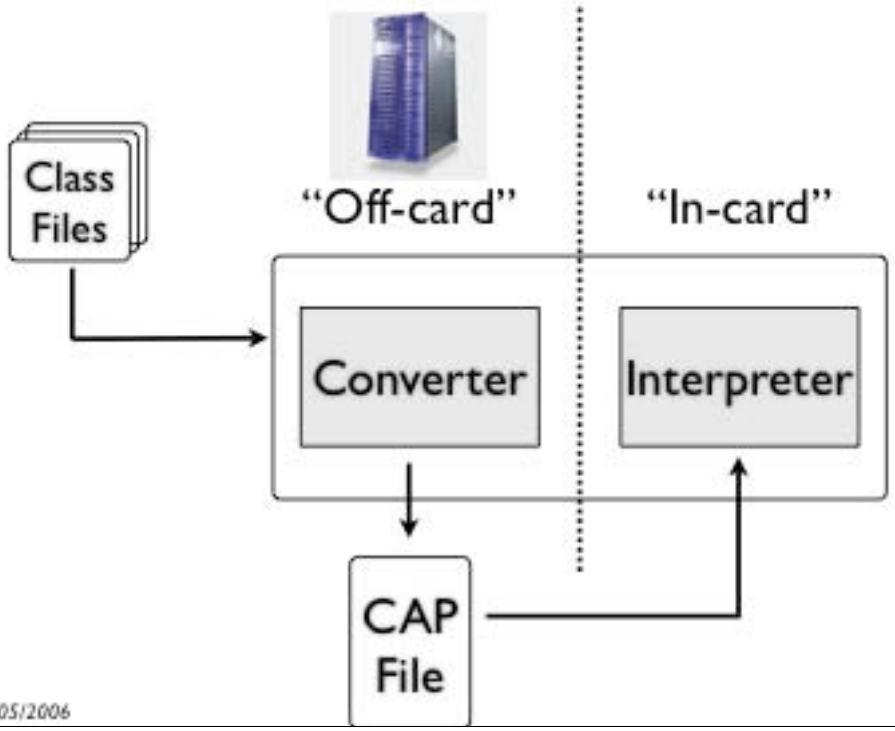
javacard.framework

- Contient les "spécificités" de la carte
- La classe `Applet` :
  - Fournit un cadre à l'exécution et à l'interaction avec le JCRE de l'applet
  - Les applets utilisateurs doivent étendre cette classe
- La classe `APDU`
  - Pour la gestion des échanges avec le monde extérieur
- La classe `PIN`
  - Assure la gestin de code secret

javacard.security

- Basé sur le paquetage java.securityy
- Permet la gestion des clés et fonction cryptographique
- En plus des algorithmes classiques, ajoute aussi la fonction génération de nombre aléatoire, signature et calcul de fonction de compressions.

La modèle machine virtuelle



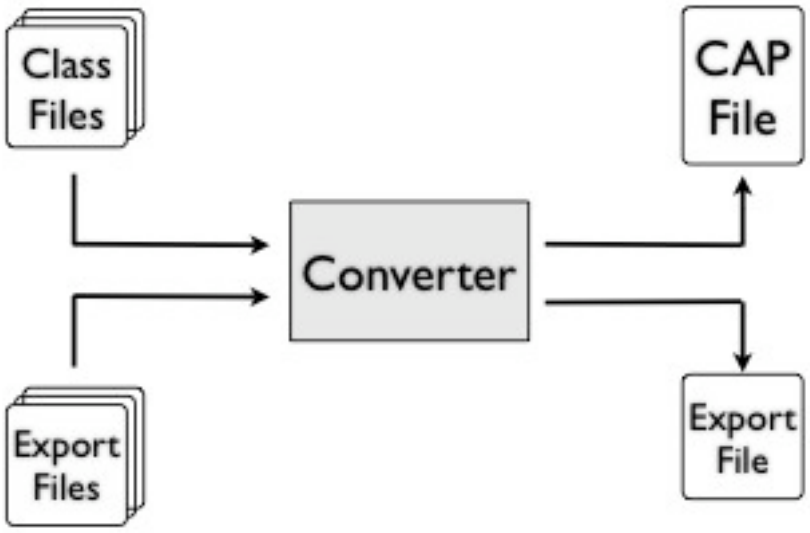
CAP file et Export File

- Le "CAP File" contient :
  - Représentation binaire exécutable de classes
    - information sur les classes
    - BC (Byte Code) exécutable
    - information nécessaire à l'édition de lien
    - information pour la vérification,...
  - Il est au format JAR (Java Archive), c'est ce qui est reçu par la carte

CAP file et Export File

- Le fichier "Export" est utilisé par le convertisseur.
- Information utilisée pour la vérification et l'édition de lien.
- Contient des informations publiques sur les API :
  - nom et champs des classes
  - signatures des méthodes
  - information pour l'édition de lien entre package
  - Il ne contient pas de BC, il peut donc être publié avec une applet permettant ainsi à celle-ci d'avoir des objets utilisables (partageables).

Le convertisseur



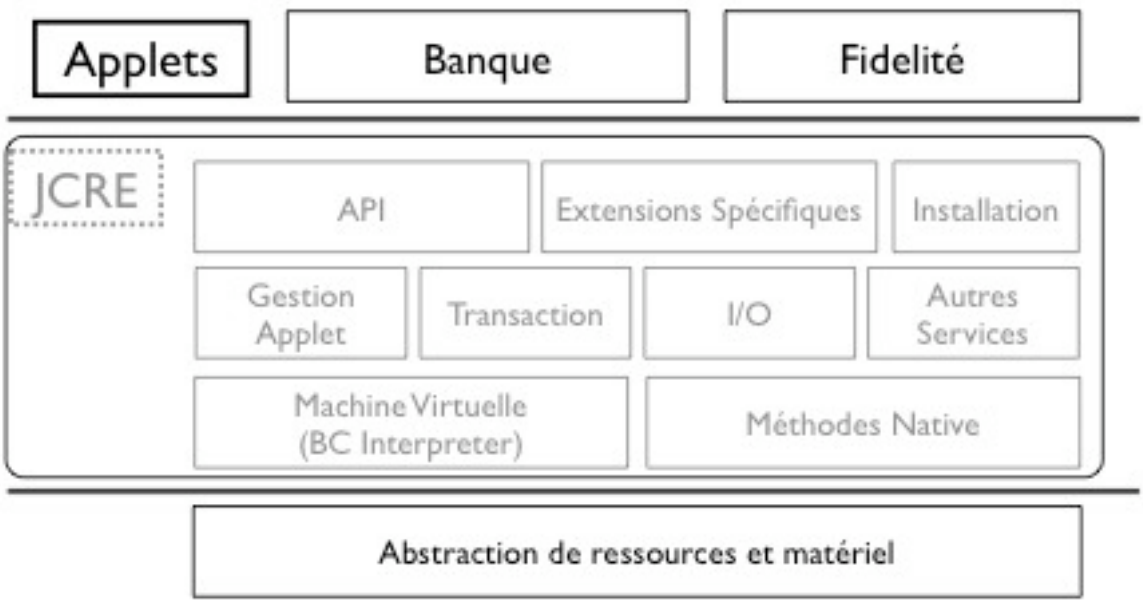
Le convertisseur

- Prend en charge les opérations faites par le class loader de la JVM :
  - Vérification de la conformité du format du Class File
  - Tester la conformité sur les aspects du langage Java
  - Initialisation des variables statiques
  - Résolution des références (classes, méthodes et champs) et mise sous forme compacte pour être plus efficace dans un petit système
  - Optimiser le BC
  - Allocation et création des structures qui représentent les classes dans la VM

L'interpréteur

- Il offre un support d'exécution au BC contenu dans le CAP file. Il permet ainsi aux applets chargées dans une carte de s'exécuter sur n'importe quelle plateforme.
- Il réalise :
  - Exécution du BC
  - Contrôle de l'allocation de la mémoire
  - Assure la sécurité (voir plus loin)
- L'installation d'applet est réalisée par un module en charge de cette action dans la carte mais aussi à l'extérieur pour la gestion du protocole (aspect sécurité XXX)

Architecture d'une Java Card



JCRE : cycle de vie et session carte

- En environnement station de travail, la VM Java est un processus, elle est donc initialisée au lancement puis à la du processus (arrêt VM) les objets en RAM sont perdus.
- Pour que les informations soient conservées d'une session à l'autre :
  - Dans le cas de la carte, l'initialisation de la VM est effectué une seule fois au "début de la vie de la carte", les objets et les données sont conservés en mémoire non volatile (NVM : EEPROM, Flash,...)
  - Lors de chaque session avec la carte :
    - Mise sous tension : le JCRE est "réactivé"
    - La carte reçoit et traite des commandes APDU (Process_APDU)
    - Mise hors tension : le JCRE est "suspendu"

Caractéristiques du JCRE

- Objets persistant et temporaire
  - Les objets Java Card sont par défaut persistants.
  - Pour des raisons d'efficacité (vitesse de lecture/écriture en NVM) et de sécurité (clé, résultat intermédiaire), les applets peuvent créer des objets temporaires.
- Opération atomique et transaction
  - La JCVM assure que pour tout les champs d'un objet ou d'une classe les écritures sont atomiques.
  - Le JCRE fournit une API aux applets permettant de regrouper plusieurs écritures et d'offrir la cohérence de ces écritures (Begin Transaction, Commit, Roll-Back)

Applet firewall et mécanisme de partage

- Chaque applet s'exécute dans un espace qui lui est propre.
  - Une applet ne peut avoir d'action sur une autre applet.
  - La JCVM doit assurer cette caractéristique.
- Il existe un mécanisme de partage, qui permet à une applet d'accéder à des services offerts par une applet ou par le JCRE.

Construction d'application JavaCard

- Une application carte
  - Code dans la carte
    - (application serveur = Applet Javacard)
  - Code dans le terminal
    - (application cliente)
- Une application construite en 3 étapes
  - Ecriture de l'application serveur (applet)
  - Installation de l'applet dans la javacard
  - Ecriture de l 'application cliente

Écrire applet Java Card

- Java Card API 2.1
- Etapes du développement d'une applet :
  - spécifier les fonctions de l'applet :
    - spécifier les AID
    - écrire le corps de applets
    - compiler (.class)
    - convertir (.cap)
    - charger dans la carte

Java Card - S

- Pour certaines applications la technologie est parfois trop "riche" et le coût de la licence élever par rapport aux prix final. Pour cela une version S a été introduite.
- La "Java Card S" permet aux licenciés de concevoir des cartes dont seule la fonction de chargement dynamique d'applet est absente. Ce qui permet de
  - Récupérer du code,
  - Utiliser des composant moins cher,
  - Diminuer les coûts de certification.

La Java Card 2.2

- Canaux logiques
- Suppression d'applets et de paquetage
- Mécanisme de suppression d'objet (explicite)
- Java Card Remote Method Invocation
- Support pour l'AES et des courbes elliptiques
- Support pour le sans contact

Conclusion

- Java Card donne un langage de programmation aux applications dan sla carte avec de bonnes propriétés
- L'infrastructure de chargement est définie par OP (Open Platform)...
- Une version 3.0 est en cours et devait apporter encore de progrès

Bibliographie

- http://java.sun.com/products/javacard/
- Technology for Smart Cards: Architecture and Programmer's Guide, Zhiqun Chen, Addison Wesley, September, 2000
- Understanding Java Card 2.0, Zhiqun Chen & Rinaldo Di Giorgio
  - http://www.javaworld.com/javaworld/jw-03-1998/jw-03-javadev.html
- http://javacardforum.org
- Java Card les produits des fabricants de cartes

Java Card Benchmark

pierre.paradinas @ cnam.fr
http://cedric.cnam.fr/mesure
Cnam/Cedric
Mobile and Embedded Systems Group

Agenda

- Motivations
- Some issues in the benchmark definitions
- Rules to apply in the definition of benchmark
- Presentation of the SCCB
  - Smart Card Cnam Benchmark
  - Initial project
  - Actual Project : MESURE
- Q&A

Motivations (Technical)

- Benchmark are new in the arena of smart card.
- Smart cards were designed to "provide a dedicated solution" :
  - Cards used in a domain answer to the requirement of it, performances are take into account as part of the application complete design.
- More and more :
  - Cards are "open platforms", programs or applets are added/loaded and executed on the platform :
    - There is a challenge to know the over all performances of platform and application running on it,
    - Evaluation of a platform performance is a first step.

SC manufacturers point of view

- Why benchmarks are important in the smart card arena :
  - Standards will be more and more important in the SC industry (like in Information Technology),
  - Differentiation are the necessary goals of company for their products, and more and more in the context where products are standardized.

SC manufacturer and differentiation

- Differentiation are on different product features :
  - Pricing,
  - Delivery times,
  - Quality,
  - Security,
  - ...
- and PERFORMANCE will be the next requirement.
- One element is not the differentiator but an element of a set of differentiator.

SC customers (users) point of view

- For application point of view :
  - Understand the platform performance :
    - Evaluation, prediction,
    - Set expectations,
    - Allow to choose the right product in a range of product.
  - How a service will be deliver in term of QoS is important.
  - How the PERFORMANCE are within my Applet ?
    - In term of execution time,
    - In term of memory consumption,
    - ...

Performance vs Security

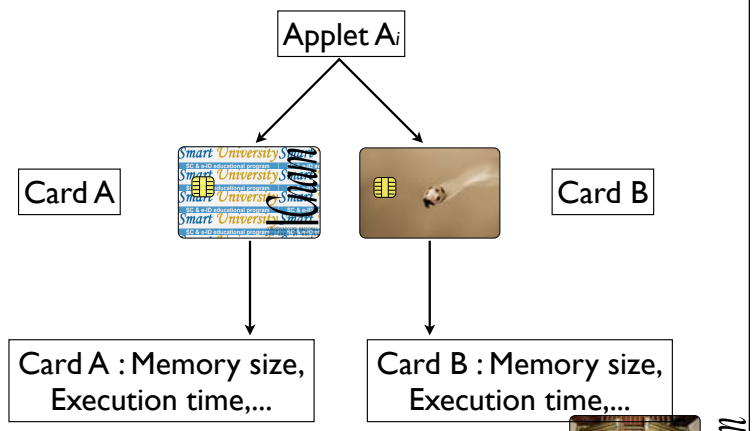
- Return to DES smart card implementation step

Feasability and fit-In

Optimization

Protection

Performance



Motivations (summary)

- This is the way, to emphasis your products as SC manufacturers,
- This is the way, to position your products as SC manufacturers,
- This is the way to compare and measure products as SC-users.

About some issues in benchmarking...

- How to define a benchmark,
- 3 approaches :
  - Measure on the system itself,
  - Simulation of the system,
  - Analytical model.

Measure of the system

- System is the Java Card.
- The more accurate approach is the measure of the system itself.
- This is also the measure with the high level of "trust".

Measure of the system (Architecture)

Computer :
- Loads and manages the applets,
- Drives the reader,
- Collects results.

Simulation model

- A program represents card features

Java Card Simulator

Pros/Cons of simulation approach

- Pros :
  - As it is a program, it is easy to improve and modify the model.
  - The cost is link to the level of detail included in the model.
- Cons :
  - Difficult to simulate every single and basic functions,
    - BtW, cache memory and NVM simulation is not quite easy task!
    - Accuracy is less than direct measure.

Approach and model comparison*

	Analytical Model (+)	Simulation	Measurement
Flexibility	High	High	Low
Cost	Low	Medium	High
"Trust"	Low	Medium	High
Accuracy	Low	Medium	High

* From David J. Lilja Book. (+) based on mathematical description of a system

What we may measure ?

- In the Java Card context :
  - Execution time of a command received by the card,
  - Size of elements (memory consumption),
  - Ability to support function like :
    - Card usage (durability).
    - Card pull out (atomicity),

What are there features ?

- Measures must be :
  - Repeatable,
  - Easy to realize,
  - Consistency and reliability (if MetricA<MetricB with an application card B is better than card A),
  - Linear (If new card is 2 time faster with the metric, an application may run 2 time faster also),
  - Independence, there is no "optimize" Java Card with the goal to be "better" on the benchmark.

Clock rate, Mips, ..., APDUs

- On a computer clock rate, Mips and Mflops are not acceptable measures of performances.
- If we define APDUs as number of APDUs manage by a card per second ?
  - It is easy to realize but not acceptable!
- Specification ideas :
  - A set of program are executed on a system we want to measure,
  - Each program result is divided by the time obtain by a reference system,
  - A geometric average is produce and give a number as performance metric.

What we propose

- An open benchmark of Java Card performances,
- Based on an accurate technical solution,

Our motivations :

- There is "no" benchmark to day in smart card industry.
  - What means "no" :
    - It is not public,
    - It is not accepted,
    - It is not opened.
  - Benchmarks exist in :
    - R&D department of smart card manufacturer,
    - Some smart cards users organization.

Our approach in the project

- Inventory of measurable performances,
- Applet reference design and development,
- Performance evaluation in term of time and memory consumption,
- Set up a project proposal to get funding from public research funding
- Comparison with existing in house benchmark,
- Benchmark tools packaging as an open software,
- Publishing on the web.

Our approach in the project

- Inventory of measurable performances,
- Firts applet reference design and development,
- Performance evaluation in term of time, memory consumption and power (in progress 1st step of card collection),
- Set up a project proposal to get funding from public research funding : MESURE Project (see end of presentation).

P. Paradinas -

What is necessary to benchmark

- In Java Card Technology :
  - Performance,
  - Security,
  - Compatibility in term of interoperability.
- Our goals and project is to provide open benchmark on the first point.
- Others are in out of our scope.

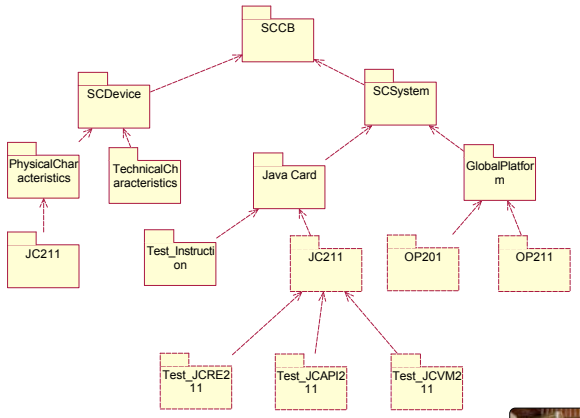
P. Paradinas -

Not an easy task

- Hardware configuration :
  - NVM :
    - EEPROM/FLASH/FeRAM,...
  - Communication support
    - CL, Serial, USB, MMC,..
- OS :
  - memory model,
  - VM implementation model.
- Garbage collector.
- But this is the challenge !

P. Paradinas -

Smart Card Cnam Benchmark



SCCB measures

- SCDevice checks hardware function like :
  - Technical features : information received and interpret from the ATR,
    - memories (test on NVM features) :
      - available size of memory, size of object,...
      - evolution of behavior during “memory life cycle”
        - GC (if exist),
        - durability.
    - stack size (number of call).

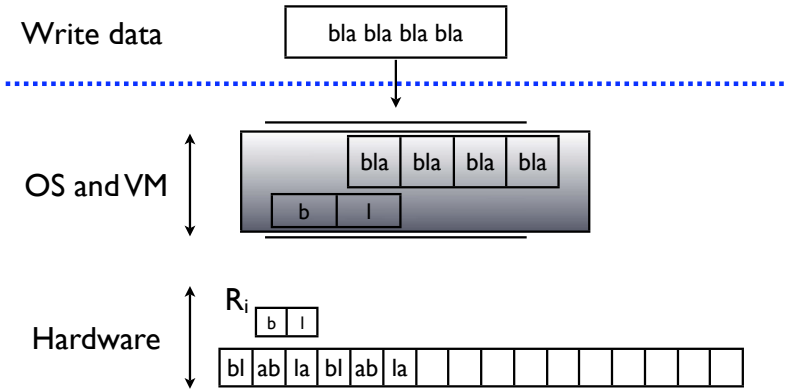
SCCB measures (Cont’d)

- SCSystem evaluates performance of JC and OP (Each packages possible to evaluate are measured) :
  - Tests instruction (while, if,...)
  - Others instructions are also implemented.
    - ArithmeticMetric,
    - LogicalMetric,
    - LoopMetric,

Function write memory

- Write memory is :
  - Long operation compare to read,
  - Flash from ST22 (STM) :
    - 32-bit word Erase in 2 ms typical
    - 2K byte sector erase in 50 ms typical
  - Measure have to take into account the size of buffer or/and cache.
    - A "memory write time" have to be executed with different size of buffer.

Write memory and buffers size



Evaluation context

- Based on Windows platform and XP :
  - Windows XP used with action :
    - To reduce and avoid constraints due to the Windows OS latency,
    - PCs are not connected (autonomous),
- Variance on commands (command are repeated and an average calculated),
- Different PC platforms and readers will be used.

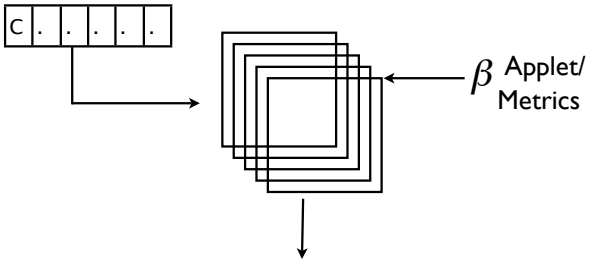
USB and high precision reader

- We develop the benchmark for 2 class of reader :
  - simple and cheap USB reader :
  - test and evaluation reader (Microprosss MP 300)

- Assumption :
  - Result CardA < CardB with the 2 classes of reader are equivalent

P. Paradinas -

SCCB references



* Σ (α.metrics) / nbcards = 100
* A card will be ranked after a complete test of Applet/Metrics compare to base 100

P. Paradinas -

How to set up a base 100 ?

- For an application domain what are the features of the card used and with what ratio.
- For each features what are the relevant measure and ratio.
- For a features *i* : $M_{i_1} .. M_{i_n}$
- We note F_i , the performance of a platform for the feature *i,* where :

$$F_i = \sum_{j=1}^{n_j} \beta_{ij} \times M_{ij}$$

P. Paradinas -

How to set up the base 100 ? (Cont'd)

- There β are independent of the domain (they are related to chip, VM and API implementation).
  - We want to avoid that a "cache" is the only part of the memory activated and in that case produce better result than a card where there is a lot of memory page default.
- For a domain k the $F_1..F_f$ are the performances of each features.
  - These performances have to be aligned with the goal to be representative of the domain.

$$P_k = \sum_{i=1}^{f} \alpha_{ik} \times F_i$$

The base 100

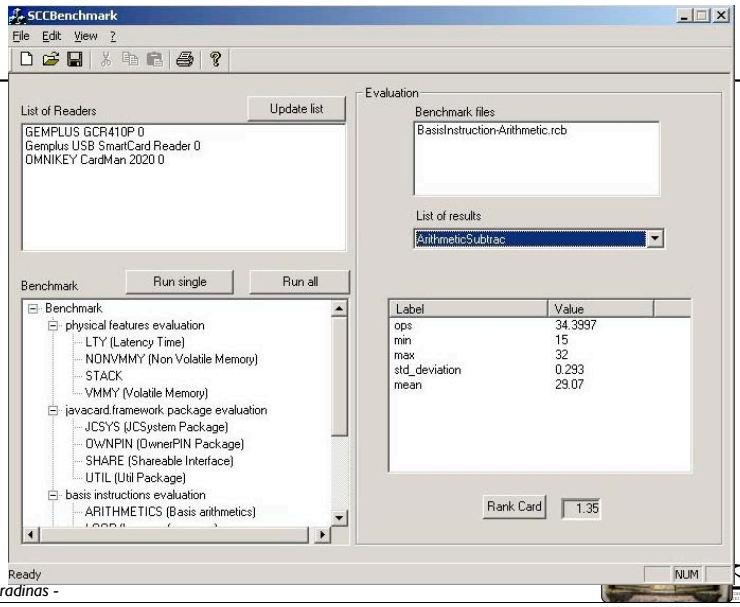
- The base 100 is obtained for a domain *k* with :

$$R_k = \sum_{i=1}^{n} \frac{P_i}{n}$$

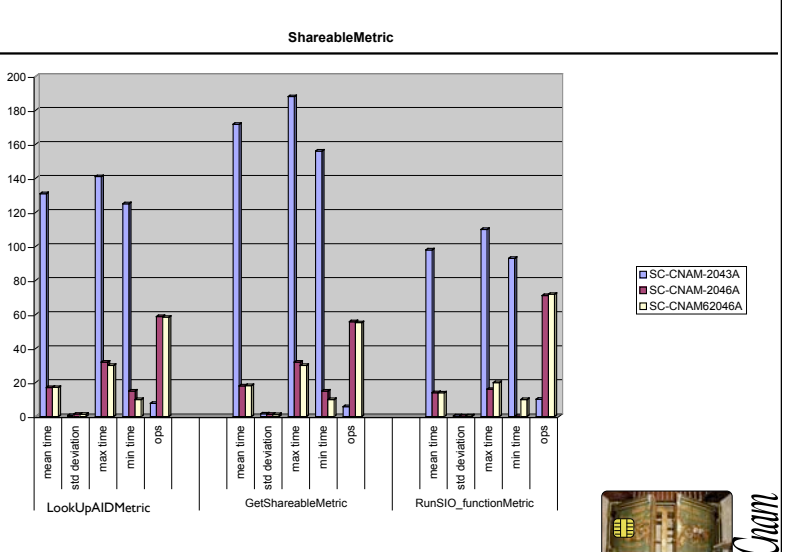
- It is important to verify that the result of card from a domain is consistent with applet of a domain.

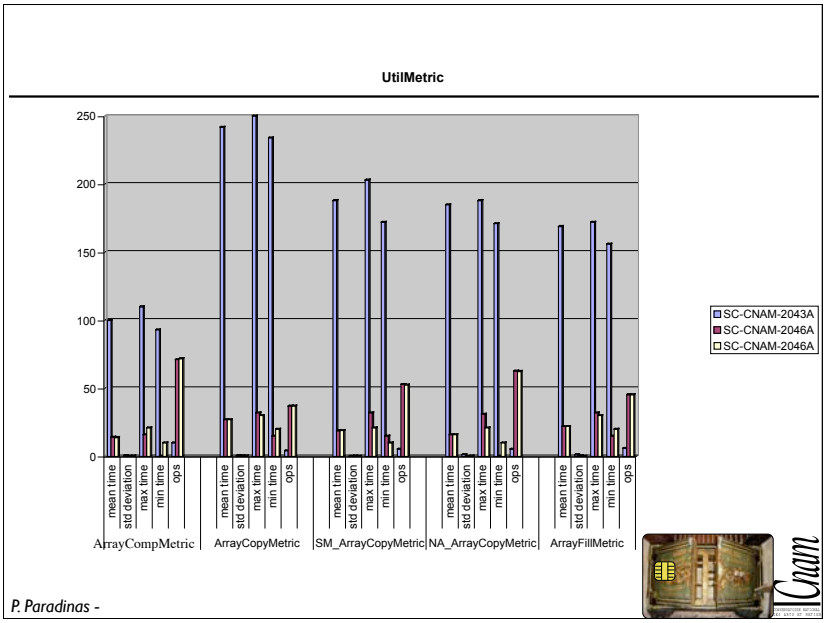
About domain adequation

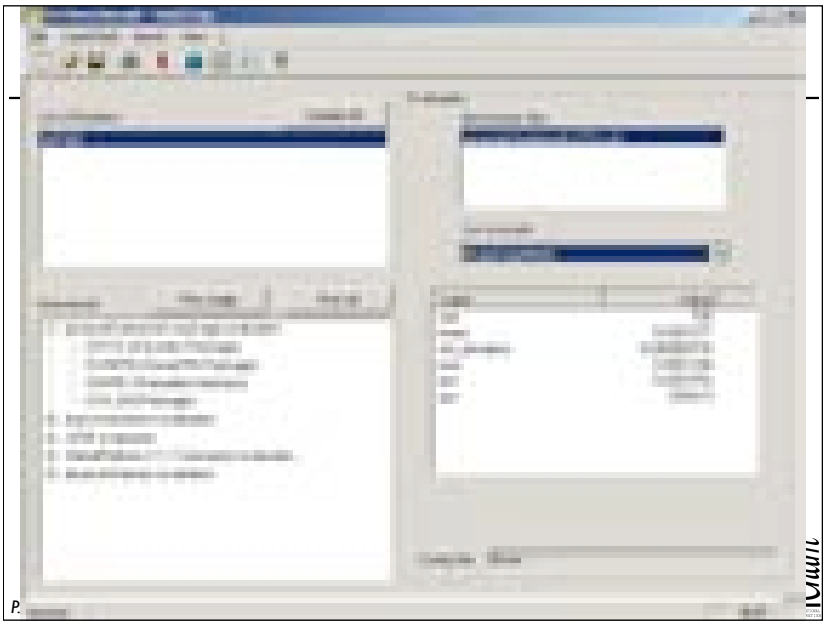
- Each domain needs specific requirements :
  - Payment :
    - Transaction management
  - CL :
    - Transaction
  - ID :
    - Data transfert, perform crypto function
  - GSM :
    - File management

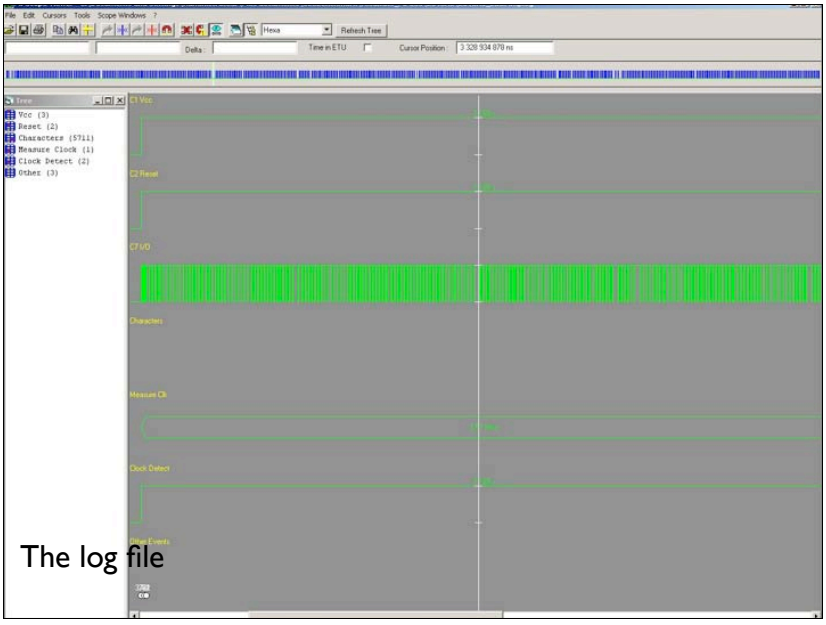


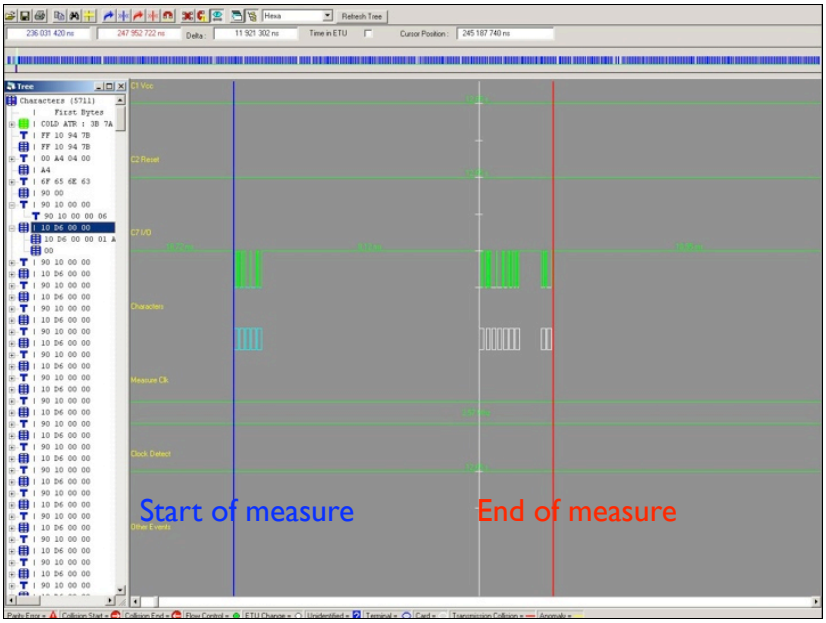
Some intermediate results




UtilMetric
250
200
150
100
50
0
SC-CNAM-2043A
SC-CNAM-2046A
SC-CNAM-2046A
mean time
std deviation
max time
min time
ops
ArrayCompMetric
ArrayCopyMetric
SM_ArrayCopyMetric
NA_ArrayCopyMetric
ArrayFillMetric
P. Paradinas -


P.


The log file



The MESURE project

- MESURE is a project funded by french administration :
  - http://www.gip-anr.fr/ & http://www.rntl.org/
    - Embedded systems area selected in 2005.
- Project partners :
  - CNAM/Cedric (Paris), academic partner and project leader,
    - *Pierre Paradinas.*
  - RD2P (Lille University),
    - *Gilles Grimaud.*
  - Trusted Labs, (Sophia Antipollis),
    - *Eric Vétillard.*

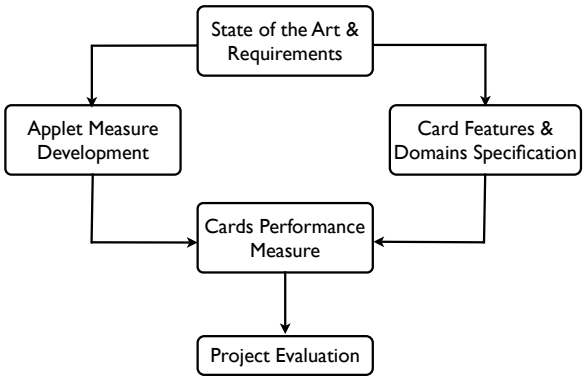
P. Paradinas -

Project goals

- Propose a tools for performance and features Java Card evaluation,
- Delivering an open source platform for Java Card Benchmark,
- Establish a reference in term of performance measurement.

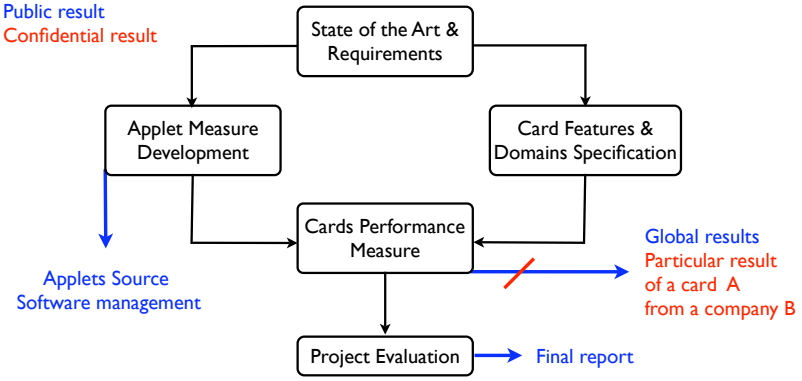
P. Paradinas -

Projects chart



Project deliverables

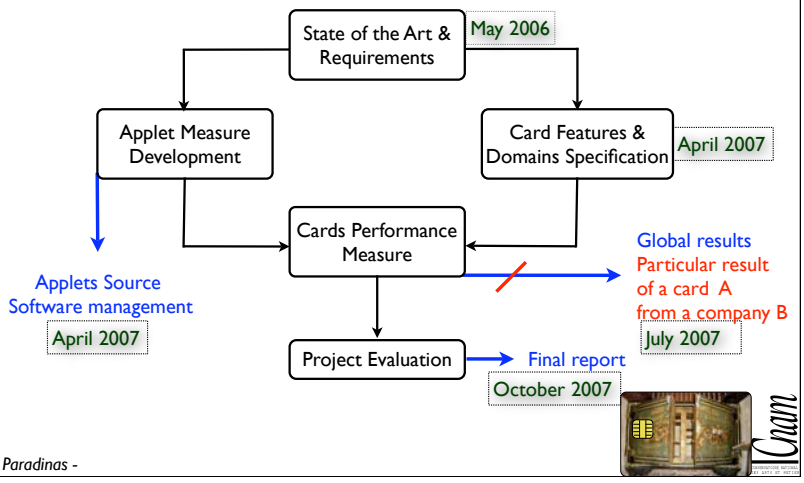
Color code :
Public result
Confidential result



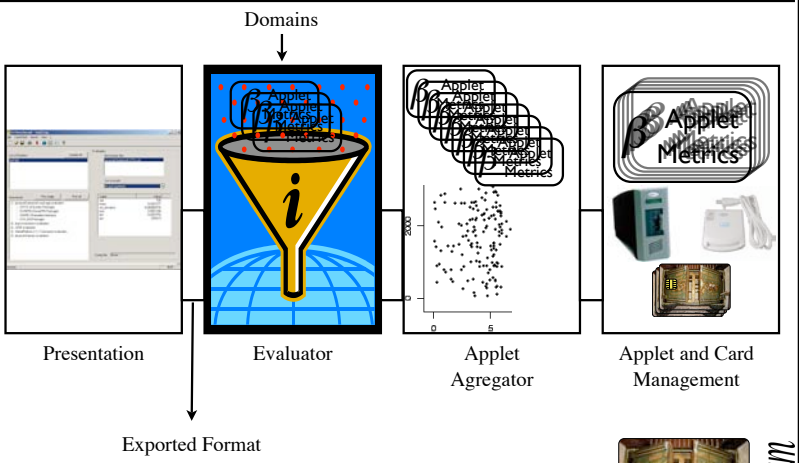
What we will not perform...

- Java Card Technology specification conformance testing,
  - Related to SC-manufacturers, SUN Test Suite.
- Security evaluation,
  - CC, customer evaluation requirement
- Measure and publication on one product results.
- Global base 100, all applets and how the base is calculated will be published

Project schedule



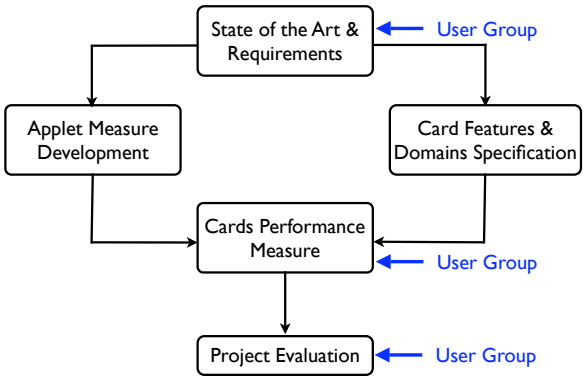
Software Architecture



User group

- To reinforce adoption,
- To get feedback,
- To disseminate results,
- The project will launch user group.
  - *The first meeting will take place @ E-smart (Sophia Antipolis, September 2006)*

Meeting point within user-group



Project contacts

- Web page :
  - http://cedric.cnam.fr/mesure/
- Project :
  - Pierre . Paradinas @ cnam . fr

Benchmark value added

- Product and new generation product (or standard) **differentiation**,
- Plan improvement on new implementation,
- Provide a user point of view,
  - Product performances comparison and evaluation.

New era for research...

- Benchmark may help on QoS,
- Benchmark may help on consumption with a better understanding of performance (locally and globally).
- Open question is how "security may be measured and ranked",

Thank you.

Virtualisation de Système et Sécurité

pierre.paradinas @ cnam.fr
http://cedric.cnam.fr
Cnam/Cedric
Mobile and Embedded Systems Group

Agenda

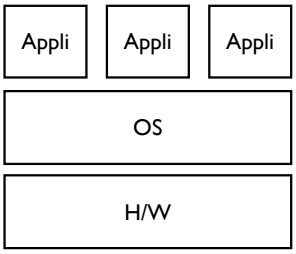
- Virtualisation de système :
  - Tendance,
  - Exemple.
- Sécurité dans la virtualisation.
- Conclusion et discussions.

La virtualisation

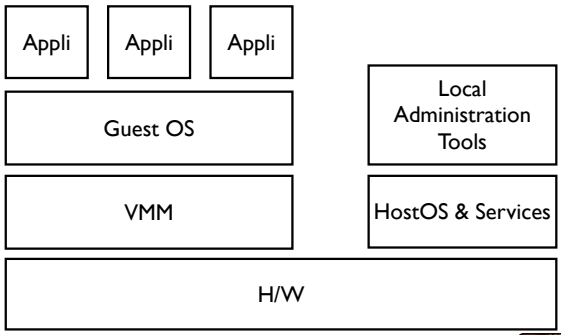
- De plus en plus présente, phénomène de mode ?
- Mais ceci est pas nouveau...
  - L'architecture IBM 360 offrait déjà des notions de machine virtuelle.

Architecture

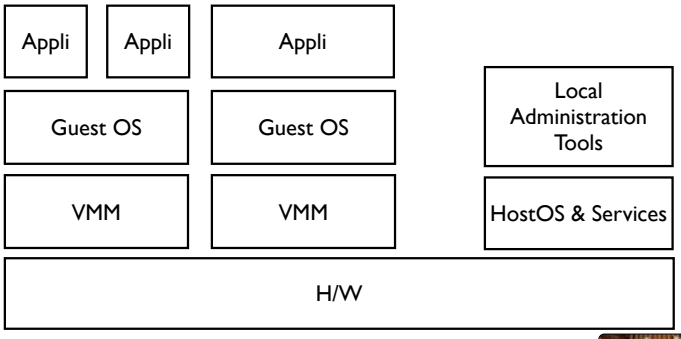
- Sans virtualisation... classique:)



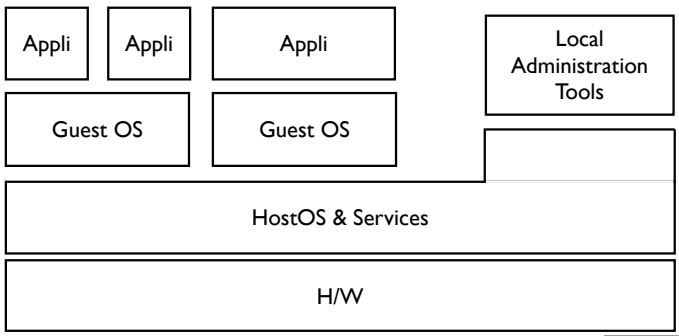
Architecture de virtualisation



Architecture de virtualisation



Architecture de virtualisation



Virtualisation...

- Une machine virtuelle est exécutée sur une machine réelle.
- Une machine virtuelle (programme) est constituée d'une suite binaire :
  - correspondant au code de la machine (algorithme),
  - et à l'état de celle-ci (data structure).
    - Algorithms + Data Structures = Programs (N.Wirth).

Type de virtualisation

- Full-virtualization :
  - Le Guest-OS n'est pas modifié,
  - Les appels systèmes sont traités par le VMM (Virtual Machine Monitor)
  - Exemple :
    - WMWare Server
- Para-Virtualization
  - Le Guest-OS est modifié,
  - Les appels systèmes sont ceux offerts par le VMM
  - Exemple :
    - XEN, WMWare Server

Intérêts/Inconvénients de l'approche

- Approche facilement déployable, grande extensibilité :
  - "*copier*" le fichier puis "*lancer*" son exécution...
    - Vous disposer alors d'un nouveau serveur !
- Du point de vue de l'administration et des coûts de maintenance :
  - Moins de machine physique,
  - Meilleure utilisation de la puissance des machines,
  - Administration plus centralisé pour certains aspects.

Intérêts/Inconvénients de l'approche

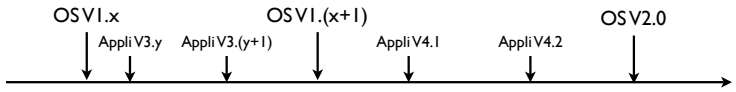
- Le nombre de serveur peut augmenter très vite et fortement.
- Plus vite que la puissance des machines et des capacités de stockage.
- Pour les administrateurs de "parcs" informatique ou de "système de traitement" de l'information la tâche est rendu plus lourde et complexe.
  - L'administration est un mélange de procédure automatique et manuelle.
  - Les serveurs (réels ET virtuels) sont en augmentation
    - ==> Charge et risque en augmentation.

Risques

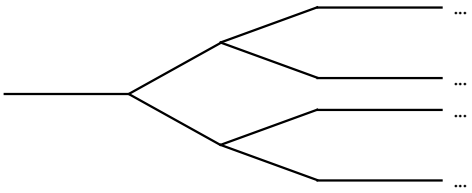
- Revue de risques liés à l'approche.

Risques liés au cycle de vie

Système classique : évolution linéaire



Virtualisation : évolution arborescente



Risques liés au cycle de vie

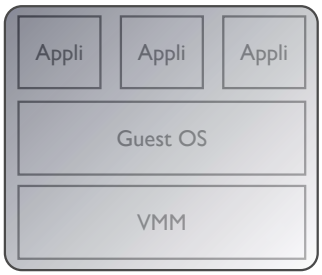
- La cohérence d'un parc informatique est difficile à obtenir dans un contexte de virtualisation sur une même machine on peut avoir des occurrences différentes d'un OS et/ou d'une application dont une pourra être en retard en terme de faibles par rapport à des attaques spécifiques.
- Les VMs disposent de point de reprises (rollback, mobilité), qui eux aussi pourront réduire des efforts de correction par retour à une situation mauvaise.

Risques liés au mécanismes

- Par exemple de le cas de :
  - protocole du type : S-Key,
  - Générateur aléatoire et sa fraîcheur
  - ...
- En fait, la capacité à interrompre puis reprendre une exécution doit être très "encadrée" :
  - ==> Par exemple avec des notions de non-roolback ou de roolback à des points "obligés"

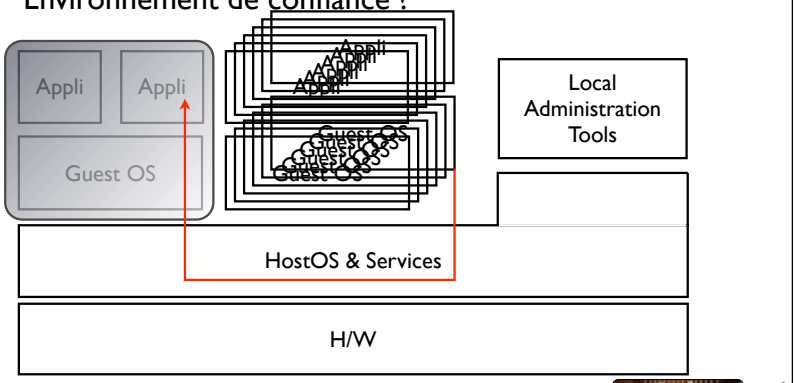
Risques liés à la mobilité/complexité

Environnement de confiance



Risques liés à la mobilité/complexité

Environnement de confiance ?



Autres risques

- Lien entre machine physique et entité
  - Via les adresses MAC, n° de machines et listes des employés, on peut s'y retrouver.
  - Avec une approche par VM.....
- Cycle de vie des données
  - Dans le cas ou les machines virtuelles sont transportées elles le sont en l'état et avec des données intermédiaires parfois très intéressantes.
    - Un peu comme des attaques sur le cache.
- Dans un parc informatique, les nomades (!) sont souvent la porte d'entrée (virus), comme une blessure/plaie en médecine...

Des offres disponibles

- WMWare, MS, INTEL,...

Windows Server System

Linux-VServer

Intel

vmware

Xen Source

OS X : Virtual PC

P. Paradinas -

Biblio

- When Virtual is Better than Real
  - Peter Chen, Brian Noble
- Xen and the Art of Virtualization
  - Paul Barham et al.
- When virtual is harder than real : security challenges in virtual machine based computing envirinnements.
  - T. Garkinkel & M. Rosenblum

P. Paradinas -

Cryptosystèmes basés sur l'identité

Sandra Marcello

Cryptologue à Thales
France

Sur la cryptologie basée sur l'identité (Identity Based Encryption)

Sandra Marcello, Thales

Les origines : Shamir 1984

Contenu de l'article:

- Shamir pose le problème de l'existence de solutions pour chiffrer et signer des messages en utilisant comme clef publique le nom du destinataire du message ou de la personne qui a signé le message .
- Shamir propose une solution partielle à savoir un schéma de signature basée sur l'identité.

2001 .....

En 2001, deux schémas de chiffrement IBE ont été proposés, l'un par Boneh et Franklin l'autre par Cocks.

Nous nous intéresserons à celui de Boneh et Franklin qui est illustré sur des courbes elliptiques à l'aide du couplage de Weil.

La première solution IBE complète date de 2001, en effet du schéma de chiffrement de Boneh-Franklin il est possible de dériver un schéma de signature.

Contexte de l'IBE :chiffrement asymétrique

2 Clefs différentes :

- 1 clef pour chiffrer (clef publique :le nom)
- 1 clef pour déchiffrer (clef privée)

Processus de génération de clefs

- Cas des systèmes de chiffrement à clefs publiques Principes possibles:
  1. Choix aléatoire d'une clef privée
  2. Construction de la clef publique à partir de la clef privée et d'une "fonction à sens unique"

"Processus" de génération de clefs (suite)

- Cas des systèmes de chiffrements IBE
  1. Postulat : la clef publique est le nom du destinataire du message
  2. Nécessité d'une autorité de confiance appelée PKG (Private Key Generator).
     Le PKG génère aléatoirement (pour lui-même) une clef : appelée clef maître (master key) mk.

"Processus" de génération de clefs (fin)

3. Le PKG génère la clef privée du destinataire à l'aide:
   - du nom du destinataire (sa clef publique)
   - de la clef maître
   - et d'une "fonction à sens unique"

   Seul le PKG peut générer la clef privée.

Fonctionnalités apportées par l'IBE

- "Atomisation possible" : concaténation du nom et la date du jour (limites de la méthode requêtes quotidiennes de clefs privées..)
- Délégation des clefs de déchiffrements (pour un laptop, pour déléguer des taches .....)
- chiffrement "forward": possibilité de lire un message dans le futur à une certaine date.

Cadre du schéma de Boneh-Franklin

Boneh et Franklin illustre leur schéma avec un exemple dans le cas des courbes elliptiques en utilisant le couplage de Weil. Ce schéma se place dans la lignée d'une utilisation constructive des couplages de Weil (et de Tate) en cryptologie. LA première utilisation constructive est due à Joux pour un protocole tripartite d'échange de clefs.

Le schéma de Boneh-Franklin est défini pour toute application bilinéaire admissible.

Les couplages de Weil et de Tate peuvent être utilisés pour construire des applications bilinéaires admissibles utilisées pour les schémas IBE.

Utilisation des couplages de Weil et de Tate en cryptographie avant l'IBE.

- Réduction MOV et FR
- Protocole d'échange de clefs tripartite de Joux (2000)

Définition d'une application bilinéaire admissible

Soit $(G,+),(G_1,.)$ deux groupes cycliques d'ordre premier q avec pour éléments neutres $1_G, 1_{G_1}$. Une application $\hat{e} : G \times G \Rightarrow G_1$ est appelée application bilinéaire admissible ssi elle vérifie les propriétés suivantes:

- $\hat{e}$ est une application bilinéaire *i.e*:

$$\forall g \in G, \forall a,b \in (\mathbb{Z}/q\mathbb{Z})^* \quad \hat{e}(ag,bg) = \hat{e}(g,g)^{ab}.$$

- $\forall g \in G$, avec $g \neq 1_G$: $\hat{e}(g,g) = g_1 \neq 1_{G_1}$.
- l'application $\hat{e}$ est efficacement calculable.

Composition d'un schéma de chiffrement IBE

Il se compose de 4 algorithmes.

1. **Setup**: Le PKG avec comme entrée le paramètre de sécurité k génère les paramètres publics notés ***params*** et génère aléatoirement sa clef maître mk.
2. **extract**: Le PKG génère la clef privée du destinataire du message.
3. **Encrypt**: Le rédacteur du message, chiffre le message à l'aide des paramètres publics et de la clef publique du destinataire
4. **Decrypt**: Le destinataire du message déchiffre le message à l'aide de sa clef privée et des paramètres publics.

Remarque

L'ordre des étapes 2 et 3 peut être inversé.

Schéma de chiffrement de Boneh-Franklin (simplifié)

- **Setup :**L'entrée est un entier k (appelé paramètre de sécurité). Les étapes sont les suivantes:
  1. Un nombre premier q (dont l'écriture en binaire est de longueur k) est génère aléatoirement. Sont générés également, deux groupes $(G,+),(G_1,.)$ d'ordre q, une application bilinéaire admissible $\hat{e}: G\times \rightarrow G_1$. Un générateur $P \in G$ est choisi aléatoirement.
  2. Choix aléatoire de $s \in (\mathbb{Z}/q\mathbb{Z})^*$ et calcul $P_{pub} = sP$.

Setup: (fin)

3. Choix de fonctions de hachage cryptographique
   $H_1 : \{0,1\}^* \rightarrow G^*$ et $H_2 : G_1 \rightarrow \{0,1\}^n$ pour n (longueur du message à chiffrer).

- L'espace des message est : Me= $\{0,1\}^n$.
- L'espace des chiffrés est : Ch= $G^* \times \{0,1\}^n$.
- Le système de paramètre est :
  $params =< q, G, G_1, \hat{e}, n, P, P_{pub}, H_1, H_2 >$.
- La clef maître est $mk = s \in (\mathbb{Z}/q\mathbb{Z})^*$.

- **Extract:** Pour une identité donnée $\mathbb{ID} \in \{0,1\}^*$. Les étapes sont les suivantes :
  1. Calculer $Q_{\mathbb{ID}} = H_1(\mathbb{ID}) \in G^*$
  2. Comme s est la clef maître, on calcule la clef privée associée à $\mathbb{ID}$ et notée :
  $$d_{\mathbb{ID}} = sQ_{\mathbb{ID}}.$$

- **Encrypt:** Chiffrement d'un message $M \in$Me avec la clef publique $\mathbb{ID} \in \{0,1\}^*$. Les étapes sont les suivantes :
  1. Calculer $Q_{\mathbb{ID}} = H_1(\mathbb{ID}) \in G^*$.
  2. Choix aléatoire $r \in (\mathbb{Z}/q\mathbb{Z})^*$.
  3. Détermination du chiffré C avec
  $g_{\mathbb{ID}} = \hat{e}(Q_{\mathbb{ID}}, P_{pub}) \in G_2^*$,
  $$C =< rP, M \oplus H_2({g_{\mathbb{ID}}}^r) > .$$

- **Decrypt :** Le résultat du chiffrement du message M avec la clef publique $\mathbb{ID}$ est noté : C=$< U, V >\in$ Ch. Pour déchiffrer C à l'aide la clef privée $d_{\mathbb{ID}}$ on calcule:
  $$V \oplus H_2(\hat{e}(d_{\mathbb{ID}}, U)) = M.$$

Schéma de chiffrement de Boneh-Franklin ("original")

- **Setup :** Même algorithme que le **Setup** du schéma simplifié avec en plus le choix de deux fonctions de hachages cryptographiques :
  - $H_3 : \{0,1\}^n \times \{0,1\}^n \rightarrow (\mathbb{Z}/q\mathbb{Z})^*$,
  - $H_4 : \{0,1\}^n \rightarrow \{0,1\}^n$.

- L'espace des message est : Me= $\{0,1\}^n$.
- L'espace des chiffrés est : Ch= $G^* \times \{0,1\}^n \times \{0,1\}^n$.
- Le système de paramètres est :
  params=$< q, G, G_1, \hat{e}, n, P, P_{pub}, H_1, H_2, H_3, H_4 >$.
- La clef maître est $mk = s \in (\mathbb{Z}/q\mathbb{Z})^*$.

- **Extract:** Identique à celui du schéma simple

- **Encrypt:** Chiffrement d'un message $M \in$Me avec la clef publique $\mathbb{ID} \in \{0,1\}^*$. Les étapes sont les suivantes :
  1. Calculer $Q_{\boldsymbol{D}} = H_1(\mathbb{ID}) \in G^*$ (étape identique).
  2. Choix aléatoire $\sigma \in \{0,1\}^n$.
  3. Détermination du chiffré C avec
     $g_{\boldsymbol{D}} = \hat{e}(Q_{\boldsymbol{D}}, P_{pub}) \in G_2^*$, (comme plus haut)

$$C =< rP, \sigma \oplus H_2(g_{\boldsymbol{D}}^r), M \oplus H_4(\sigma) > .$$

- **Decrypt :** Le résultat du chiffrement du message M avec la clef publique $\mathbb{ID}$ est noté : C=$< U, V, W >\in$ CH. Pour déchiffrer C à l'aide la clef privée $d_{\boldsymbol{D}}$. Les étapes sont les suivantes :
  1. Calculer $V \oplus H_2(\hat{e}(\mathbb{ID}, U)) = \sigma$.
  2. Calculer $W \oplus H_4(\sigma) = M$.
  3. Calculons $r = H_3(\sigma, M)$. Regarder si $U = rP$. Si cette condition n'est pas vérifiée, rejeter le chiffré.
  4. M est le mess sage déchiffré.

Sécurité

La sécurité du schéma de chiffrement de Boneh-Franklin repose sur une hypothèse analogue à celle de Diffie-Hellman. Il s'agit de l'hypothèse Bilinéaire de Diffie-Hellman.

Schéma de signature dérivé du schéma de Boneh-Franklin

Naor a observé que de tout schéma IBE on peut déduire un schéma de signature.

L'idée est la suivante:

- La clef privée du schéma de signature est la la clef maître du schéma.
- La clef publique est le système de paramètres *Params* du schéma IBE.
- La signature pour un message M est le chiffrement pour l'identité ID=M.
- Pour vérifier la signature, on choisit un message aléatoire M' que l'on chiffre avec l'identité M, et on tente de déchiffrer avec la signature de M.

L'algorithme de déchiffrement est ici randomisé.

Le schéma de signature obtenu à partir du schéma de B-F produit des signatures courtes.

Schéma de chiffrement de Waters 2005

Schéma de chiffrement sans fonctions de hachage....

Idée: identité est un vecteur dont les bits non nuls sont utilisés pour choisir les bits d'un vecteur aléatoire qui vont être utilisés pour construire la clef privée.

Motivation des schémas HIDE (Hierachical ID Encryption)

- La génération par le PKG, dans le cas des schémas IBE, des clefs privées est coûteuse, et nécessite des canaux sûrs pour pour transmettre les clefs.
- On n'a besoin que des paramètres publics de la racine de Bob pour lui écrire.
- Le PKG doît vérifier les identités.
- Quand un noeud est compromis seul une partie de l'arbre est compromise et non tout le système.

Description des schémas HIDE

notion de ID-uplet: définit la position de l'utilisateur dans la hiérarchie, $(ID_1, \cdots, ID_t)$.

Ils sont constitués de 5 algorithmes.

1. **Root Setup:** Le PKG racine prend en entrée un paramètre de sécurité et donne en retour le système public de paramètres *Params* et la clef secrète de la racine. *Params* contient une description de l'espace des messages et de l'espace des chiffrés.
2. **Lower-level Setup :** À l'aide de *Params*, un Lower-level PKG peut générer un "lower-level secret" ou bien un secret aléatoire utilisable une seule fois pour chaque extraction.
3. **Extraction:** Un PKG peut générer une clef privée pour l'un de ses "enfants" à l'aide de *Params* et de sa clef secrète.

4. **Encryption :** Chiffrement à l'aide de *Params* et de l'ID-uplet.
5. **Decryption:** Déchifrement à l'aide de *Params*, C et la clef privée du destinataire.

Comment échanger une clef après 30 ans de Diffie-Hellman ?

Serge Vaudenay

Professeur à l'EPFL
Suisse

How to Agree on a Key after 30 Years of Diffie-Hellman?

Communication Security and Manual Key Establishment

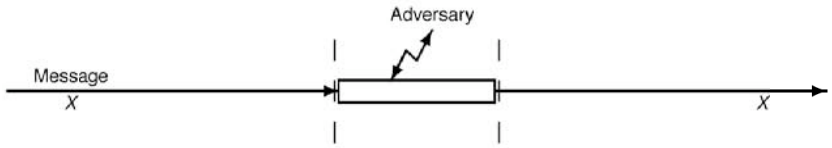
Serge Vaudenay

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

http://lasecwww.epfl.ch/

LASEC

Basic Security Properties

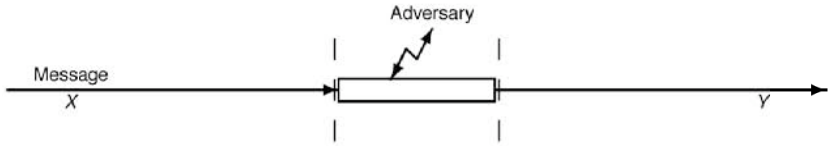


- **Confidentiality** (C): only the legitimate receiver can get X
- **Authentication** + **Integrity** (A+I): only the legitimate sender can insert X and the received message must be equal to X

Several Levels of Confidentiality

- **Weak confidentiality**: the adversary cannot retrieve X
- **Confidentiality with respect to a hard-core function** f: given a function f on the message space, the adversary cannot guess $f(X)$ with a significant advantage
- **Semantic security**: the adversary cannot propose two disjoint sets of messages so that she can guess to which X belongs when a challenger encrypts a random message from one of these two sets

Malleability: Authentication without Integrity

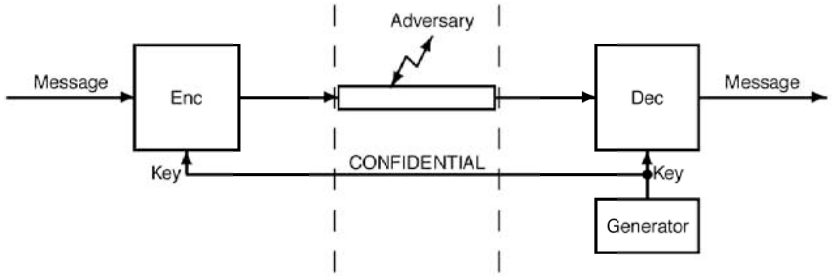


An adversary can replace X by some Y such that $X \sim Y$
(The Receiver only has insurrance that *some* message was sent by the right sender.)

Confidentiality without Integrity or Authentication

- **Confidential but Non-authenticated**: the adversary cannot read a sent message, but she can insert a message so that the receiver can receive an X of her choice
- **Confidential but Non-integer**: the adversary cannot insert a message of her choice but can modify a sent message so that the receiver will receive some Y such that $X \sim Y$ even though the adversary does not learn X and Y
  Example: the adversary can replace X by $Y = X \oplus \Delta$ for a Δ of her choice even though she cannot get any informaiton about X

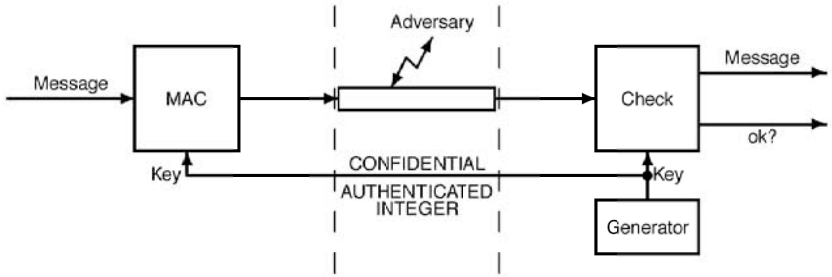
Confidentiality by Symmetric Encryption



Two Categories of Symmetric Encryption

stream ciphers	block ciphers
RC4	DES
GSM–A5/1	3DES
Bluetooth–E0	IDEA
DVB-CSA	BLOWFISH
...	RC5
	AES
	KASUMI
	SAFER
	CS-Cipher
	FOX
	...

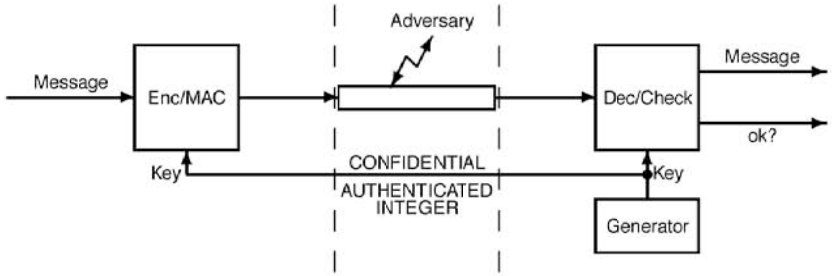
Authentication and Integrity by MAC



Three Categories of MAC

from stream ciphers	from block ciphers	from hash functions
Wegman-Carter	EMAC	**HMAC**
LFSR-Toeplitz	XCBC	UMAC
bucket hashing	RMAC	...
square hash	TMAC	
...	**OMAC**	
	...	

A+I+C by Combined Mode of Operation



Several Combined Modes of Operation

CCM
CS
CWC
EAX
GCM
IACBC
IAPM
OCB
PCFB
XCBC
...

Other Security Properties (Message Level)

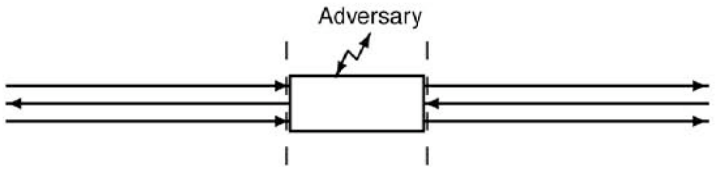
- **Freshness**: the received X was not received before
- **Liveliness**: when a message is sent, it will eventually be delivered
- **Timeliness**: (stronger liveliness) time of delivery is upper bounded

Various Human-to-Human Communication Channels

	encounter	telephone	handwritten mail	email
authentication	☺	☺	☺	☹
integrity	☺	☺	☺	☹
confidentiality	☺	☹	☹	☹
freshness	☺	☺☹	☺	☹
liveliness	☺	☺	☹	☹

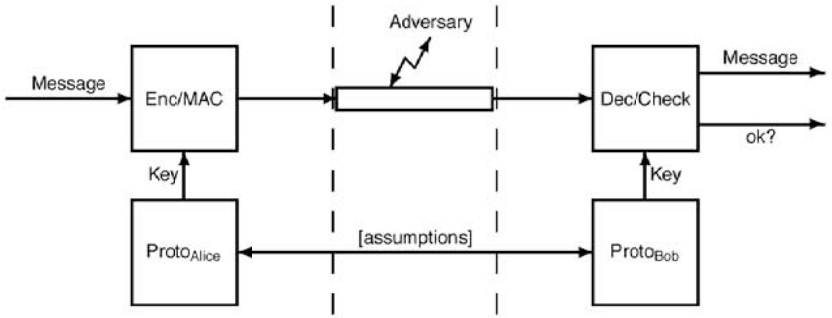
"the more human, the more secure"

Security at the Protocol Level: Session Setup + Integrity

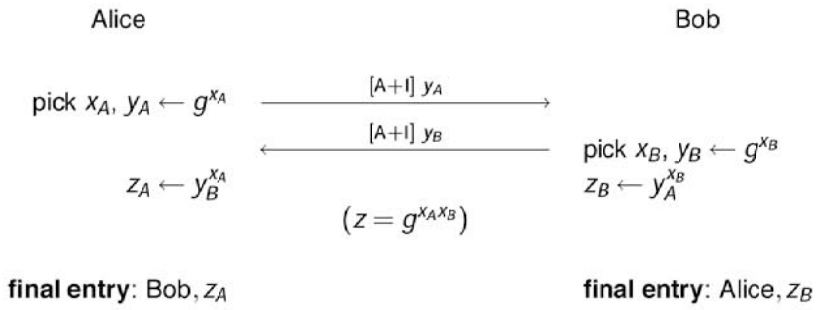


- **Key establishment**: set up A/I/C key material for message security
- **Sequentiality**: whenever a participant has seen a message sequence starting with $X_1, \ldots, X_t$, X_t coming in, then the other participant has seen a message sequence whose first t messages are $X_1, \ldots, X_t$
- **Termination fairness**: making sure that the last message on both ends is the same one

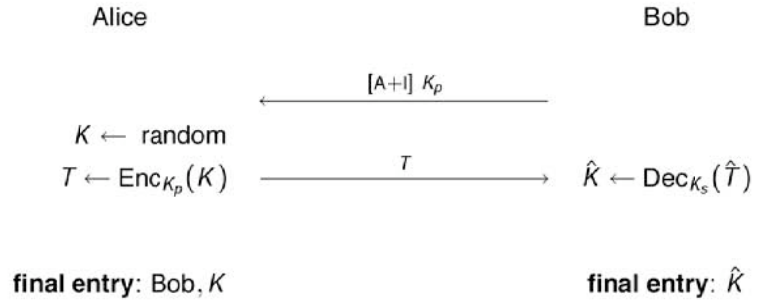
Secure Com. using a Key Agreement Protocol



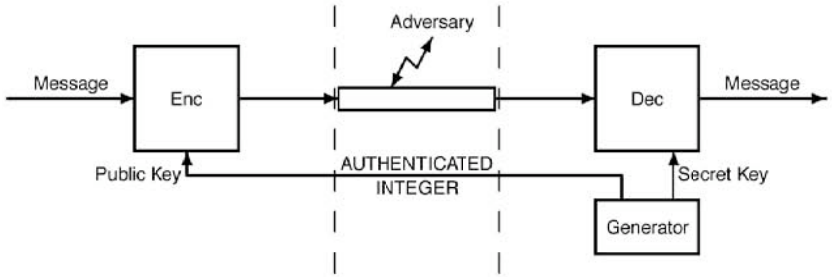
Diffie-Hellman 1976: Key Agreement Protocol



Semi-Authenticated Key Agreement Protocol



Public-Key Cryptosystems



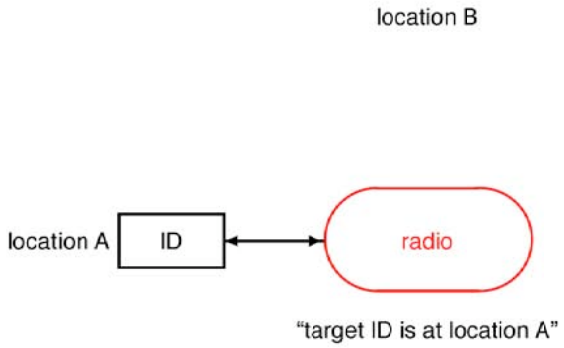
Session Integrity using A + I Message Security

Message freshness + liveliness + timeliness usually protected at the protocol level (except for the last message) by using message A + I

- acknowledge receipt of every message
- insert a sequence number in packets and check that received packets have consecutive sequence numbers
- insert an increasing nonce value (e.g. a clock value) + check for no packet loss by other means
- timeout

SSL or SSH: $Y = \mathrm{Enc/MAC}(\mathrm{seq}||X)$ where seq is implicit

Privacy



Privacy: a Multi-Level Security Issue

- anonymity: we do not know who (or to whom someone) is talking
- untraceability: after talking to someone, we cannot figure out that he is talking to someone else
- unlinkability: we cannot figure out that two talking devices are the same

→ actually, no stable definition at the moment

Security Properties and Toolkits

level	property	toolkit
message	authentication	MAC
	integrity	(comes with MAC for free)
	confidentiality	symmetric encryption
	freshness	solutions at the protocol level
	liveliness	solutions at the protocol level
	timeliness	solutions at the protocol level
protocol	key establishment	key agreement
	sequentiality	various protocol options
	termination	atomic commitment
all	privacy	?

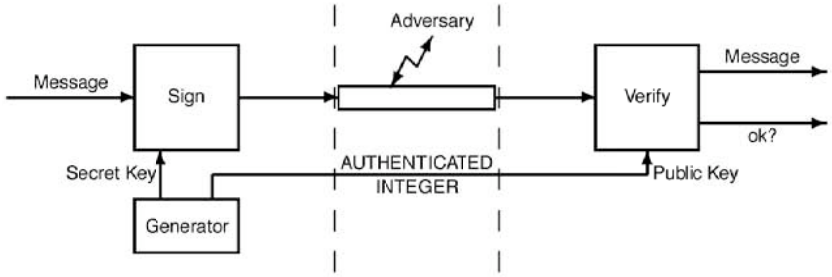
Conclusion on Communication Security

- two levels: message transmission level and protocol level
- at the message transmission lvel: symmetric cryptography
- at the protocol level: public-key cryptography

References

1. **Vaudenay**. *A Classical Introduction to Cryptography: Applications for Communication Security*. Springer. 2005.
2. **Stinson**. *Cryptography, Theory and Practice (2nd Edition)*. CRC. 2002.
   (lecture notes + exercise book available)
3. **Menezes-van Oorschot-Vanstone**. *Handbook of Applied Cryptography*. CRC. 1997.
   (reference book)
4. **Ferguson–Schneier**. *Practical Cryptography*. Wiley & Sons. 2003.
   (crypto for dummies)
5. **Levy**. *Crypto*. Penguin-Books. 2001.
   (history of modern cryptography)

Digital Signature



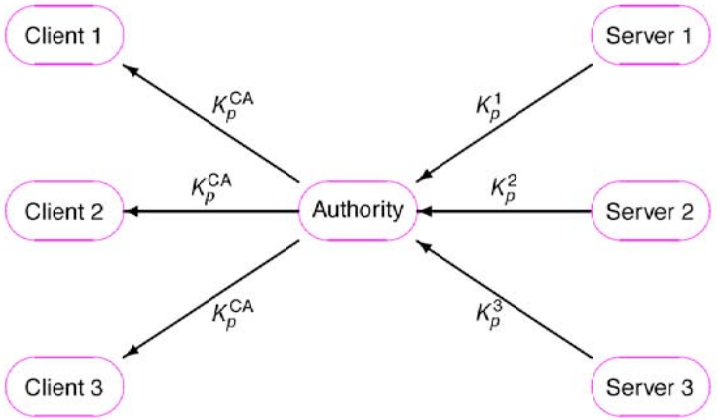
An X.509 Certificate Example: Overall Structure

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 674866 (0xa4c32)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=ZA, ST=Western Cape, L=Cape Town,
            O=Thawte Consulting cc, OU=Certification Services Division,
            CN=Thawte Server CA/Email=server-certs@thawte.com
        Validity
            Not Before: Jun  2 13:10:11 2003 GMT
            Not After : Jun 11 10:21:15 2005 GMT
        ...
        X509v3 extensions:
            X509v3 Extended Key Usage: TLS Web Server Authentication
            X509v3 Basic Constraints: critical CA:FALSE
    Signature Algorithm: md5WithRSAEncryption
        8d:7b:78:60:88:c4:13:4e:94:0d:bc:3b:1b:1c:b6:c9:bc:b1:
        0b:ed:7d:eb:6f:08:3a:ba:6d:21:36:93:38:36:66:7b:a7:bc:
        c0:3f:c4:e0:cf:b4:02:58:be:a6:b9:1d:45:a2:c4:58:38:07:
        e4:63:1a:d9:b9:8d:27:7c:93:67:31:82:6f:a3:3c:86:0c:e0:
        10:71:de:f2:e9:74:af:ac:76:b4:5b:8e:48:57:9d:8f:12:f6:
        72:63:8a:79:b4:74:e0:ba:ca:ac:1a:36:b4:16:38:c1:c5:d2:
        73:ed:e8:64:b0:ae:9e:e2:36:c7:0c:77:92:cc:c7:c0:e0:8a:
        54:24
```

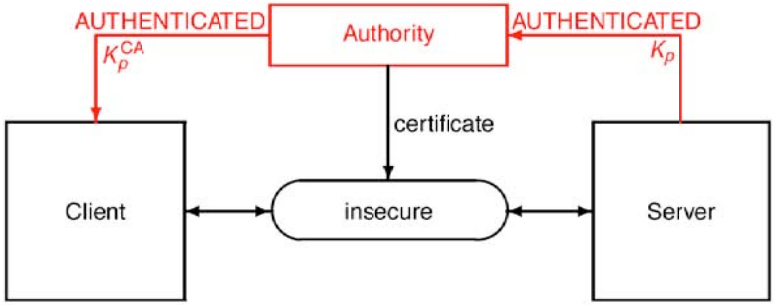

An X.509 Certificate Example: Subject

```
Subject: C=CH, ST=Bern, L=Bern,
    O=Switch - Teleinformatikdienste fuer Lehre und Forschung,
    CN=nic.switch.ch
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (1024 bit)
        Modulus (1024 bit):
            00:d0:0e:b7:16:bf:86:59:c3:97:e6:02:33:59:90:
            65:29:b0:69:73:64:83:03:1b:df:62:a8:4d:c0:4f:
            3c:d9:12:6b:8c:57:95:e1:57:e8:48:a6:7f:dd:15:
            8b:9d:ad:93:dc:78:af:06:1a:ce:0f:7b:cc:c4:6f:
            a0:06:26:40:73:04:d3:da:7b:20:c1:15:37:8c:2f:
            58:c4:d4:c1:4b:18:84:5c:54:f1:b1:a0:44:3c:e2:
            0e:8a:a2:63:48:6b:34:c7:10:9d:a1:23:56:77:f5:
            4e:3d:38:9a:70:5e:03:02:30:45:ee:81:e4:94:96:
            47:18:9e:47:37:bb:18:f6:87
        Exponent: 65537 (0x10001)
```


Architecture based on Certificate Authority



Certificate-Based Semi Authenticated Communication



History

- First version by Netscape in 1994
- Microsoft version PCT in 1995
- SSLv3 by Netscape in 1995
- IETF version TLS/1.0 in 1997 [RFC2246]

Goal: secure any communication (e.g. HTTP) based on TCP/IP

Common Use Principle

- client-server communications, random client, corporate server
- trusted third party: certificate authority (CA)
- A+I secure channel with CA to be used only once
- authentication of server based on public key
- authentication of client (if needed) based on password
- interoperable cipher suites

TLS Record Protocols

- Handshake Protocol (for initiating a session)
- Change Cipher Spec Protocol (for setting up cryptographic algorithms)
- Alert Protocol (for managing warnings and fatal errors)
- Application Data Protocol

Session State

- Session identifier
- Peer certificate (if any)
- Cipher suite choice
  - Algorithm for authentication and key exchange during handshake
  - Cipher Spec: symmetric algorithms (encryption and MAC)
- Master secret (a 48-byte symmetric key)
- nonces (from the client and the server)
- sequence numbers (one for each communication direction)
- compression algorithm (if any)

A session can include several connexions

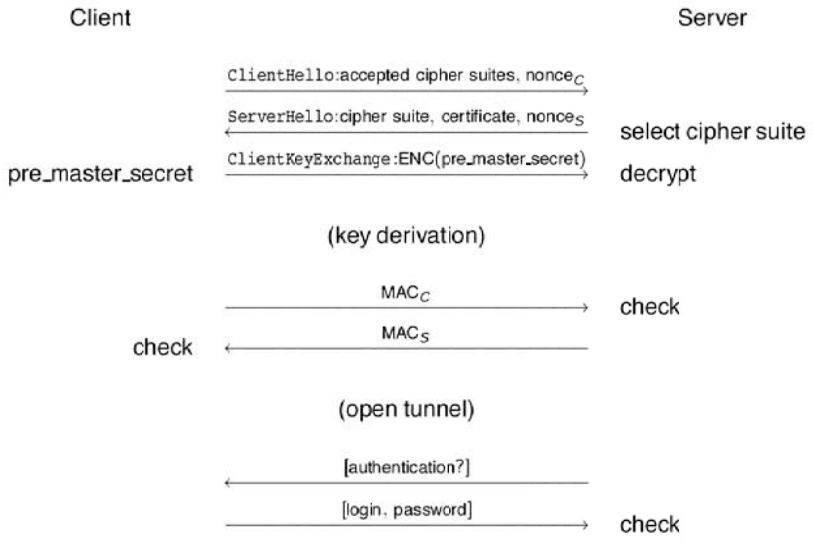
Original TLS Cipher Suites — i

CipherSuite	Key Exchange	Cipher	Hash
TLS_NULL_WITH_NULL_NULL	NULL	NULL	NULL
TLS_RSA_WITH_NULL_MD5	RSA	NULL	MD5
TLS_RSA_WITH_NULL_SHA	RSA	NULL	SHA-1
TLS_RSA_EXPORT_WITH_RC4_40_MD5	RSA	RC4_40	MD5
TLS_RSA_WITH_RC4_128_MD5	RSA	RC4_128	MD5
TLS_RSA_WITH_RC4_128_SHA	RSA	RC4_128	SHA-1
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5	RSA	RC2_40	MD5
TLS_RSA_WITH_IDEA_CBC_SHA	RSA	IDEA	SHA-1
TLS_RSA_EXPORT_WITH_DES40_CBC_SHA	RSA	DES40	SHA-1
TLS_RSA_WITH_DES_CBC_SHA	RSA	DES	SHA-1
TLS_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES_EDE	SHA-1
TLS_DH_DSS_EXPORT_WITH_DES40_CBC_SHA	DH_DSS	DES40	SHA-1
TLS_DH_DSS_WITH_DES_CBC_SHA	DH_DSS	DES	SHA-1
TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA	DH_DSS	3DES_EDE	SHA-1
TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA	DH_RSA	DES40	SHA-1
TLS_DH_RSA_WITH_DES_CBC_SHA	DH_RSA	DES	SHA-1
TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA	DH_RSA	3DES_EDE	SHA-1

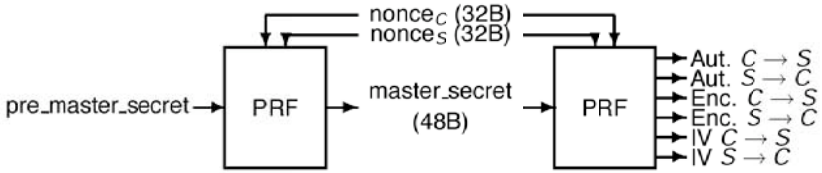
Original TLS Cipher Suites — ii

CipherSuite	Key Exchange	Cipher	Hash
TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA	DHE_DSS	DES40	SHA-1
TLS_DHE_DSS_WITH_DES_CBC_SHA	DHE_DSS	DES	SHA-1
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	DHE_DSS	3DES_EDE	SHA-1
TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA	DHE_RSA	DES40	SHA-1
TLS_DHE_RSA_WITH_DES_CBC_SHA	DHE_RSA	DES	SHA-1
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	DHE_RSA	3DES_EDE	SHA-1
TLS_DH_anon_EXPORT_WITH_RC4_40_MD5	DH_anon	RC4_40	MD5
TLS_DH_anon_WITH_RC4_128_MD5	DH_anon	RC4_128	MD5
TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA	DH_anon	DES40	SHA-1
TLS_DH_anon_WITH_DES_CBC_SHA	DH_anon	DES	SHA-1
TLS_DH_anon_WITH_3DES_EDE_CBC_SHA	DH_anon	3DES_EDE	SHA-1

A Typical TLS Session with RSA



Key Derivation



pre_master_secret is 48B for RSA key exchange or the obtained Diffie-Hellman key for DH_RSA, DH_DSS, DHE_RSA, DHE_DSS, and DH_anon

PRF

Given a secret, a seed, and a string label we define a sequence

$$
\begin{aligned}
a_0 &= \text{seed} \\
a_i &= \text{HMAC}_{\text{hash}}(S, a_{i-1}) \\
r_i &= \text{HMAC}_{\text{hash}}(S, a_i || \text{seed}) \\
\text{P\_hash}(S, \text{seed}) &= r_1, r_2, r_3, \ldots \\
\text{PRF}(\text{secret}, \text{label}, \text{seed}) &= \text{P\_MD5}(S1, \text{label} || \text{seed}) \oplus \\
&\quad\ \text{P\_SHA1}(S2, \text{label} || \text{seed})
\end{aligned}
$$

where $S1$ and $S2$ are the two halves of secret.
(If secret has an odd length, its middle byte is both the last byte of $S1$ and the first byte of $S2$.)

Using PRF

We define

$$
\begin{aligned}
\text{h\_handshake} &= \text{MD5(handshake)}||\text{SHA1(handshake))} \\
\text{MAC}_C &= \text{PRF(master\_secret,"client finished",h\_handshake)} \\
\text{MAC}_S &= \text{PRF(master\_secret,"server finished",h\_handshake)} \\
\text{master\_secret} &= \text{PRF(pre\_master\_secret,"master secret",nonce}_C||\text{nonce}_S) \\
\text{key\_block} &= \text{PRF(master\_secret,"key expansion",nonce}_S||\text{nonce}_C)
\end{aligned}
$$

handshake is the concatenation of all hanshake messages
MAC_C and MAC_S are of 12 bytes
key_block is the concatenation of the four private keys and the two initial vectors.

Record Protocol

- split the application data into fragments of at most 2^{14} Bytes and send the fragments separately.
- (optional) compress the fragment
- append a MAC to the fragment
  The MAC is computed on a sequence number, the compression and TLS version materials, the compressed fragment.
- encrypt all this
- send this after a record header (type, version, length)

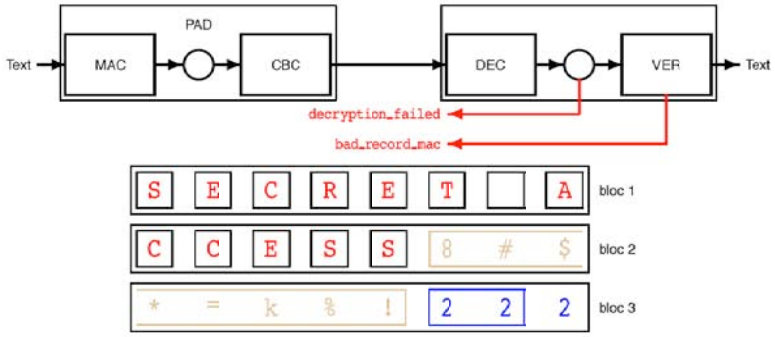
MAC in Record Protocol

More precisely the MAC of a fragment is computed as the HMAC with key `MAC_write_secret` on

```
seq_num
TLSCompressed.type,TLSCompressed.version,TLSCompressed.length
TLSCompressed.fragment
```

- `MAC_write_secret` is the MAC key of the sender
- `seq_num` is the sequence number of the fragment
- `TLSCompressed.fragment` is the compressed fragment
- `TLSCompressed.length` is its actual length
- `TLSCompressed.type`
- `TLSCompressed.version` are some information about the TLS protocol (namely, the compression algorithm) that is being used

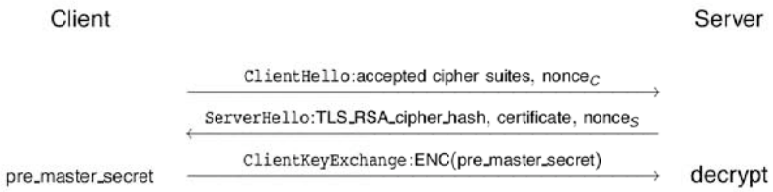
Using Block Ciphers in CBC Mode



Using Stream Ciphers

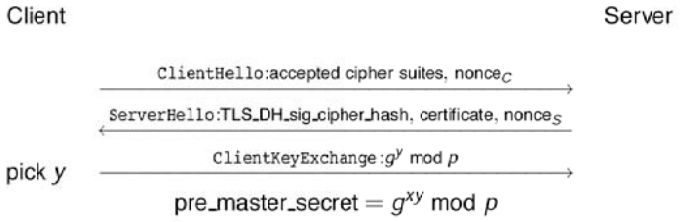
The RC4 stream cipher is used as a key-stream generator with one-time pad. The internal state of the generator is kept in the connection state so that the RC4 automaton continuously generates keystreams in order to encrypt the fragments sequence.

RSA Key Exchange



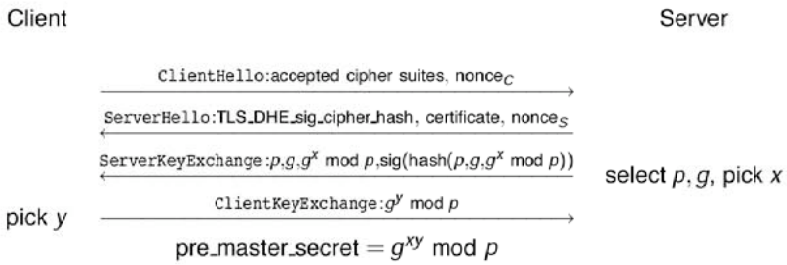
- RSA encryption is PKCS#1v1.5
- the RSA public key must be authenticated

DH_sig Key Exchange



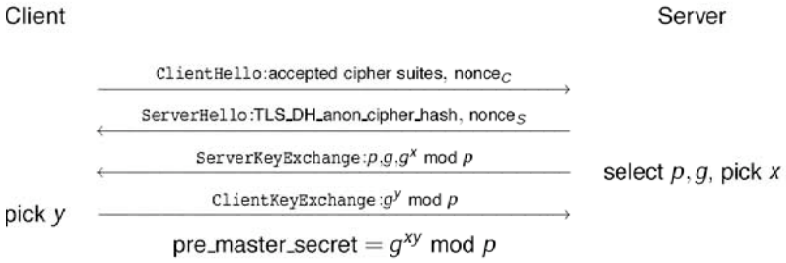
- the certificate is signed using sig algorithm
- the certificate includes $p, g, g^x \bmod p$
- this is fixed Diffie-Hellman where parameters are chosen by the server and server uses a fixed x

DHE_sig Key Exchange



- the sig public key must be authenticated in the certificate
- $g^y \bmod p$ is not authenticated!

DH_anon Key Exchange

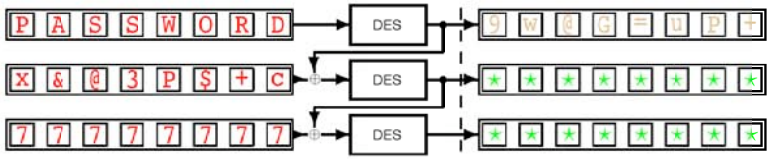


- Diffie-Hellman protocol is not authenticated!

TLS (In)security

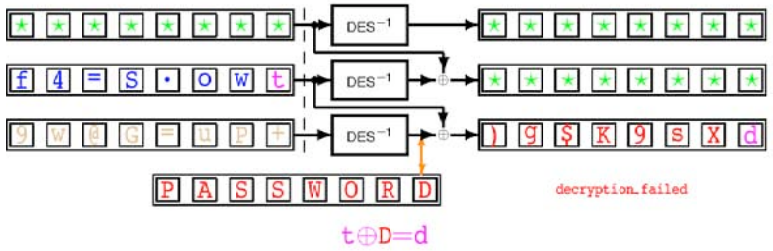
- confidentiality ☺/☹ (side channel)
- authentication ☺
- integrity ☺
- freshness ☺
- liveliness ☺ (except for the last message)
- key establishment ☺/☹ (side channel)
- sequentiality ☺ (except for messages loss)
- privacy ☺

CBCPAD Encryption

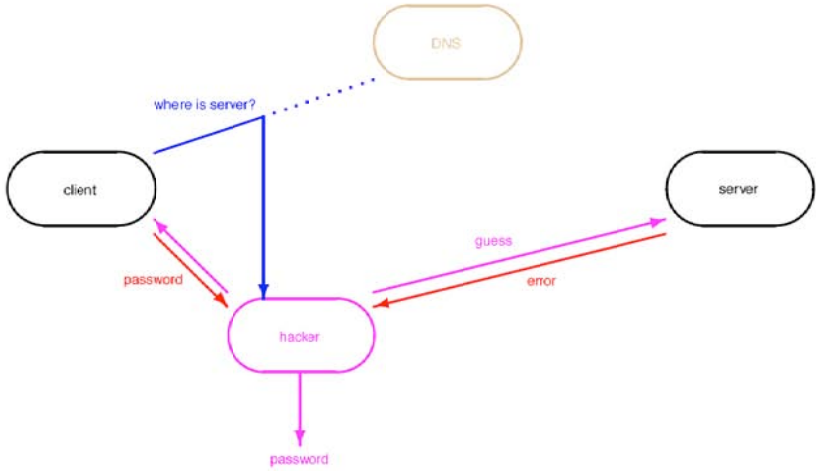


We would like to decrypt 9w@G=uP+

CBCPAD Decryption



The Attack Overview

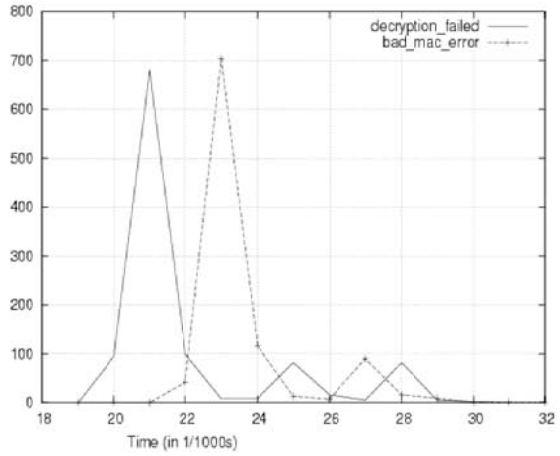


Application to TLS: Two Problems

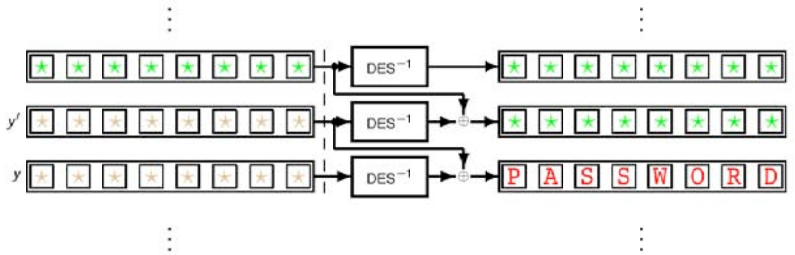
- Errors are sent encrypted through the Record Protocol
  $\longrightarrow$ errors are not available in clear
- Both types of errors are fatal alerts
  $\longrightarrow$ after one query the session is broken
  $\longrightarrow$ if the session restarts, it uses a fresh symmetric key

Using an Extra Side Channel

`bad_record_mac` errors take longer time to answer than `decryption_failed` errors



Multi-Session Attack

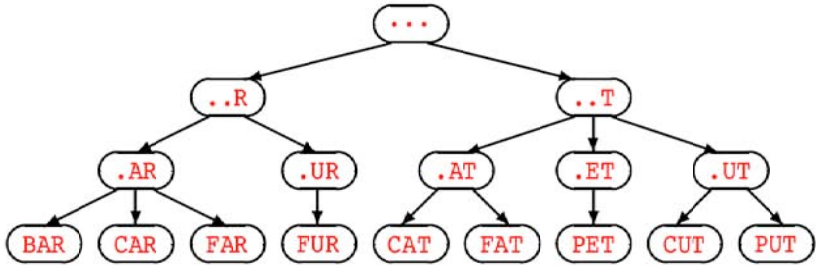


"does $\mathrm{DES}^{-1}(y)$ end with byte string u?"

$\Downarrow$

"does $\mathrm{DES}^{-1}(y) \oplus y'$ end with byte string u?"

Dictionary Attack



Application Model

We consider 4 senarii for blocks of $b = 8$ characters

- random characters in an alphabet of 256 letters (full byte)
- random characters in an alphabet of 64 letters (alphanumerical character)
- block in a dictionary of $D = 712'786$ words

Numerical Values

Uniform distribution, $|Z| = 256$

p	50%	60%	70%	80%	90%	99%
C	4239	4750	5353	6139	7397	11335

Uniform distribution, $|Z| = 64$

p	50%	60%	70%	80%	90%	99%
C	1140	1269	1421	1620	1938	2934

Dictionary

p	50%	60%	70%	80%	90%	99%
C	166	181	199	223	261	380

Password Interception

- IMAP client: Outlook Express 6.x from Microsoft under Windows XP
- IMAP Rev 4 server
- Outlook checks (by default) for messages automatically every 5 minutes each folder created on the IMAP user account
- E.g. five folders (in, out, trash, read, and draft)
  $\longrightarrow$ 60 sessions every hour
- Outlook sends the login and password to the IMAP server using the following format:
  `XXXX␣LOG|IN␣"user|name"␣"p|assword"|<0x0d><0x0a><HMAC1><HMAC2>...`

  Here `XXXX` are four random digits which are incremented each time Outlook connects to the server.

Conditions for a Successful Attack

- A critical piece of information is repeatedly encrypted at a predictable place.
- A block cipher in CBC mode is chosen.
- The attacker can sit in the middle and perform active attacks.
- The attacker can distinguish time differences between two types of errors.

Here we focused on the password access control in the IMAP protocol. We can also consider the basic authentication in HTTP which is also used for access control. This means that we can consider intercepting the password for accessing to an Intranet server.

Countermeasures

- The attack against WTLS was published in 2002.
- A countermeasure for TLS has been implemented in OpenSSL 0.9.6d and following versions:
  only the `bad_mac_error` error message is sent when an incorrect padding or an incorrect MAC are detected.
- This countermeasure is not enough because of timing attacks.
- A new countermeasure was implemented in OpenSSL 0.9.6i:
  we always check a MAC even if the padding is not correct.
- Other possible countermeasure:
  invert the padding and the MAC!

Lessons

- There are flaws, even in well esbablished standards
- We can make timing attacks over a network
- The order MAC-PAD-Encrypt should be reconsidered

Conclusion on TLS

- SSL puts together all cryptographic ingredients quite nicely
- it is permanently improved to fix mistakes and use the state-of-the-art cryptography

Further Readings

- **D. Bleichenbacher**.
  Chosen Ciphertext Attack Against Protocols Based on the RSA Encryption Standard PKCS#1.
  In *Advances in Cryptology (CRYPTO'98)*, LNCS vol. 1462, pp. 1–12, 1998.
- **B. Canvel, A. Hiltgen, S. Vaudenay, M. Vuagnoux**.
  Password Interception in a SSL/TLS Channel.
  In *Advances in Cryptology (CRYPTO'03)*, LNCS vol. 2729, pp. 583–599, 2003.

The Bluetooth Principles

- short-range wireless technology
- designed to transmit voice and data
- for a variety of mobile devices (computing, communicating, ...)
- bring together various markets

- 1Mbit/sec up to 10 meters over the 2.4-GHz radio fequency
- robustness, low complexity, low power, low cost

Bluetooth History

- 10th Century: Viking King Harald Blåtand (Harold Bluetooth) tried to unify Danemark, Norway, and Sweden
- 1994: Ericsson initiated a study to investigate the feasibility
- May 20, 1998: Bluetooth publicly announced, controled by the Special Interest Group (SIG) formed by

  Ericsson, IBM, Intel, Nokia, and Toshiba
- July 1999: Bluetooth 1.0 Specification Release
- November 2004: Bluetooth 2.0 Specification Release

Bluetooth Security

- mode 1: non-secure
- mode 2: service level enforced security
- mode 3: link level enforced security

Security from an Outside View

(for security mode 3)

- discoverable vs non-discoverable (privacy)

	non connectable	connectable
non discoverable	off	cruise mode
discoverable	—	setup mode

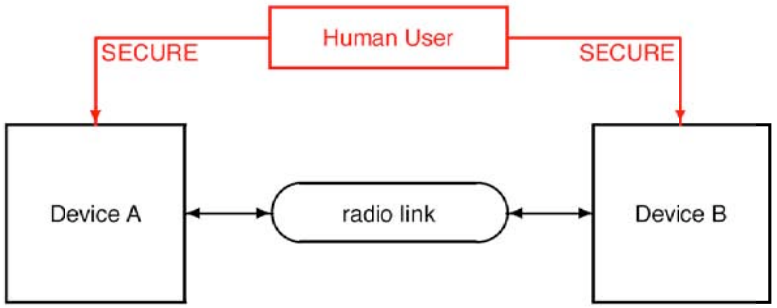
set mode ←

- pairing based on PIN code introduced by a human operator
  sometimes a fixed manufactures PIN code (sometimes null)

  pairing protocol ←
- database of paired devices

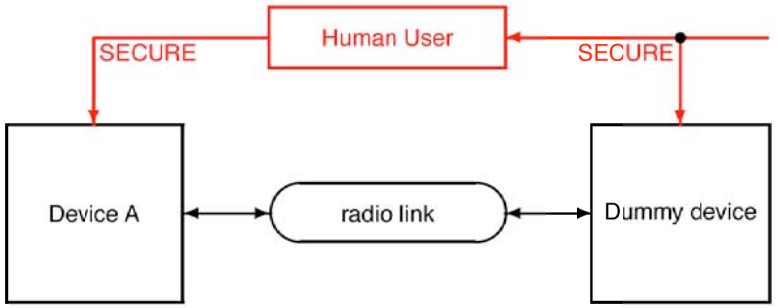
  list of paired devices ←

Bluetooth Channels



- secure channel for a PIN only
- security based on an ephemeral PIN

... with a Dummy Device



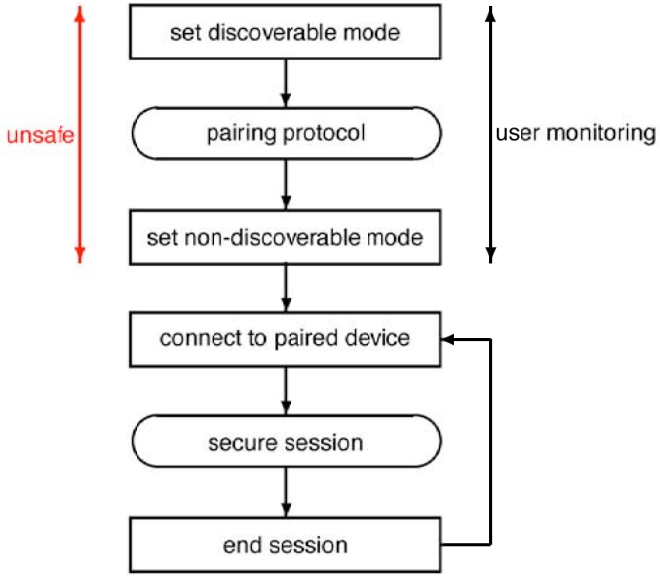
- limited keyboard and screen (button and LED only)
- manufactured PIN and semi-permanent unit key

Key Management from an Inside View

- pairing generates an ephemeral key K_{init}
- pairing leads to a long-term 128-bit link key K_{link}
- (dummy devices have a long-term key K_{unit} which can be forced to become the link key)

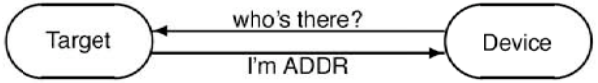
- link key used to authenticate devices and to derive an encryption key K_c
- effective length of K_c from 8 to 128 bits (for regulation purposes)

Privacy in Bluetooth

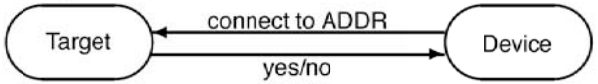


Discovery and Connection Protocols

- Discovery protocol:



- Connection protocol:



A Menagery of Primitives

- E0: encryption (stream cipher)
- E1: peer authentication (MAC)
- E21: key derivation (for K_{link})
- E22: key derivation (for K_{init})
- E3: key derivation (encryption key)

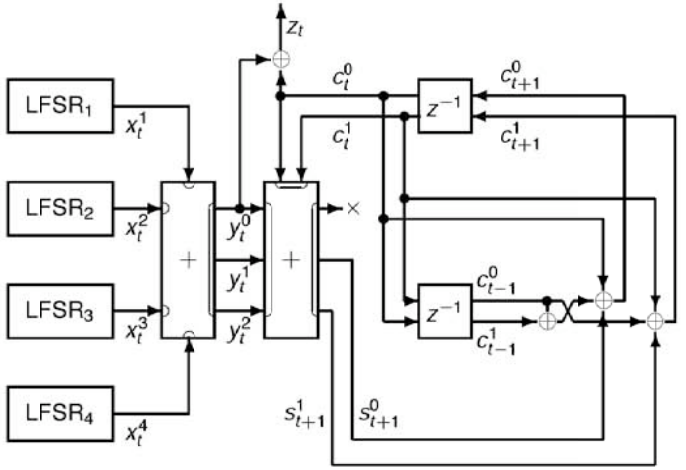
Bluetooth E0

- Designed by the Special Interest Group (SIG)
- Bluetooth standard
- default encryption scheme
- dedicated to lightweight hardware
- stream cipher with bit streams
- key of (up to) 128 bits, 26-bit clock, and 48-bit address

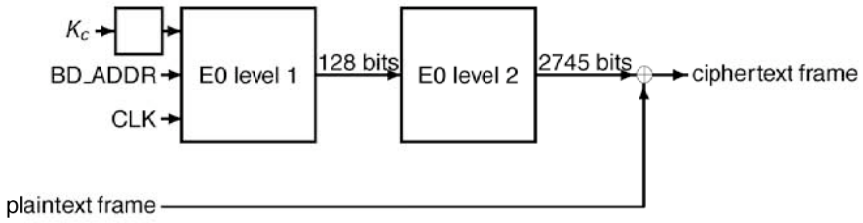
Why A Stream Cipher?

- low complexity overhead
- very advanced research on analysis
- state of the art grading (according to Adi Shamir 2003):
  - *Limited cryptanalytic tools*
  - *Narrow choice of primitives*
  - *Many insecure schemes*
  - *Challenge: Improve weak theory and weak practice*
  - *Final grade: 4*

Bluetooth E0 (1 Layer over 2)



E0 Key Schedule



- Frames are limited to 2745 bits
- Clock-based resynchronization using an additional E0 level

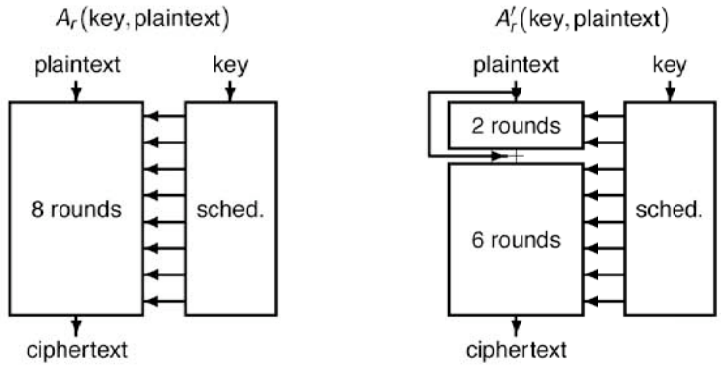
Academic Attacks on (Single-Level) E0

Attack	Precomputation	Time	Data	Memory
Initial state recovery given a long frame				
Armknecht-Krause 2003	-	2^{68}	2^{23}	2^{46}
Courtois 2003	2^{28}	2^{49}	2^{23}	2^{37}
Ekdahl 2003	2^{54}	2^{63}	2^{34}	2^{34}
Lu-Vaudenay 2004	2^{37}	2^{39}	2^{39}	2^{27}
Initial state recovery given many short frames				
Saarinen 2000	-	2^{93}	2^{7}	-
Krause 2002	-	2^{77}	2^{7}	-
Fluhrer-Lucks 2001	-	2^{51}	2^{8}	2^{51}
Lu-Vaudenay 2004	-	2^{16}	2^{16}	2^{16}

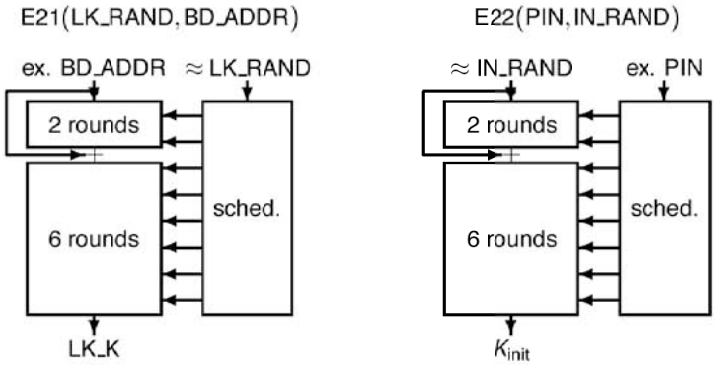
Key Recovery Attacks against Two-Level E0

Attack	Precomputation	Time	Data	Memory
exhaustive search	-	2^{128}	2^{8}	-
time-memory tradeoffs	2^{128}	2^{86}	2^{8}	2^{86}
dictionary attack	2^{128}	1	2^{8}	2^{128}
Saarinen 2000	-	2^{93}	2^{7}	-
Krause 2002	-	2^{113}	2^{7}	-
Fluhrer-Lucks 2001	-	2^{73}	2^{43}	2^{51}
Golić 2002	2^{80}	2^{70}	2^{17}	2^{80}
Lu-Vaudenay 2004	-	2^{45}	2^{42}	-
Lu-Meier-Vaudenay 2005	2^{38}	2^{38}	2^{28}	2^{33}

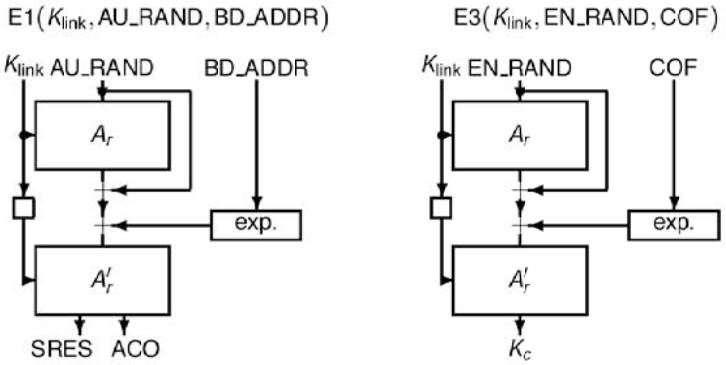
A Strange Hash Mode for SAFER+



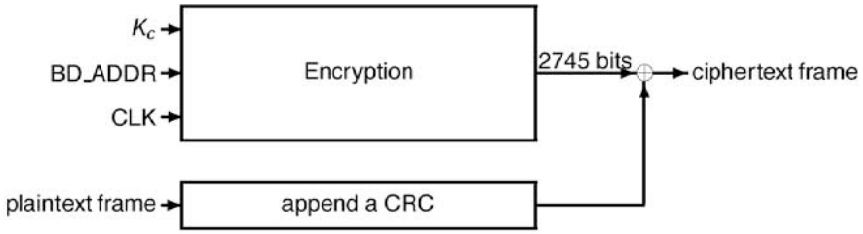
E21 and E22



E1 and E3



A Strange Authenticated Mode of Operation

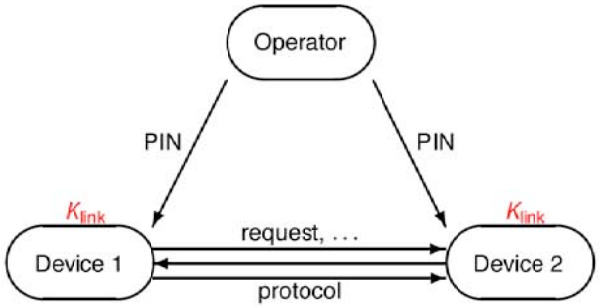


Integrity Insecurity

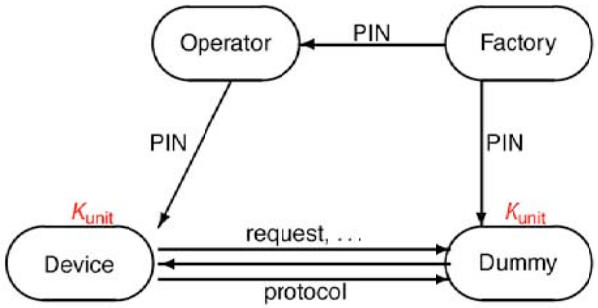
CRC pad is a rather weak integrity protection

→ packets are easily malleable (Borisov-Goldberg-Wagner 2001)

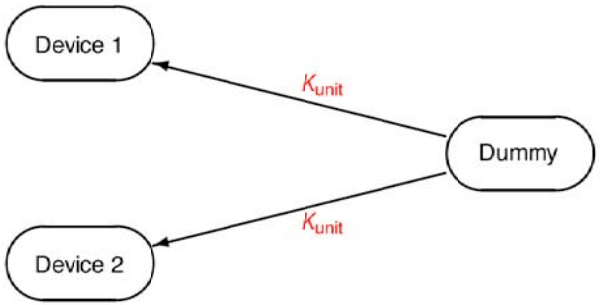
Device Pairing



Pairing with a Dummy Device



Unit Key is Shared with Many Devices



Peer Authentication

Master A		Slave B
pick challenge	challenge →	
check	response ←	compute response
	challenge ←	pick challenge
compute response	response →	check

$$\text{response} = \text{MAC}(\text{challenge})$$

Key Establishment (In)security

Theorem

Under some "reasonable assumptions", the pairing protocol is secure if either PIN *has large entropy or the protocol is run through a private channel.*

- ☺ a cheap pragmatic security
- ☹ pretty weak security

devastating sniffing attacks in other cases! (Jakobsson-Wetzel 2001)

Missing Security Protection

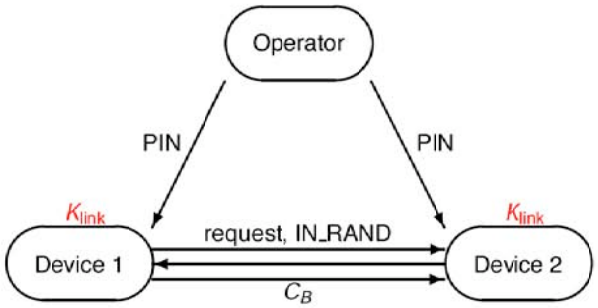
- Cryptographic pseudorandom generator
  → some device may have poor generators
- Liveliness + session sequentiality
  → some packets may be removed (Kügler 2003)
- Strong anonymity
  → traceability (Jakobsson-Wetzel 2001)
- termination fairness

Bluetooth (In)security

Current (mode 3) security is rather poor:

- confidentiality ☺ (attacks still academic so far)
- authentication ☺ (not academic though: by encryption)
- integrity ☹
- freshness ☺
- liveliness ☹
- key establishment ☹ (yes, but...)
- sequentiality ☺/☹ (message loss)
- privacy ☹

Device Pairing



Typical Bluetooth Pairing

Master A		Slave B
user inputs PIN code		
pick IN_RAND	$\xrightarrow{\text{IN\_RAND}}$	user inputs PIN code
$K_{init} = E22(PIN, IN\_RAND)$		$K_{init} = E22(PIN, IN\_RAND)$
pick LK_RAND_A		pick LK_RAND_B
$C_A = LK\_RAND_A \oplus K_{init}$		$C_B = LK\_RAND_B \oplus K_{init}$
	$\xrightarrow{C_A}$	
	$\xleftarrow{C_B}$	
$LK\_RAND_B = C_B \oplus K_{init}$		$LK\_RAND_A = C_A \oplus K_{init}$
compute K_{link}		compute K_{link}

$$K_{link} = E21(LK\_RAND_A, BD\_ADDR_A) \oplus E21(LK\_RAND_B, BD\_ADDR_B)$$

... with a Dummy Device

Master A		Slave B
user inputs PIN code		
pick IN_RAND	$\xrightarrow{\text{IN\_RAND}}$	user inputs PIN code (or not)
$K_{init} = E22(PIN, IN\_RAND)$		$K_{init} = E22(PIN, IN\_RAND)$
	$\xleftarrow{C_B}$	$C_B = K_{unit} \oplus K_{init}$
$K_{unit} = C_B \oplus K_{init}$		K_{unit}

link key is forced to be the unit key

→ problem if dummy device is paired with multiple devices

Peer Authentication

Master A		Slave B
pick AU_RAND_B	$\xrightarrow{AU\_RAND_B}$	
check $SRES_B$	$\xleftarrow{SRES_B}$	compute $SRES_B$
	$\xleftarrow{AU\_RAND_A}$	pick AU_RAND_A
compute $SRES_A$	$\xrightarrow{SRES_A}$	check $SRES_A$

$$SRES_d = E1(K_{link}, AU\_RAND_d, BD\_ADDR_d)$$

Insecurity (Jakobsson-Wetzel 2001)

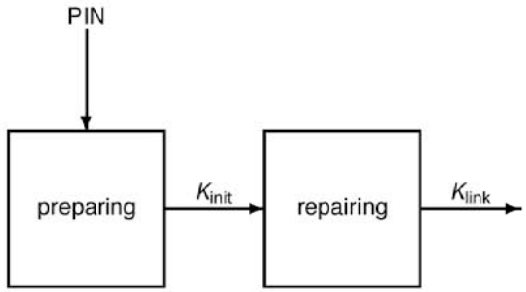
Assumption: pairing not made in a private environment (channel not confidential) and guessable PIN (lazy operator)

1. sniff the pairing protocol, get IN_RAND, C_A, C_B
2. $\longrightarrow$ can compute K_{link} from PIN
3. sniff a peer-authentication protocol, get rand, $F(\text{rand}, K_{\text{link}})$
4. $\longrightarrow$ can check a guess on K_{link}
5. run an offline exhaustive search on PIN

Possible Countermeasures

- do not use short PIN
  $\longrightarrow$ not realistic
- at least, do not use trivial PIN
  $\longrightarrow$ educational issue
- other countermeasure from the application layer?
  $\longrightarrow$ will this be effective?

Two Phases: Preparing and Repairing

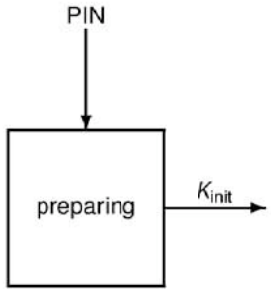


Pairing in Two Phases

Master A		Slave B
user inputs PIN code		
pick IN_RAND	$\xrightarrow{\text{IN\_RAND}}$	user inputs PIN code
$K_{\text{init}} = \text{E22}(\text{PIN}, \text{IN\_RAND})$		$K_{\text{init}} = \text{E22}(\text{PIN}, \text{IN\_RAND})$
pick LK_RAND_A		pick LK_RAND_B
$C_A = \text{LK\_RAND}_A \oplus K_{\text{init}}$		$C_B = \text{LK\_RAND}_B \oplus K_{\text{init}}$
	$\xrightarrow{C_A}$	
	$\xleftarrow{C_B}$	
$\text{LK\_RAND}_B = C_B \oplus K_{\text{init}}$		$\text{LK\_RAND}_A = C_A \oplus K_{\text{init}}$
compute K_{link}		compute K_{link}

preparing and repairing

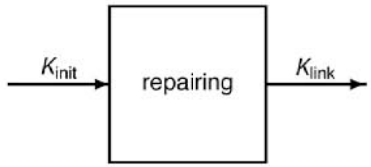
Preparing Phase



Goal

If either PIN *is confidential with large entropy or the protocol is private, then* K_{init} *is private.*

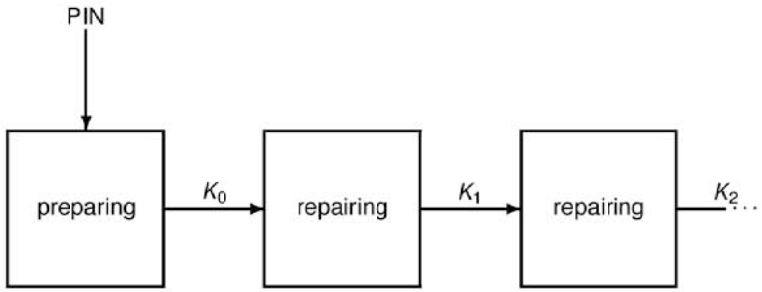
Repairing Phase



Goal

If either K_{init} *is private or the protocol is private, then* K_{link} *is private.*

A Possible Better Usage



Forward secrecy: if K_i is securely set up and if K_{i+t} is the first compromised key, all communications using $K_i, \ldots, K_{i+t-1}$ are safe

Advantage of Our Approach

- only solution without any public-key cryptography so far (efficient)
- only positive result on the Bluetooth pairing so far ☺
- pragmatic solution based on mobility

Conclusion on Bluetooth

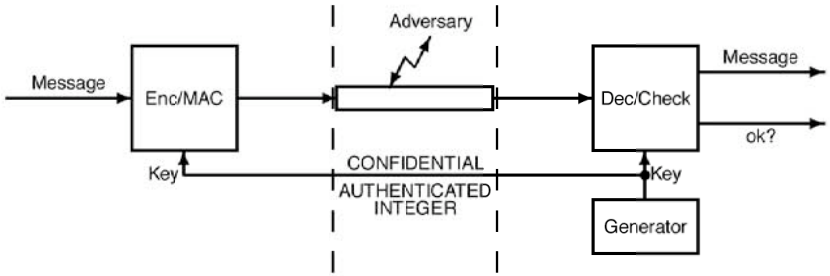
- Very efficient architecture (no public-key techniques)
- Default security is not very high
- Default security is not so low if properly used

Further Readings

- **M. Jakobsson**, **S. Wetzel**.
  Security Weaknesses in Bluetooth.
  In *Topics in Cryptology (CT–RSA'01)*, LNCS vol. 2020, pp. 176–191, 2001.
- **S. Vaudenay**.
  On Bluetooth Repairing: Key Agreement based on Symmetric-Key Cryptography.
  In *Information Security and Cryptology (CISC'05)*, LNCS vol. 3822, pp. 1–9, 2005.

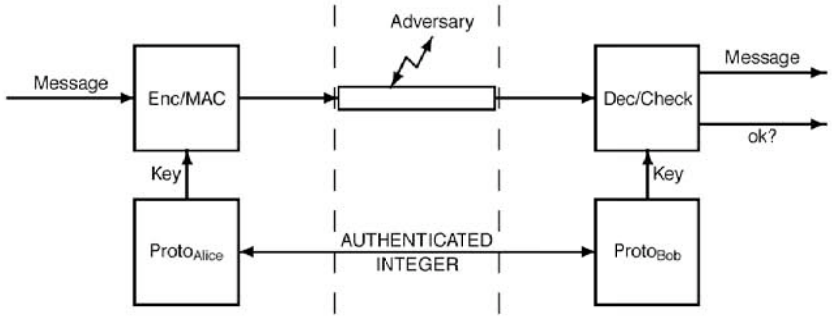
...based on C+A+I Channels

The Conventional Model



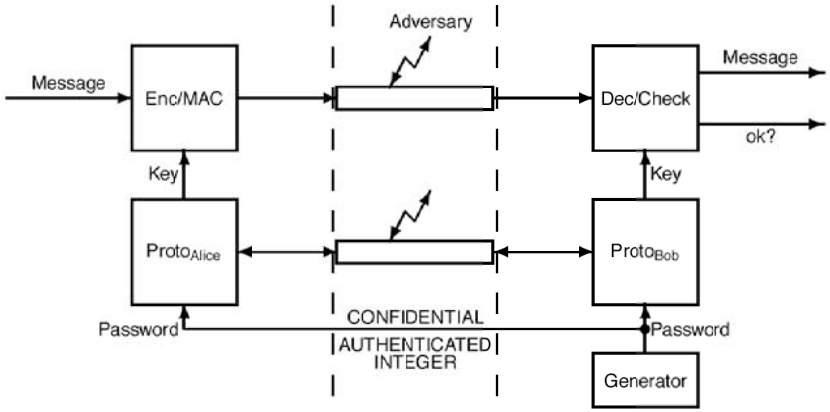
...based on A+I Channels

The Merkle Model 1975



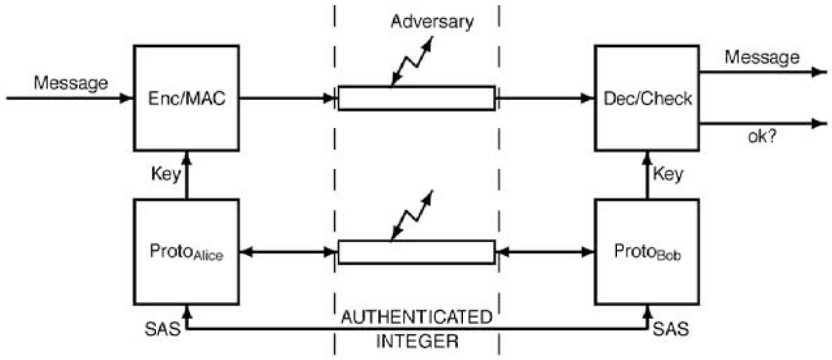
...based on C+A+I Narrowband Channels

The Bellovin-Merritt Model 1992

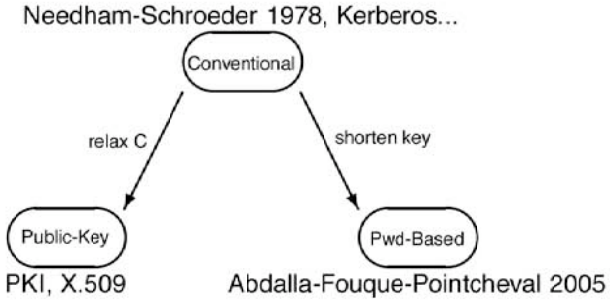


...based on A+I Narrowband Channels

The SAS-based Model 2005



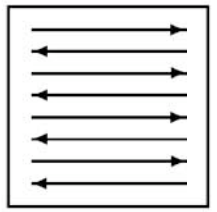
Solutions based on a Trusted Third Party



Pwd-Based Authenticated Key Agreement Protocols

Alice (ID_A)
password: (ID_B, w)

Bob (ID_B)
password: (ID_A, w)



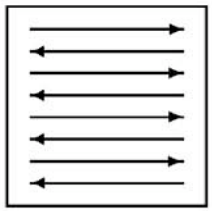
private output: K

private output: K

Pwd-Based Access Control Protocols

Alice (ID)
password: w

Bob
password: (ID, w)



output: ok

(Bad) Example: Password Access Control

Alice (ID)
password: w

Bob
password: (ID_A, w)

$\mathrm{ID}||w \longrightarrow$

check $\mathrm{ID}_A = \widehat{\mathrm{ID}}$
check $w = \hat{w}$

output: ok

w is disclosed

(Bad) Example: Challenge/Response Protocol

Alice (ID)
password: w

Bob
password: (ID_A, w)

$\xrightarrow{\mathrm{ID}}$ check $\mathrm{ID}_A = \widehat{\mathrm{ID}}$

$\xleftarrow{\mathrm{chall}}$ pick chall

$\mathrm{res} = \mathrm{MAC}_w(\mathrm{chall}) \xrightarrow{\mathrm{res}}$ check $\mathrm{res} = \mathrm{MAC}_w(\mathrm{chall})$

output: ok

subject to offline dictionary attacks

On Passwords with less than 48 Bits of Entropy

An 8 i.u.d. random characters password in $\{\mathtt{a}, \ldots, \mathtt{z}, \mathtt{A}, \ldots, \mathtt{Z}, 0, \ldots, 9\}$
$\longrightarrow$ entropy less than 48 bits

- classical conventional cryptography may require about 300 cycles on a P4 2GHz to check a guess ($= 2^{22.6}$ guesses per second)
  $\longrightarrow$ $256d$ to find a password with a PC
- time-memory tradeoffs cracked a (36-bit entropy) password within a few seconds (complexity $N^{\frac{2}{3}}$ + precomputation N)
  $\longrightarrow$ $1h$ to find a password (+ a year of precomputation)
- special purpose hardwares cracked 56-bit keys within a day
  $\longrightarrow$ $5\,min$ to find a password
- `distributed.net` cracked 64-bit keys within 1757 days in 2002
  $\longrightarrow$ $40\,min$ to find a password

Online Dictionary Attack: a Generic Attack

generic

1: **repeat**
2: make a new guess $\hat{w}$ following a dictionary
3: simulate Alice with password $\hat{w}$
4: launch an instance of the Bob protocol
5: run the complete protocol
6: **until** Bob accepts
7: print $\hat{w}$

How to Solve the Problem?

- make sure that systems refuse "too obvious" passwords and that passwords are regularly changed
  $\longrightarrow$ users do not remember strong passwords
  $\longrightarrow$ users divert it (write passwords on a post'it, ...)
- use tokens (securelD, challenge/response with strong password calculator, smart cards, one-time passwords, ...)
  $\longrightarrow$ a pain in the neck for users in most of cases
- find a pragmatic and technical solution
  $\longrightarrow$ leak no information which could be used to run offline attacks
  $\longrightarrow$ live with online dictionary attacks (slow down tests, audit, ...)

a protocol is secure if this attack is the best one

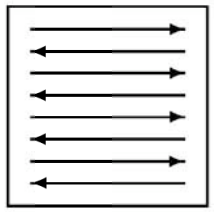
Existing Protocols

- Bellovin-Merritt 1992: EKE
  - general construction paradigm
  - can be based on ElGamal, Diffie-Hellman, RSA or other
  - informal (no security proof)
- Lucks 1997: OKE (later broken)
- Wu 1997: SRP (Secure Remote Password), quite popular
- EKE variants based on Diffie-Hellman
  - Bellare-Pointcheval-Rogaway 2000: EKE2
  - Boyko-MacKenzie-Patel 2000: PAK
  - Bellare-Rogaway 2000: AuthA (several variants)
  - Katz-Ostrovski-Yung 2001 (security proof without random oracles)
  - MacKenzie 2002: the PAK suite (PPK, PAK-X, PAK-Y, PAK-Z, ...)
  - Abdalla-Chevassut-Pointcheval 2005: another EKE+AuthA variant
  - others: SPEKE, augmented EKE, B-SPEKE, AMP, Jiang-Gong, ...
- protocols based on RSA
  - MacKenzie-Patel-Swaminathan 2000: SNAPI
  - Zhang 2004: PEKEP

Message Authentication Protocols

Alice (ID_A)
input: m

Bob



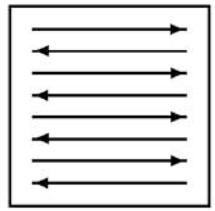
output: ID_A, m

Application: Semi-Authenticated Key Agreement

Alice (ID_A)

Bob

message: K_p



receive K_p
pick K

$\text{Enc}_{K_p}(K)$

decrypt ⟵ encrypt

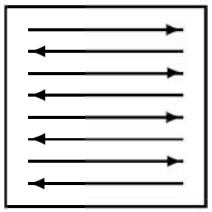
final entry: $(?, K)$

final entry: (ID_A, K)

Message Cross-Authentication Protocols

Alice (ID_A)
input: m_A

Bob (ID_B)
input: m_B



output: ID_B, m_B

output: ID_A, m_A

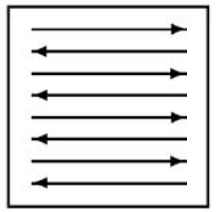
Application: Authenticated Key Agreement (e.g. DH)

Alice (ID_A)

Bob (ID_B)

pick x_A
message: $y_A = g^{x_A}$

pick x_B
message: $y_B = g^{x_B}$



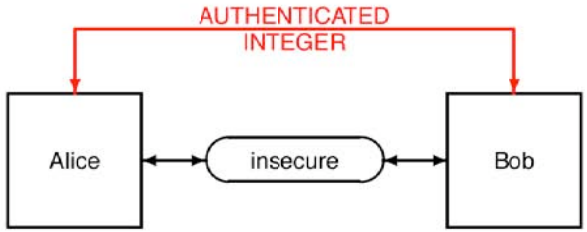
receive y_B
$K \leftarrow y_B^{x_A}$

receive y_A
$K \leftarrow y_A^{x_B}$

final entry: (ID_B, K)

final entry: (ID_A, K)

Communication Model



- secure channel (A+I) with low bandwidth

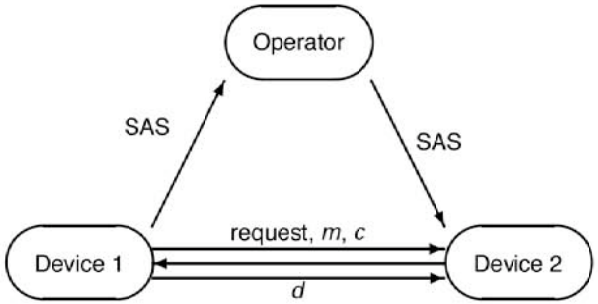
Communication Model: Adversary Capabilities

Regular channels: the adverary can do whatever he/she wants with the messages: modify, create, swap, remove, stall, ...

(Weak) authenticated channels: the adversary cannot modify nor create messages. He/she can swap, remove, stall, ...

(Strong) authenticated channels: same plus some additional assumptions!
E.g. messages must be either deliver at once or removed (stall-free channels).

Application I: Personal Area Network Setup



Application II: Peer-to-Peer PGP Channel Setup

Application III: Disaster Recovery

- on the road, after a key loss (computer crash, stolen laptop)
  $\longrightarrow$ set up of a security association
- PKI collapse (company bankrupt, main key sold, act of God)
  $\longrightarrow$ set up of a security association

Folklore (Balfanz-Smetters-Stewart-Chi Wong 2002)

Alice		Bob
input: m		
	$\xrightarrow{m}$	
$h \leftarrow H(m)$	$\xrightarrow{\text{authenticate}_{\text{Alice}}(h)}$	check $h = H(\hat{m})$
		output: Alice, $\hat{m}$

Example

Faculté informatique et communications
Institut de systèmes de communication
Laboratoire de sécurité et de cryptographie

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Serge VAUDENAY
Professeur

EPFL IC ISC LASEC
INF 241
Station 14
CH - 1015 Lausanne

Tél.: +41 21 6937696
Fax: +41 21 6936870
serge.vaudenay@epfl.ch
http//:lasecwww.epfl.ch

12E7 CAE2 2119 086C DC3D 8EAB 2EA5 9621 5E8C 7956

Security

Theorem

If H is a collision resistant hash function onto $\{0,1\}^k$, the protocol resists to impersonation attempts.

- ☺ provable security, efficient (assuming collision resistance)
- ☹ this requires SAS of at least 160 bits

Gehrmann-Mitchel-Nyberg 2004: The MANA I Protocol

Alice		Bob
input: m		
	$\xrightarrow{m}$	
pick $K \in_U \{0,1\}^k$		
$\mu \leftarrow H_K(m)$	$\xrightarrow{\text{authenticate}_{\text{Alice}}(K\|\|\mu)}$	check $\mu = H_K(\hat{m})$
		output: Alice, $\hat{m}$

Insecurity of MANA I

Alice		Bob
input: m		
pick $K \in_U \{0,1\}^k$	$\xrightarrow{m} \cdots$	
$\mu \leftarrow H_K(m)$	$\xrightarrow{\text{authenticate}_{\text{Alice}}(K\|\|\mu)} \cdots$	
	find $\hat{m}$ s.t. $H_K(m) = H_K(\hat{m})$	
	$\cdots \xrightarrow{\hat{m}}$	
	$\cdots \xrightarrow{\text{authenticate}_{\text{Alice}}(K\|\|\mu)}$	check $\mu = H_K(\hat{m})$
		output: Alice, $\hat{m}$

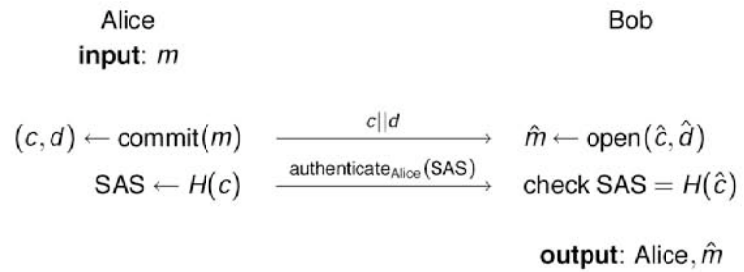
Security of MANA I

Theorem

Using a universal hash function family H which produces ℓ-bit codes and in a **strong communication model**, *the maximal probability of success of an impersonation of Alice when limited to Q_A runs of Alice's protocol and Q_B runs of Bob's protocol is at most $Q_A Q_B 2^{-k-\ell}$.*

- ☺ can work with SAS of $k + \ell = 20$ bits
- ☹ strong requirement on the communication model

Pasini-Vaudenay 2006: SAS-Based NIMAP



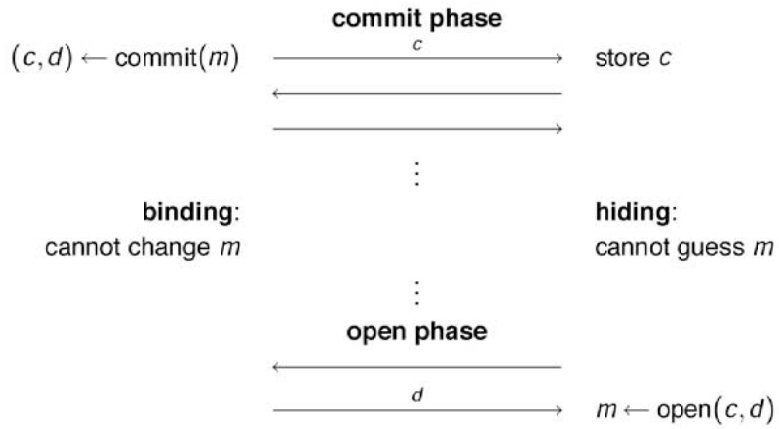
Security

Theorem

Under "reasonable assumptions" on the commitment scheme and if the hash function resists second preimage attacks, the maximal probability of success of an impersonation of Alice when limited to Q_A runs of Alice's protocol is at most $Q_A\varepsilon$.

- ☺ provable security, efficient
- ☺ can work with SAS of 80 bits (the least possible for NIMAP)

Commitment Schemes



Example (Based on ROM)

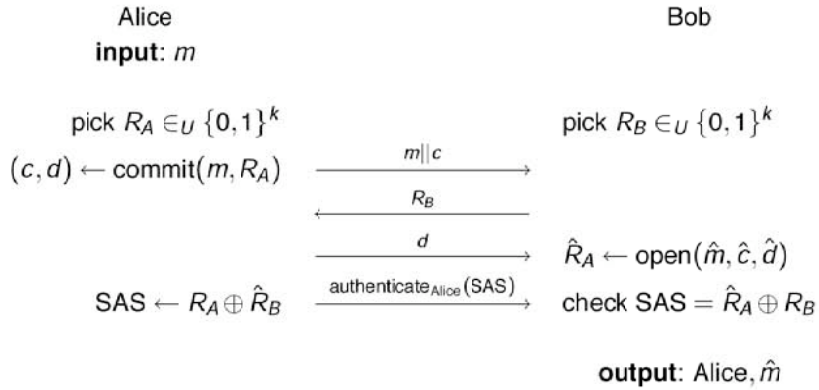
Commit: to commit on m:

1. pick a random e, set $d = m||e$
2. send d to a random oracle H
3. get c

Decommit: check that $H(d) = c$, parse $d = m||e$ and output m

$\longrightarrow$ Instanciation: take $H =$ SHA1 and hope it makes sense...

Vaudenay 2005: SAS-Based Authentication



Security

Theorem

Under "reasonable assumptions" on the commitment scheme, the maximal probability of success of an impersonation of Alice when limited to Q_A runs of Alice's protocol and Q_B runs of Bob's protocol is at most $Q_A Q_B 2^{-k} + \varepsilon$.

- ☺ provable security, efficient
- ☺ can work with SAS of 20 bits

Example (Based on ROM)

Commit: to commit on r with tag m:

1. pick a random e, set $d = r||e$
2. send $m||d$ to a random oracle H
3. get c

Decommit: check that $H(m||d) = c$, parse $d = r||e$ and output r

$\longrightarrow$ Instanciation: take $H = \text{SHA1}$ and hope it makes sense...

Hoepman 2004: Authenticated Key Agreement Protocol

Alice		Bob
pick x_A, $y_A \leftarrow g^{x_A}$		pick x_B, $y_B \leftarrow g^{x_B}$
	$\xrightarrow{\text{commit}(y_A)}$	
	$\xleftarrow{\text{commit}(y_B)}$	
$\text{SAS}_A \leftarrow \text{pieceof}(y_A)$	$\xrightarrow{\text{authenticate}_{\text{Alice}}(\text{SAS}_A)}$	
	$\xleftarrow{\text{authenticate}_{\text{Bob}}(\text{SAS}_B)}$	$\text{SAS}_B \leftarrow \text{pieceof}(y_B)$
	$\xrightarrow{\text{open}(y_A)}$	check commit, SAS
check commit, SAS	$\xleftarrow{\text{open}(y_B)}$	
$z_A \leftarrow (\hat{y}_B)^{x_A}$		$z_B \leftarrow (\hat{y}_A)^{x_B}$
final entry: Bob, z_A		**final entry**: Alice, z_B

Properties

- ☺ can work with SAS of 20 bits
- ☹ 4-move + SAS exchange
- ☹ two different SAS
- ☹ security unclear at this time, hard to instanciate

Zimmermann 1995: PGPfone

Alice		Bob
pick x_A, $y_A \leftarrow g^{x_A}$		pick x_B, $y_B \leftarrow g^{x_B}$
	$\xrightarrow{\text{commit to } (y_A)}$	
	$\xleftarrow{y_B}$	
$z_A \leftarrow \hat{y}_B^{x_A}$	$\xrightarrow{\text{open commitment}}$	$z_B \leftarrow \hat{y}_A^{x_B}$
$\text{SAS} \leftarrow \text{truncH}(y_A \Vert \hat{y}_B)$	$\xrightarrow{\text{authenticate}_{\text{Alice}}(\text{SAS})}$	$\text{SAS} \stackrel{?}{=} \text{truncH}(\hat{y}_A \Vert y_B)$
check SAS is the same	$\xleftarrow{\text{authenticate}_{\text{Bob}}(\text{SAS})}$	
final entry: Bob, z_A		**final entry**: Alice, z_B

Properties

- ☺ can work with SAS of 20 bits
- ☺ 3-move + SAS exchange
- ☺ a single (2-way) SAS
- ☹ security might be tricky to prove (?)

A 3-move + 2-way SAS Cross-Authentication Protocol

Alice **input**: m_A		Bob **input**: m_B
pick $K \in_U \{0,1\}^\kappa$		pick $R \in_U \{0,1\}^\rho$
$(c,d) \leftarrow \text{commit}(m_A, K)$	$\xrightarrow{m_A, c}$	
	$\xleftarrow{m_B, R}$	
	$\xrightarrow{d}$	$\hat{K} \leftarrow \text{open}(\hat{m}_A, \hat{c}, \hat{d})$
$\text{SAS} \leftarrow \hat{R} \oplus h_K(\hat{m}_B)$	$\xrightarrow{\text{authenticate}_{\text{Alice}}(\text{SAS})}$	$\text{SAS} \stackrel{?}{=} R \oplus h_{\hat{K}}(m_B)$
check SAS is the same check Alice $\neq$ Bob	$\xleftarrow{\text{authenticate}_{\text{Bob}}(\text{SAS})}$	
final entry: Bob, $\hat{m}_B$		**final entry**: Alice, $\hat{m}_A$

Security

Theorem

With a commitment scheme based on the random oracle model and a universal hash function family, the maximal probability of success of an impersonation of Alice when limited to Q runs of Alice/Bob is at most $Q(2^{-\rho} + \varepsilon_h + \varepsilon_c)$.

- ☺ can work with SAS of 20 bits
- ☺ 3-move + SAS exchange
- ☺ a single (2-way) SAS
- ☹ using a random oracle

Random Oracle Commitment

(same as the extractable commitment based on ROM)

Commit: to commit on k with tag m:

1. pick a random e, set $d = e||k$
2. send $d||m$ to a random oracle H
3. get c

Decommit: check that $H(d||m) = c$, parse $d = e||k$ and output k

Epsilon: $\varepsilon_c = q^2(2^{-\text{length}(e)} + 2^{-\text{length}(c)})$

$\longrightarrow$ Instanciation: take $H = \text{SHA1}$ and hope it makes sense...

Universal Hash Function Family

Definition

A universal hash function family is a family of mappings from $\{0,1\}^*$ to $\{0,1\}^\rho$ indexed by a κ-bit key K such that

$$\forall a,b,\alpha,\beta \quad (a \neq b) \quad \Pr_K[h_K(a)=\alpha, h_K(b)=\beta] \leq 2^{-2\rho} + 2^{-\rho}\varepsilon_h$$

Heuristic example: $h_K(x) = \text{trunc}_\rho(\text{SHA1}(K||x))$

Fact

$$\forall a,\alpha \qquad \Pr_K[h_K(a)=\alpha] \leq 2^{-\rho} + \varepsilon_h$$
$$\forall a,b,\delta \quad (a \neq b) \quad \Pr_K[h_K(a) \oplus h_K(b) = \delta] \leq 2^{-\rho} + \varepsilon_h$$

Conclusion

- secure communications over insecure channels *can* be manually set up by a human operator
- public-key -less solutions: although pretty weak, Bluetooth standards *can* offer a pragmatic costless security when properly used
- applications: personal area network, peer-to-peer, disaster rescue

References

1. **D. Balfanz, D. K. Smetters, P. Stewart, H. Chi Wong.**
   Talking to Strangers: Authentication in Ad-Hoc Wireless Networks.
   In *Network and Distributed System Security Symposium Conference (NDSS 02)*, 2002.
2. **C. Gehrmann, C. Mitchell, K. Nyberg.**
   Manual Authentication for Wireless Devices.
   In *RSA Cryptobytes*, vol. 7, pp. 29–37, 2004.
3. **S. Vaudenay.**
   Secure Communications over Insecure Channels Based on Short Authenticated Strings.
   In *Advances in Cryptology (CRYPTO'05)*, LNCS vol. 3621, pp. 309–326, 2005.
4. **S. Pasini, S. Vaudenay.**
   Secure Communications over Insecure Channels Using an Authenticated Channel.
   In *Topics in Cryptology (CT-RSA'06)*, LNCS vol. 3860, pp. 280–294, 2006.
5. **S. Pasini, S. Vaudenay.**
   SAS-Based Authenticated Key Agreement.
   To appear in PKC'06, LNCS, 2006.

Déni de service et sécurité des réseaux de télécommunication

Yvo Desmedt

Professeur à l'University College London
Grande-Bretagne

Déni de service et sécurité des réseaux de télécommunication

Yvo Desmedt
BT Chair of
Information Security
University College London
Grande-Bretagne

École de Printemps
Cryptographie et sécurité informatique
Vendredi 28 avril, 2006

Yvo Desmedt was also partially supported by NSF CCR-0209092, EPSRC EP/C538285/1 and DARPA F30602-97-1-0205. He is also a courtesy professor at Florida State University (USA).

OVERVIEW

Important note: t and k are used as synonyms.

1. INTRODUCTION

September 11, 2001 the US was attacked by a new type of attack: the use of commercial airplanes to attack buildings.

Lessons:

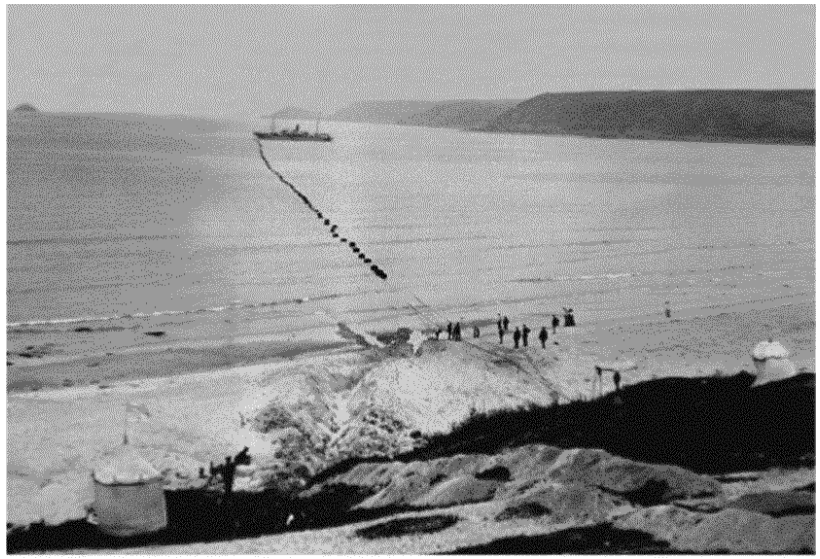
- It demonstrates that if one wants to achieve true information security, one should not only consider what hackers do today, but what they may do.
- we should consider how dependent we are on certain technologies.

Today's denial of service, are primarily:

- Computer viruses, worms and hybrids
- Ping, syn attacks and variants

Today's denial of service attacks usually target general purpose computers, such as mainframes, laptops, etc. Attack on routers are relatively rare, a technology on which the internet depends heavily.

This situation has not always been the case. During wars, switching centers have been bombed. Moreover, underminining communication is an important strategy. See for example:



Accordingly to:
http://www.catalogue.nationalarchives.gov.uk/Leaflets/ri2026.htm

> On 5 August 1914, the cable ship Telconia lifted from the bed of the North Sea the German overseas telegraph cables. Thereafter German diplomatic communications had to go by wireless, as did signals to the High Seas Fleet and the U boats. These could be intercepted and so were sent by cypher. Cryptography had been subject to a lot of study in Britain before the War, particularly at Naval Intelligence Department, and as a result, specialists at NID were able to read many of Germany's diplomatic and operational signals.

Most researchers are worried about eavesdropping, but ignore denial of service!

Hardware type attacks may be done by previously mentioned methods. Moreover, for many routers one knows their GPS coordinates!

What is the potential of software type attacks?

At the BlackHat 2005 conference it was shown that potential attacks against routers are far from hypothetical.

Current TCP/IP protocols (even including IPSEC) do not have the resilience to deal with routers taken over by the adversary.

2. WHAT DO ROUTERS DO?

We only discuss this at a high level.

They primarily:

- construct an edge-labeled graph (label: e.g. label: delay) of the network, and
- route communication packages.

Traditional algorithms to construct the graph do not assume the existence of untrusted routers. They can deal with routers that are down.

3. DENIAL OF SERVICE DURING COMMUNICATION: THE ISSUES

There are several issues, depending on:

The type of network. We distinguish:

Point-to-point networks

(Partial) broadcast

The type of adversary. We have:

Passive adversary has control to a subset of nodes. The adversary has access to all information received by these nodes (and all secrets of these nodes).

Active adversary. The nodes over which the adversary has control

can behave in a Byzantine way. This means they can decide, not to forward information, modify, not follow the protocol, follow the protocol, etc.

Destroyed nodes, i.e. just stops communicating.

Jamming adversary in the case of (partial) broadcast, a third party can prevent communication between two parties.

The nodes controlled by the adversary. These can be specied by

a threshold. A t-bounded adversary can control up to t nodes.

an adversary structure. Let V be the nodes in the network. An adversary structure $\mathcal{A}_V$ over V is a subset of the power set 2^V such that if $B \in \mathcal{A}_V$ then subsets of B are also in $\mathcal{A}_V$.

The level of security. We distinguish between:

Perfect (see further).

δ**-reliability,** i.e. with probability at least $1 - \delta$, B terminates with the same message as A sent.

ε**-privacy.** Unconditional security (See literature: Franklin-Wright.) Edge: considered private communication.

The perfect case corresponds to $\delta = 0$ and $\varepsilon = 0$.

Decisional versus search. We distinguish between:

Decisional questions: given a network, does it allow the desired security against a type of adversary? So, issues are:

- necessary and sufficient conditions
- if possible, what protocol do the participants run
- an algorithm for deciding, or
- proving the problem is hard (e.g. **NP**-hard).

Computational questions , in particular:

Construct a network for the desired security against a type of adversary with parameters, e.g. number of nodes is given. Issues:

Bounds: Necessary conditions

Constructions: Sufficient conditions

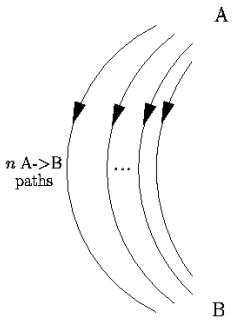
Update a network. Start from existing network (satisfying a security property). How to update it (with minimum “cost”) so it satisfies a (new) security property?

4. POINT-TO-POINT: A t-BOUNDED ADVERSARY: INTRODUCTION

If an adversary can destroy t nodes, then $t+1$ vertex disjoint (directed) paths are needed and sufficient to communicate from node A to node B. If any two non-destroyed nodes want to communicate, it is necessary and sufficient that the directed graph must be strongly $t+1$ connected.

If the adversary can be Byzantine, then one needs $2t+1$ vertex disjoint (directed) paths, respectively $2t+1$ strong connectivity.

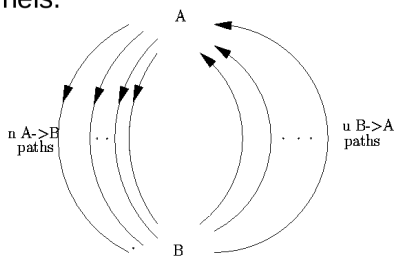
Dolev-Dwork-Waarts-Yung (1993) added privacy and studied the cases all communication links (edges in the graph) are:



one-way without feedback. It is necessary and sufficient to have $3t+1$ vertex disjoint directed paths from A to B (for any two nodes: the graph must be $3t+1$ connected).

two-way. $2t+1$ vertex disjoint paths are necessary and sufficient.

Desmedt and Wang (2002) observed this is not the most general case, since there could be feedback channels. They focussed primarily on the case the feedback channels are vertex disjoint from the forward channels.



5. POINT-TO-POINT: A t-BOUNDED ADVERSARY: SMALL ERROR

The receiver may accept an incorrect message with δ probability (any $0 < \delta < 1/2$).

No feedback:

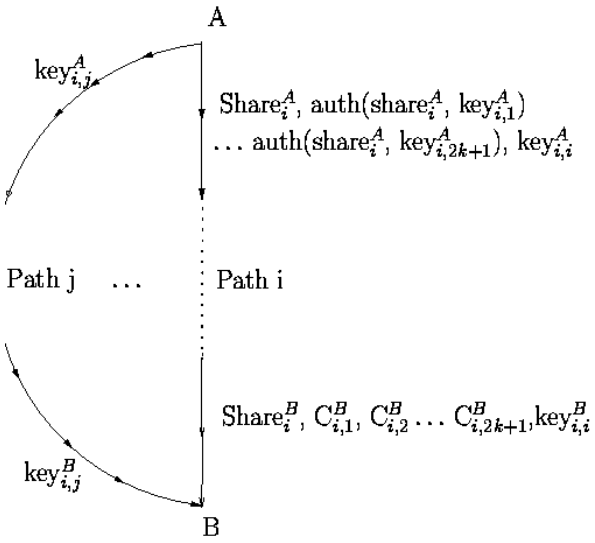
Call sender A and receiver B.

required: $2k+1$ vertex disjoint paths (Franklin-Wright).

Claim: $2k+1$ is sufficient.

Protocol A makes shares from the secret using a $k+1$-out-of-$2k+1$ perfect secret sharing scheme. Then:

For each i ($1 \leq i \leq 2k+1$), for each j:



If $|\{C^B_{i,j} : C^B_{i,j} = \text{auth}(\text{Share}^B_i, \text{key}^B_{i,j})\}| \geq k+1$, then B accepts Share^B_i. Then from accepted shares B reconstructs the secret.

Theorem 1. *Gives perfect privacy, δ reliability, by choosing proper authentication code.*

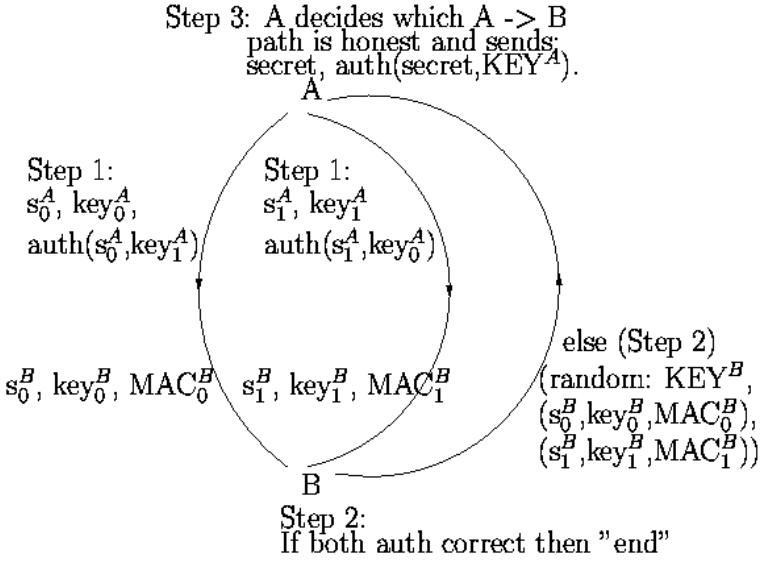
Feedback: Assume u disjoint feedback channels (i.e. from B to A).

Corollary 1. *(Follows from Franklin-Wright) Necessary conditions:*

- *# vertex-disjoint $A \to B$ paths $\geq k+1$.*
- *#disjoint $A \to B$ paths + #disjoint $B \to A$ paths $\geq 2k+1$.*

Theorem 2. *There exists an efficient protocol when there are $2k+1-u$ disjoint $A \to B$ paths that are also disjoint from the $B \to A$ paths (for any u, $1 \leq u \leq k$).*

Explain here: case $u = 1$ and $k = 1$.



6. A t-BOUNDED ADVERSARY: NO ERROR

i.e. $\delta = 0$

Theorem 3. *It is necessary to have: # vertex-disjoint $A \to B$ paths*

- $\geq 2k+1$, *and*
- $\geq 3(k-u)+1$,

Theorem 4. *There exists an efficient protocol when there are $3k+1-u$ disjoint $A \to B$ paths that are also disjoint from the $B \to A$ paths (for any u, $0 \leq u \leq k$).*

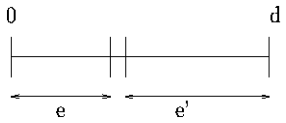
Lemma 1. *Let:*

- e *be the # errors that can be corrected,*
- $e' \geq e$ *be the # errors that can be detected simultaneously, and*
- d *be the minimum distance of the code.*

Then the necessary and sufficient conditions are that:

$$e \leq \lfloor \frac{d-1}{2} \rfloor, \quad \textit{and} \quad e + e' \leq d - 1.$$

Proof:



□

We use MDS-code as a perfect secret sharing,
$k+1$-out-of-$3k+1-u$ (input: secret + k random values). We call this an MDS secret sharing. Note:
$d - 1 = 3k + 1 - u - (k+1) = 2k - u = (k-u) + k$.

For simplicity we focus on the case $u = 1$, i.e. $3k$ disjoint $A \to B$ paths and 1 $B \to A$ path, disjoint from the $A \to B$ path.

Protocol (sketch)

Step 1 A sends share$_i$ of key via $A \to B$ path i.
Step 2 From the "shares" B detects whether #errors $\leq k-u$. If, then B corrects and send A STOP, else ask help to A (by sending back all "shares").
Step 3 If A receives STOP, A stops, else A finds "dishonests" $A \to B$ path and reliably sends this to B.
Step 4 If B found key then ignore step, else B finds key from honest shares.
Step 5 key used to send message from A to B.

General protocol (rough sketch)

Step 1 Split key into R_0, R_1. A sends share$_i$ of R_0.

Step 2 If # errors $\leq k-u$, then B recovers R_0, else B asks A's help (note: A may receive u different versions of "help", but one is correct).

Step 3 A sends share$_i$ of R_1.

Step 4 If # errors $\leq k-u$, then B recovers R_1, else B asks A's help if not asked before, else (assuming k' dishonest $A \rightarrow B$ have been identified) restart from Step 1 using $(k+1)$-out-of-$3k+1-u-k'$ MDS secret sharing however it will be used to detect errors only.

Corollary 2. *Can be extended to allow that of the $3k+1-u$ $A \rightarrow B$ paths only $3k+1-2u$ are node disjoint from the u $B \rightarrow A$ paths.*

Corollary 3. *(New) If there are $2k+1$ disjoint $A \rightarrow B$ paths and $k+1$ disjoint $B \rightarrow A$, then these do not have to be mutually disjoint.*

Results improved recently (Wang-Desmedt, unpublished):

Theorem 5. *Assume that there are u directed node disjoint paths from B to A, vertex disjoint from the forward channels. Then a necessary and sufficient condition for private message transmission from A to B against a t-active adversary is that there are $\max\{3t+1-2u, 2t+1\}$ directed node disjoint paths from A to B.*

7. Adversary structure

Let V be the nodes in the network. An adversary structure $\mathcal{A}_V$ over V is a subset of the power set 2^V such that if $B \in \mathcal{A}_V$ then subsets of B are also in $\mathcal{A}_V$.

If $\mathcal{Z}_1$ and $\mathcal{Z}_2$ are adversary structures for P, then

$$\mathcal{Z}_1 + \mathcal{Z}_2 = \{Z_1 \cup Z_2 : Z_1 \in \mathcal{Z}_1, Z_2 \in \mathcal{Z}_2\},$$

which is also an adversary structure for P.

$2\mathcal{Z}$ and $3\mathcal{Z}$ indicate $\mathcal{Z}+\mathcal{Z}$ and $\mathcal{Z}+\mathcal{Z}+\mathcal{Z}$ respectively.

Definition 1. Let $G(V,E)$ be a directed graph, A, B be nodes in $G(V,E)$, and $\mathcal{Z}$ be a an adversary structure on $V \setminus \{A,B\}$.

- A, B are called *$\mathcal{Z}$-separable* in G, if there is a set $Z \in \mathcal{Z}$ such that all paths from A to B go through at least one node in Z. We say that Z separates A and B.
- A, B are called *$(\mathcal{Z}+1)$-connected* if they are not $\mathcal{Z}$-separable in G.

A necessary and sufficient condition for A and B to privately communicate in the presence of a Byzantine adversary, in the case all communication links (edges in the graph) are:

two-way is that A, B are $(2\mathcal{Z}+1)$-connected in G (Kumar-Goundan-Srinathan-Rangan, 2002).

one-way without feedback, is that A, B are $(3\mathcal{Z}+1)$-connected in G (Desmedt-Wang-Burmester, 2005: see further).

The general case, i.e. with feedback channels has not been studied.

passive adversary

Desmedt-Wang-Burmester, 2005 observed the results of Franklin-Yung (related to partial broadcast) can easily be adapted to general adversary structure, i.e.

- a connectivity of $\mathcal{Z}+1$ and 1 strongly connected is necessary and sufficient, and
- a protocol has been proposed which is polynomial in $|V|$, the number of nodes in the graph, i.e. logarithmic in $|\mathcal{Z}|$.

Algorithm

A is the sender and B is the receiver.

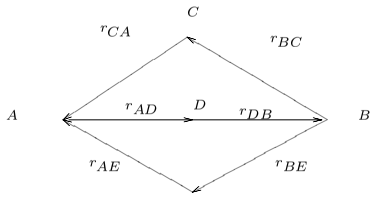
Step 1 For each edge e where u is the originator, u chooses a random message r_e and sends it to the recipient of that edge.

Step 2 Every node computes the sum of messages it has received and substracts the sum of messages it has sent out. If the node is the actual sender A, then it adds to this total the messsage M^A. Call this sum the "final result" for this node. Each final result, except the one of the actual receiver B, is propagated by the nodes openly to the receiver B.

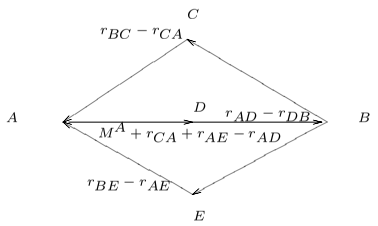
Step 3 B adds all final results, including his. The result is the message M^B.

Example:

Step 1



Step 2



Step 3 Easy to verify.

Active adversary, no privacy

Lemma 2. *Let $G = G(V, E)$ be a directed graph, A, B be nodes in G, and $\mathcal{Z}_1, \mathcal{Z}_2$ be adversary structures on $V \setminus \{A, B\}$. Then A, B are $(\mathcal{Z}_1 + \mathcal{Z}_2 + 1)$-connected if, and only if: for all sets $Z_1 \in \mathcal{Z}_1$ there is a set S_{Z_1} of paths between A and B such that,*

- *the paths in S_{Z_1} are free from nodes of Z_1,*
- *for every $Z_2 \in \mathcal{Z}_2$ there is at least one path in S_{Z_1} that is free from nodes of Z_2.*

Theorem 6. *Let $G = G(V, E)$ be a directed graph, A, B be nodes in G, and $\mathcal{Z}$ be an adversary structure on $V \setminus \{A, B\}$. We have $\mathcal{Z}$-reliable message transmission from A to B if, and only if, A, B are strongly $(2\mathcal{Z} + 1)$-connected in G.*

Algorithm

Assume that A, B are strongly $(2\mathcal{Z} + 1)$-connected in G

Let S be the set of all directed paths from A to B.

Step 1 For each path $p \in S$, A sends M^A to B over p.

Step 2 B receives M_p^B through path $p \in S$. B finds a node set $Z_1 \in \mathcal{Z}$ whose path set S_{Z_1} is such that the same message $M^B = M^A$ is received on all its paths.

Claim: $M^B = M^A$.

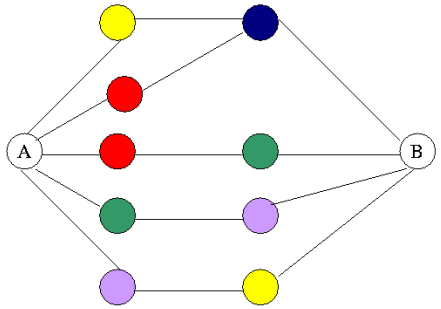
Indeed, suppose that the adversary selects $Z_2 \in \mathcal{Z}$. We have: by Lemma 2 that since A, B are $(2\mathcal{Z} + 1)$-connected, there will be a path $p_0 \in S_{Z_1}$ free from nodes of Z_2. On this path $M_{p_0}^B = M^A$. Since B receives the same message from all paths in S_{Z_1}, we must have

$M^A = M_{p_0}^B = M^B$.

It follows that B can reliably recover the message M^A.

An interesting adversary structure is the t-color adversary structure. A weakness of one router/computer can easily be exploited on another one if it runs the same platform. Vertices are given colors. t colors can be corrupted. It allows to model routers that run the same platform, i.e. have the same weakness, to be assigned the same color.

Color adversary structure is interesting to understand counter-intuitive arguments: i.e.: color separable is not linked to vertex disjoint paths.



New result:

Deciding whether a vertex colored graph with C the set of colors, is $\mathcal{Z}_{C,k}+1$-connected is co-**NP**-complete.

Proof We demonstrate the complementary problem is **NP**-complete. The reduction is from the Vertex Cover problem.

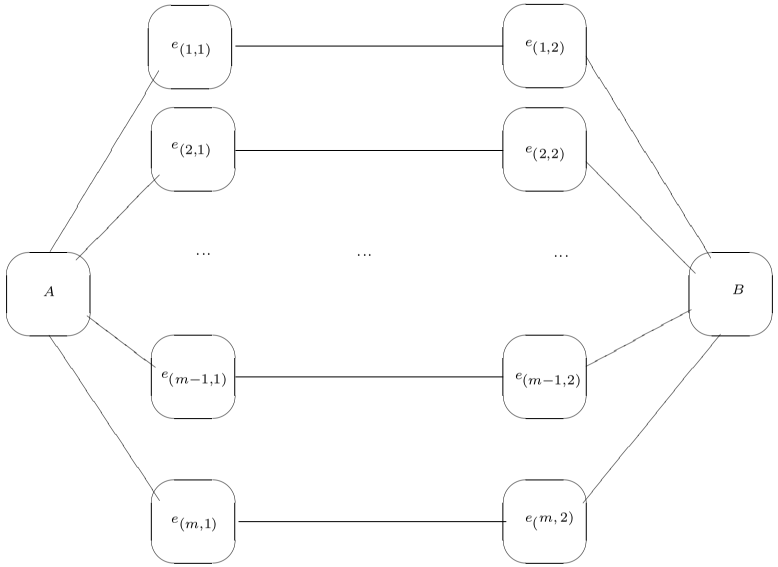
INSTANCE: A graph $G=(V,E)$ and a positive integer $k \leq |V|$.

QUESTION: Is there a vertex cover of size k or less for G, that is, a subset $V' \subseteq V$ such that $|V'| \leq k$ and, for each edge $(u,v) \in E$, at least one of u and v belongs to V'?

We now construct a network between A and B. Assume $E=\{e_1, e_2, \ldots, e_{m-1}, e_m\}$. Let us define:

- $E_1 = \{e_{(1,1)}, e_{(2,1)}, \ldots, e_{(m-1,1)}, e_{(m,1)}\}$ a set of nodes, and
- similarly E_2
- a bijection f_1 from E to E_1 such that $f_1(e_i) = e_{(i,1)}$, and
- similarly f_2 maps e_i into $e_{(i,2)}$.

We now construct the following new graph G_c:



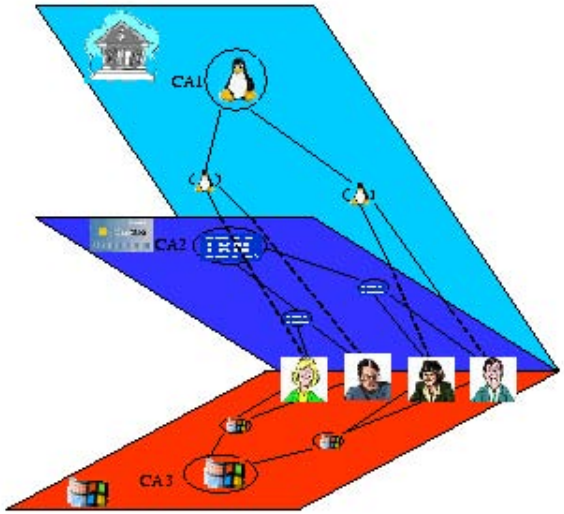
We now color the nodes in E_1 as following. Let $C = V$. Let $e_{(i,1)} \in E_1$. Let $(v_j, v_l) = f_1^{-1}(e_{(i,1)})$, where $j < l$. Color $e_{(i,1)}$ using color v_j. Similar for coloring the nodes in E_2, but we use v_l.

The graph G has a vertex cover of size k if and only if in G_c there are k colors which will disconnect A from B.

Definition 2. Let $G(V, E)$ be a directed graph, A, B be nodes in G, S be a set of simple paths in G between A and B, and G_S be the graph obtained by removing all nodes and edges of G not in S. Let $\mathcal{Z}$ be an adversary structure. We say that S is a minimal $(\mathcal{Z}+1)$-connected path-set from A to B in G, if

1. A and B are $(\mathcal{Z}+1)$-connected in G_S, and
2. for each path $p \in S$, A and B are $\mathcal{Z}$-separable in $G_{S\setminus\{p\}}$.

Theorem 7. *Let $G = G(V, E, C, f)$ be a colored graph which is $(\mathcal{Z}_{C,k}+1)$-connected. If the number of colors is minimal then the paths in a minimal path-set are node-disjoint and each path is monochrome (all nodes on one path have the same color).*



8. CENSORSHIP

Censorship has been an important tool used by authorities for different reasons. Examples:

- due in part to St. Irenaeus, a second-century bishop of Lyon, gospels other than the ones by Matthew, Mark, Luke, and John, were censored by the Church,
- due to the “Tribunal de Grande Instance de Paris Ordonnance de référé du 22 mai 2000” the book “Mein Kampf” is censored in France,
- in the US, the Rolling Stones performance during the 2006

superbowl on February 5 was censored.

Military classified material is another example.

So, a natural question is how above research can be used to censor the internet. Note: in the case of the gospel, the network was a virtual one.

When a network is designed, the authority may want to know whether it can be censored.

If the security model is an ordinary threshold one, then anybody knows who can/can not censor. If the color adversary structure is used, then the problem whether it is (at most) k-color connected, is **NP**-complete. So, the secret is a separator Z of at most k colors.

Why should this remain secret? Advantage: it may be hard for the adversary to find the secret.

So, the question becomes:

> Can the designer prove in zero-knowledge the existence of a k-color separator?

Designing an efficient zero-knowledge proof seems rather trivial.

Here an idea:

Step 1 The prover permutes all the vertices, and permute all the colors and commits to these.

Step 2 The verifier asks a binary question.

Step 3 If the question is 0, then the prover opens all commitments, else he reveals a set V' that separates A and B in this isomomorphic graph.

Step 4 The verifier, in the first case, checks the commitment. In the else case, the verifier checks that the number of colors in V' is at most k and checks V' indeed separates.

Unfortunately, above protocol is not zero-knowledge. Indeed, it

leaks the size of V', which it should not. The knowledge of the size of V' may help the verifier to find the k colors. Moreover, it also leaks the multiplicity of each color, etc.

To solve this problem, we prove the following lemma.

Lemma 3. *Let $G_c = G_c(V, E, C, f)$ be a vertex-colored graph, where C is the set of colors and $f : V \to C$. Let $C' \subseteq C$ be such that $|C'| = k$ and $V' = \{v'_i : f(v'_i) \in C'\}$ separate A and B. Let k' be the maximum number of vertex disjoint paths in (V, E)* ignoring the colors*. Let P_1, P_2, ..., $P_{k'}$ be these vertex disjoint paths. We then have that for each of these path P_i: $P_i \cap V' \neq \emptyset$. So, on each path P_i there exists a node of a color in C'.*

Proof: The proof follows trivially by contradiction. □

Zero-Knowledge interactive proof

Setting

Let $G = G(V, E, C, f)$ be a vertex-colored graph and $m = |C|$. For simplicity we assume $C = (1, 2, \ldots, m)$. Let C' and V' be as before.

Precomputation

First the verifier and the prover (separetely) compute:

- k', i.e. the maximum number of vertex disjoint paths ignoring colors.
- k' vertex disjoint paths P_1,P_1, P_2, ..., $P_{k'}$.

This can be done in polynomial time. So both prover and verifier obtain the same k' vertex disjoint paths. Let l_i be the length of the path P_i minus one, and let us call the vertices, except A and B, on

this path $v_{(i,1)}$, $v_{(i,2)}$,...,$v_{(i,l_i)}$.

Protocol

They repeat the following steps n times, where n is specified later. The randomness in each run is chosen independently.

Step 1 The prover chooses a permutation π of the colors, so $\pi \in_R sym(\{1, \ldots, m\})$. For each of the aforementioned paths P_i:

- the prover chooses a permutation $\rho_i \in_R sym(\{1, \ldots, l_i\})$, permutes the vertices (ignoring A and B) on the path P_i and sends the verifier a commitment for the permuted coloring of the permuted vertices, so formally, sends:

$$E_{(i,j)} = \text{commit}(\pi(f(v_{(i,\rho_i(j))})), r_{ij}) \text{ for } j = 1, \ldots, l_i,$$

where r_{ij} is chosen independently uniformly random, and

- for each $c_h \in C'$ ($h = 1, \ldots, k$) sends $E'_h = \text{commit}(\pi(c_h), r'_h)$, where r'_h is chosen independently uniformly random.

Step 2 The verifier flips a coin q_1 and also chooses randomly a value $q_2 \in_R \{1, \ldots, k'\}$ and sends the prover the query (q_1, q_2).

Step 3 *If* $q_1 = 0$, *then* the prover reveals π, all ρ_i and opens all commitments of the type $E_{(i,j)}$ (Note the prover does not open E'_h.),
else the prover decommits one (of the) permuted colors of the vertex set: $P_{q_2} \cap V'$. This is done by opening:

- exactly one $E_{(q_2,j')}$, and
- exactly one E'_h

such that $f(v_{(q_2,\rho_{q_2}(j'))}) = c_h$. (Note π is not opened, and neither is ρ_{q_2}).

Step 4 *If* $q_1 = 0$, *then* the verifier verifies that π and all ρ_i are permutations and all the decommitted values,
else the verifier checks that the two opened commitments and checks that they correspond to the same color.

Theorem 8. *When* n *is chosen such that* $((k'-1)/k')^n$ *is negligible, the protocol is a computational zero-knowledge interactive proof system for the color separable problem assuming that the commitment function* commit *is secure.*

Questions: I have been asked how above research can be used to:

- prevent censorship
- help in propaganda

9. FINDING THE NETWORK GRAPH WHILE UNDER ATTACK

Model

Similar to classical one:
Input: each node knows its neighbors only.
Question: find the network
Different:

- t-bounded Byzantine adversary. Special attack: can claim non-existent nodes exist.
- use a communication network (which is not immediately reconstructed)
- use a PGP like certificate graph (virtual graph).

Solutions: polynomial time solution by Burmester-Desmedt (1999) if the PGP like network is $2t+1$ strongly connected.

Lots of hueristics afterwards.

More details (sketch):

Impossible to obtain the graph G, but at best a good approximation G'.

Client: wants to find network graph. Servers: are the other nodes.

Initially: nodes only know their neighbors.

Use Round-Robin to avoid flooding. All communication is signed and all signatures are verified on consistency.

Phase 1 Not all honest nodes have been found.
Servers provide client with the list of their neighbors (one at a time).

Client checks whether all nodes that are real and responding have been found. If so, move to the next phase.

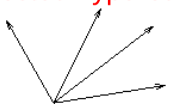
Phase 2 Same as above. However, nodes that claim new nodes are being found are declared dishonest.

10. PARTIAL BROADCAST: PASSIVE ADVERSARY

t-bounded adversary only (some results on general adversary structure easily extend). We distinguish:

Franklin-Yung (1995,2004) replaced the point-to-point network by a partial broadcast. They use a directed hypergraph. A directed hypergraph $H = (V, E)$ consists of set of vertices V, however a directed hyperedge $e \in E$ has the form (v, V'), where $v \in V$ and $V' \subset V$. When the node v uses this directed hypered all nodes in V' receive the same information (others learn nothing about that information).

Directed hyperedge



Definition 3. A hypergraph is strongly k-connected (weakly k-connected) if for all nodes $x, y \in V$, and for all $U \subset V \setminus \{x, y\}$, $|U| < k$, there remains a directed (undirected) path from x to y after the removal of U and all hyperedges (v, S) such that $U \cap (S \cup \{v\}) \neq \emptyset$.

Franklin-Yung (1995,2004) focussed on a passive adversary who eavesdrops. They demonstrated:

- that a necessary and sufficient condition for private communication secure against a t-bounded passive adversary is that the directed hypergraph is strongly 1-connected and weakly $(t+1)$-connected.
- proposed a protocol (not necessarily polynomial time).

Algorithm

A is the sender and B is the receiver.

Step 1 For each hyperedge e where u is the originator, u chooses a random message r_e and sends it to the recipients of that hyperedge.

Step 2 Every node computes the sum of messages it has received and substracts the sum of messages it has sent out. If the node is the actual sender A, then it adds to this total the messsage M^A. Call this sum the "final result" for this node. Each final result, except the one of the actual receiver B, is propagated by the nodes openly to the receiver B.

Step 3 B adds all final results, including his. The result is the message M^B.

Example: see before.

Franklin-Yung also introduced special cases, one of these is called a neighbor network, which can be represented by an ordinary graph. Ethernets are a special case of these.

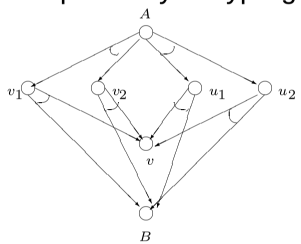
In this graph if a vertex broadcast a message, all its neighbors will receive identically the same information.

11. Active adversary (not eavesdropping)

Strongest result today by Desmedt-Wang (2002):

Definition 4. Let $H(V, E)$ be a hypergraph, $A, B \in V$ be distinct nodes of H, and $k \geq 0$. A, B are k-separable in H if there is a node set $W \subset V$ with at most k nodes such that any directed path from A to B goes through at least one node in W. We say that W separates A, B.

Remark. Note that there is no straightforward relationship between strong connectivity and separability in hypergraphs.



Theorem 9. *The nodes A, B of a hypergraph H is not $2k$-separable if and only if there are $2k+1$ directed node disjoint paths from A to B in H.*

Proof: This follows directly from the maximum-flow minimum-cut theorem in classical graph theory. □

Theorem 10. *A necessary and sufficient condition for reliable message transmission from A to B against a k-active adversary is that A and B are not $2k$-separable in H.*

Proof: First assume that A and B cannot be separated by a $2k$-node set. By Theorem 9, there are $2k+1$ directed node disjoint paths from A to B in H. Thus reliable message transmission from A to B is possible.

Next assume that A, B can be separated by a $2k$-node set W in H. We shall show that reliable message transmission is impossible. Suppose that π is a message transmission protocol from A to B and let $W = W_0 \cup W_1$ be a $2k$-node separation of A and B with W_0 and W_1 each having at most k nodes. Let m_0 be the message that A transmits. The adversary will attempt to maintain a simulation of the possible behavior of A by executing π for message $m_1 \neq m_0$. The strategy of the adversary is to flip a coin and then, depending on the outcome, decide which set of W_0 or W_1 to control. Let W_b be the chosen set. In each execution step of the transmission protocol, the adversary causes each node in W_b to follow the protocol π as if the protocol were transmitting the message m_1. This simulation will succeeds with nonzero probability. Since B does not know whether

$b = 0$ or $b = 1$, at the end of the protocol B cannot decide whether A has transmitted m_0 or m_1 if the adversary succeeds. Thus with nonzero probability, the reliability is not achieved. □

12. ACTIVE AND EAVESDROPPING ADVERSARY

Has primarily been studied for the case of neighbor networks.

Franklin and Wright (1998,2000) introduced the concept of interiorly neighborhood-disjoint lines.

Formally, A and B are connected by n interiorly neighborhood-disjoint lines if there are n lines $p_1, \ldots, p_n \subseteq V$ with the following properties:

- For each $1 \leq j \leq n$, the j-th line p_j is a sequence of $m_j + 2$ nodes $A = X_{0,j}, X_{1,j}, \ldots, X_{m_j+1,j} = B$ where $X_{i,j}$ is a neighbor of $X_{i+1,j}$.
- For each i_1, i_2, j_1, and j_2 with $j_1 \neq j_2$, the only possible common neighbors of X_{i_1,j_1} and X_{i_2,j_2} are A and B.

Franklin-Wright (1998,2000) proved that:

- If $n > t$, $\delta > 0$ and $\varepsilon > 0$, then there is an efficient (ε, δ)-secure message transmission protocol between A and B.
- If $n > \lceil 3t/2 \rceil$ and $\delta > 0$, then there is a δ-reliable and perfectly private message transmission protocol.
- If $t < n \leq \lceil 3t/2 \rceil$ and $\delta > 0$, then there is an exponential bit complexity $(0, \delta)$-secure message transmission protocol between A and B.

They opened the question:

> is it possible to efficiently achieve perfect privacy when $t < n \leq \lceil 3t/2 \rceil$, i.e. a polynomial time $(0, \delta)$-secure message transmission protocol?

Wang-Desmedt (1999,2001) gave an affirmative answer using a constructive proof.

They extended the results by introducing:
A and B are weakly (n,t)-connected (Wang-Desmedt)

- if there are n vertex disjoint paths $p_1,\ldots,p_n$ between A and B and,
- for any vertex set $T \subseteq (V \setminus \{A,B\})$ with $|T| \le t$, there exists an i $(1 \le i \le n)$ such that all vertices of p_i have no neighbor in T.

If A and B are weakly (n,t)-connected for some $t < n$, then there is a perfectly private transmission protocol which is an efficient $(0,\delta)$-secure message transmission protocol between A and B.

Wang-Desmedt posed as open problem whether:

the weakly (n,t)-connectivity condition is necessary.

Desmedt-Wang (2002) gave a counter example.

The neighbor network G in Figure 1 is not weakly $(2,1)$-connected.

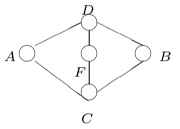


Figure 1: A counterexample

In the following we present a $(0,\delta)$-secure message transmission protocol against a 1-active adversary from A to B.

Franklin-Wright protocols we need

Basic Propagation Protocol (Franklin and Wright)

In this protocol, A tries to propagate a value s^A to B.

- In round 1, A multicasts s^A.
- In round ρ for $2 \le \rho \le m+1$, each $X_{\rho-1,j}(1 \le j \le n)$ expects to receive a single element from $X_{\rho-2,j}$. Let $u_{\rho-1,j}$ be this value if a value was in fact received, or a publicly known default element otherwise. At the end of round ρ, $X_{\rho-1,j}$ multicast $u_{\rho-1,j}$.
- In round $m+2$, B receives a single element from each $X_{m,j}$, or substitutes the default element. Let s^B_j be the value received or substituted on line j.

From now on when a party substitutes the default element, we just say that the party substitutes.

Full Distribution Protocol (Franklin and Wright)

In this protocol, each internal node $X_{i,j}$ tries to transmit an element $s_{i,j}$ to both A and B.

- In round 1, each $X_{i,j}(1 \le i \le m, 1 \le j \le n)$ multicast $s_{i,j}$ to $X_{i-1,j}$ and $X_{i+1,j}$.
- In round ρ for $2 \le \rho \le m+1$:
  - For $1 \le j \le n$ and $\rho \le i \le m$, each $X_{i,j}$ expects to be the intended recipient of an element from $X_{i-1,j}$ (initiated by $X_{i-\rho+1,j}$). Let $u_{i,j}$ be the received value or a default value if none is received.
  - For $1 \le j \le n$ and $1 \le i \le m-\rho+1$, $X_{i,j}$ expects to be the intended recipient of an element from $X_{i+1,j}$ (initiated by $X_{i+\rho-1,j}$). Let $v_{i,j}$ be the received value or a default value if none if received.

– For $1 \leq j \leq n$, B expects to be the intended recipient on the j-th line of a single element (initiated by $X_{m-\rho+2,j}$). Let $s^B_{m-\rho+2,j}$ be the received value or a default value if none is received.

– For $1 \leq j \leq n$, A expects to be the intended recipient on the j-th line of a single element (initiated by $X_{\rho-1,j}$). Let $s^A_{\rho-1,j}$ be the received value or a default value if none is received.

– $X_{i,j}$ multicasts $u_{i,j}$ to $X_{i+1,j}$ if $\rho \leq i \leq m$, and $v_{i,j}$ to $X_{i-1,j}$ if $1 \leq i \leq m-\rho+1$.

Fact: (Franklin and Wright) If there are no faults on the j-th line, then $s^A_{i,j} = s^B_{i,j}$ for all $1 \leq i \leq m$. Further, if $X_{i,j}$ is the only fault on the j-th line, then $s^A_{i,j} = s^B_{i,j}$.

In this protocol, A reliably transmits a message M^A to B.

Definition 5. We let $\text{auth}(M; a, b) := aM + b$.

Note that each authentication key $key = (a, b)$ can be used to authenticate one message M without revealing any information about any component of the authentication key.

Reliable Transmission Protocol (Franklin and Wright)

- The nodes on all the n lines execute an instance of the Full Distribution Protocol, which takes place during rounds 1 through $m+1$. The element that $X_{i,j}$ initiates is $(a_{i,j}, b_{i,j})$ which is randomly chosen from $\mathbf{F}^2$. Let $(a^A_{i,j}, b^A_{i,j})$ and $(a^B_{i,j}, b^B_{i,j})$ be the values that A and B receive or substitute as the element initiated by $X_{i,j}$.
- The nodes on all the n lines execute an instance of the Basic

Propagation Protocol from A to B, which takes place during rounds $m+2$ through $2m+3$. The element that A initiates is $\{(i, j, M^A, \text{auth}(M^A; a^A_{i,j}, b^A_{i,j})) : 1 \leq i \leq m, 1 \leq j \leq n\}$. In round $2m+3$, B receives or substitutes $\{(i, j, M^B_{i,j,k}, u^B_{i,j,k}) : 1 \leq i \leq m, 1 \leq j \leq n\}$ on the k-th line, $1 \leq k \leq n$.

- Let $r_k(M) := \{j : \exists i (M = M^B_{i,j,k}$ and $u^B_{i,j,k} = \text{auth}(M^B_{i,j,k}; a^B_{i,j}, b^B_{i,j}))\}$, that is, for any message M, $r_k(M)$ denotes the set of all line indices j such that M is "correctly" authenticated by some keys $(a^B_{i,j}, b^B_{i,j})$ according to the information B received on the k-th line. B outputs M^B that maximizes $\max_k |r_k(M^B)|$.

Fact: (Franklin and Wright) If $\delta > 0$, $n > t$, and $|\mathbf{F}| > mn^2/\delta$, then the Reliable Transmission Protocol is an efficient δ-reliable message transmission protocol.

$(0, \delta)$-secure message transmission protocol against a 1-active adversary from A to B.

Step 1 A chooses two random pairs $(r^A_1, r^A_2) \in_R \mathbf{F}^2$ and $(r^A_3, r^A_4) \in_R \mathbf{F}^2$. A sends (r^A_1, r^A_2) to C and (r^A_3, r^A_4) to D.

Step 2 B chooses two random pairs $(r^B_1, r^B_2) \in_R \mathbf{F}^2$ and $(r^B_3, r^B_4) \in_R \mathbf{F}^2$. B sends (r^B_1, r^B_2) to C and (r^B_3, r^B_4) to D.

Step 3 C chooses a random pair $(a_1, b_1) \in_R \mathbf{F}^2$. C sends $(a_1 + r^A_1, b_1 + r^A_2)$ to A and $(a_1 + r^B_1, b_1 + r^B_2)$ to B.

Step 4 D chooses a random pair $(a_2, b_2) \in_R \mathbf{F}^2$. D sends $(a_2 + r^A_3, b_2 + r^A_4)$ to A and $(a_2 + r^B_3, b_2 + r^B_4)$ to B.

Step 5 From the messages received from C and D, A computes (a^A_1, b^A_1) and (a^A_2, b^A_2).

Step 6 From the messages received from C and D, B computes (a^B_1, b^B_1) and (a^B_2, b^B_2).

Step 7 B chooses a random $r \in_R \mathbf{F}$, computes $s_1 = \text{auth}(r; a_1^B, b_1^B)$ and $s_2 = \text{auth}(r; a_2^B, b_2^B)$. Using the probabilistically reliable message transmission protocol of Franklin and Wright B transmits $\langle r, s_1, s_2 \rangle$ to A.

Step 8 Let $\langle r^A, s_1^A, s_2^A \rangle$ be the message received by A in the last step, A computes the key index set $K_{\text{index}} = \{i : s_i^A = \text{auth}(r^A; a_i^A, b_i^A)\}$. A also computes the shared secret $K^A = \sum_{i \in K_{\text{index}}} a_i^A$.

Step 9 Using the probabilistically reliable message transmission protocol of Franklin and Wright, A transmits $\langle K_{\text{index}}, M^A + K^A \rangle$ to B, where M^A is the message.

Step 10 Let $\langle K_{\text{index}}^B, c^B \rangle$ be the message that B received in the last step. B computes the shared secret $K^B = \sum_{i \in K_{\text{index}}^B} a_i^B$, and decrypts the message $M^B = c^B - K^B$.

13. ADDING JAMMING

We distinguish between:

13.1. Jamming adversary only in traditional broadcast

Well known approach:

Sender and receiver agree on random "seed," and use a psuedo-random generator for frequency hopping.

More interesting case:

t receivers collaborate with adversary. Above fails. The problem and solutions go back to the the 1960's, e.g. Kautz-Singleton, 1964.

Solution:

Superimposed codes (also viewed as a covering problem: Erdós-Frankl-Furedi), i.e.:

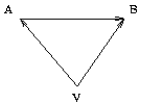
Use set of indices of seeds. Each receiver is given a "block" (subset). The blocks are such that: none is covered by the union of t others.

Lots of research, in particular by Russian scientists, such as: Dyachkov-Rykov, Levenshtein (jointly with Ericson).

(See also Desmedt-Safavi Naini-Wang-Batten-Charnes-Pieprzyk, 2000 for bounds.)

13.2. JAMMING IN NETWORKS

Third party can prevent communication between two parties. Studied by Desmedt-Wang-Safavi Naini-Wang (2005):



V can jamm nodes A and/or B

Model

Use radio network, i.e. a directed colored-edge multigraph $R(V, E, F, c)$, where V is the node set, E is the directed edge set, F is the frequency (color) set, and c is a map from E to F (the map c assigns a frequency to each edge).

Cases (jamming only):

Receiver-jamming: A node v can receiver-jam on a frequency f if there is a directed edge e from v to some node u with $c(e) = f$. The result of receiver-jamming by v on frequency f is that for any node u such that there is a directed edge e from v to u, u cannot receive any message transmitted on the frequency f by any node.

Sender-jamming: A node v can sender-jam on a frequency f if there is a directed edge e from v to some node u with $c(e) = f$. The result of sender-jamming by v on frequency f is that for any node u such that there is a directed edge e from v to u, u cannot send any message on the frequency f to any node.

Destroy-jamming: A node v can destroy-jam on a frequency f if

there is a directed edge e from v to some node u with $c(e) = f$. The result of destroy-jamming by v on frequency f is that for any node u such that there is a directed edge e from v to u, u cannot receive or send any message on any frequency.

Receiver jammers only (no privacy)

Let $R(V, E, F, c)$ be a radio network, and $S \subset V$ be a node set.

The reduced radio network $R(V \setminus S, E_{V \setminus_{rj} S}, F, c)$ is defined by letting $E_{V \setminus_{rj} S} = E \setminus E_S^{rj}$, where E_S^{rj} is the set of the following directed edges:

- all edges originated from nodes in S.
- all edges e from u to v such that there is an edge e' from some node in S to v and $c(e) = c(e')$.

Theorem 11. *Reliable message transmission from u to v in a radio network $R(V, E, F, c)$ against a t-receiver-jamming adversary is possible if and only if for any t-node set S, there is a directed path from u to v in the reduced radio network $R(V \setminus S, E_{V \setminus_{rj} S}, F, c)$.*

Other cases studied

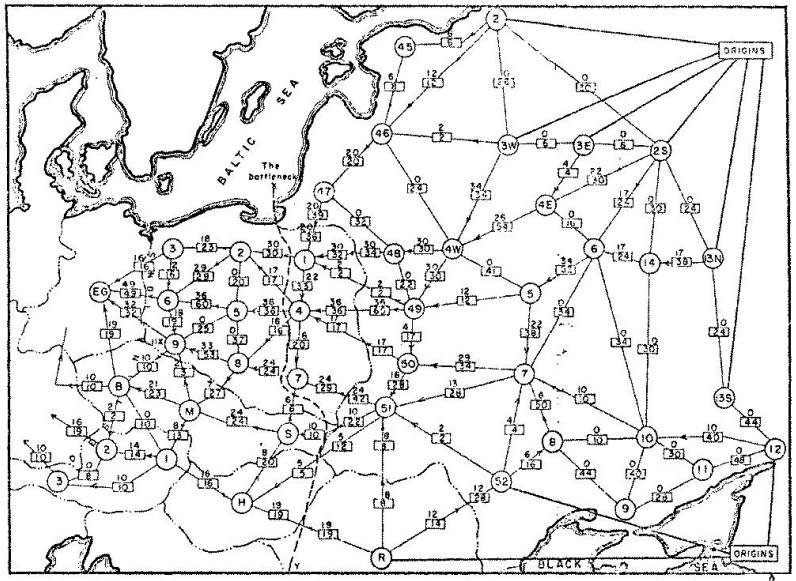
- receiver-and-sender jammers
- destroy-jamming
- adding active (Byzantine) adversaries
- adding privacy

Theorems are similar.

14. EXTENSIONS

Immediate applicable outside cryptography:

In the 1950s, Harris-Ross, of the US Air Force, analyzed the rail network that linked the Soviet Union to its satellite countries in Eastern Europe using graph theory. They used it to evaluate the maximum flow of goods via rail from the Soviet Union to Eastern Europe and which rail links to blow up to disrupt the network.

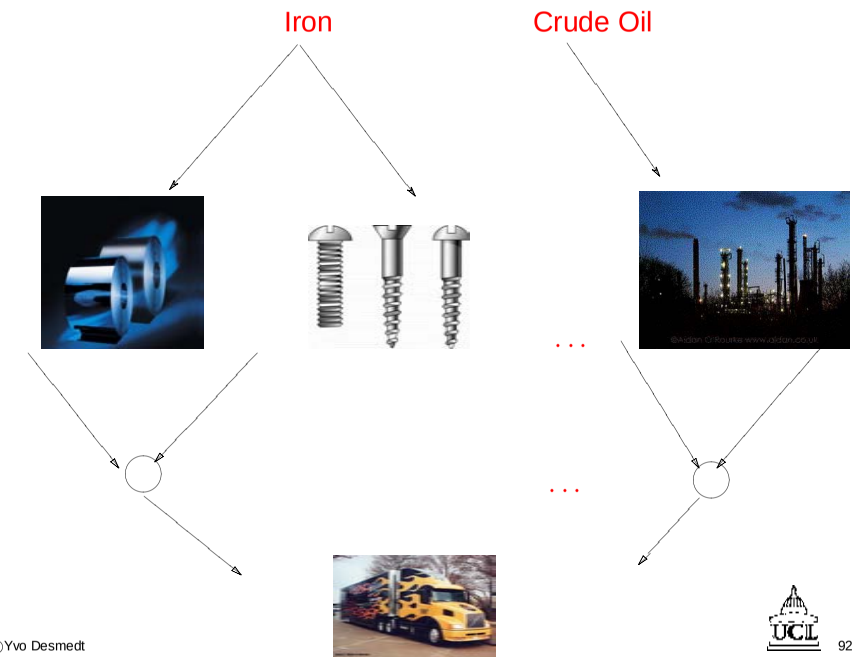


Sources: e.g., Alexander Schrijver, “On the history of the transportation and maximum flow problems”October 26, 2001. It states:

> the interest of Harris and Ross was not to find a maximum flow, but rather a minimum cut

Extensions:

Above works for distribution networks. What about the following:



Above is a PERT (Program Evaluation and Review Technique) graph. It is unable to model redundancy.

Solution: use AND/OR graphs.

Interesting DOS problems.

15. Conclusions

Open problems (R&D):

- Updating a current network: only 1 known result! Very important in practical world.
- Partial broadcast:
  - Necessary and sufficient conditions for ethernet type networks.
  - More results on how to construct such networks.
  - Jamming: plenty of decisional as well as computational questions.
- Privacy: lots of open problems in partial broadcast.
- Unknown graph: Is $2k+1$ in Burmester-Desmedt necessary? Neighbor networks suggest it may not.

- Adversary controls nodes: makes historic sense. What if the adversary structure is over nodes and edges, i.e. the adversary can destroy edges? (Easy to solve for point-to-point networks when dealing with destructive, i.e. not Byzantine, attacks only.)
  Reason: in reality nodes could be better protected.

We do not pretend to have surveyed all related papers on the topic. For example: the work on private/reliable communication in which two graphs are used one being the communication graph and the other an “authentication graph” where an edge indicates that parties share a secret key (See Beimel-Franklin 1997 and Beimel-Malka 2005.)