An intrusion-tolerant security server for an open distributed system

*Laurent Blain** *Yves Deswarte***

*LAAS-CNRS **LAAS-CNRS & INRIA
7, Avenue du Colonel Roche
31077 TOULOUSE Cedex
France

Abstract

This paper describes a new approach for security in open distributed systems. This approach is currently developed in the framework of the Delta 4 project. After a few reminders about two existing distributed security architectures, the proposed "intrusion-tolerant" approach is specified. It is based on a fragmentation-scattering technique applied to a security server running on several security sites. These sites are such that intrusions into a number of sites less than a given threshold have no consequence on the global security. The different security services provided are then presented together with a proposed multi-categories discretionary policy.

Résumé

Ce papier décrit une nouvelle approche pour la sécurité dans les systèmes répartis ouverts. Cette approche est actuellement développé dans le cadre du projet ESPRIT Delta-4. Après quelques rappels sur deux architectures de sécurité existantes, l'approche "tolérante aux intrusions" proposée est spécifiée. Elle est basée sur la technique de fragmentation-dissémination appliquée à un serveur de sécurité réparti sur plusieurs sites de sécurité. Ces sites sont tels qu'un des intrusions dans un nombre de sites inférieur à un seuil donné n'ait pas de conséquences sur la sécurité globale. Les différents services de sécurité fournis sont ensuite présentés, ainsi qu'une politique d'autorisation discrétionnaire multi-catégories.

INTRODUCTION: SECURITY FOR AN OPEN DISTRIBUTED SYSTEM

The security approach presented here is developed within the framework of the Delta-4 project. Delta-4 means Definition and Design of an **open** Dependable **Distributed** architecture. Notice the two important characteristics: openness and distribution.

Openness means that security can be implemented on any machines. The security part of an open system must not add "too much" specific hardware and software. Furthermore, security must not be so much of a constraint such that it becomes difficult to add new hosts to the system. These two considerations involve important differences with approaches such as the Red Book criteria [NCSC 87a] which impose rigid uses of software and hardware.

Distribution involves the choice of a distributed model for the security architecture. A classical one is the client-server model. An application running on one or several hosts provides services to clients on the same or other hosts. The client (the server) does not need to know on which host the server (the client) resides. This corresponds to transparent distribution. Security can follow the same model.

These two characteristics (openness and distribution) led us to give a particular approach in order to apply our security techniques.

Firstly, to keep the openness and the flexibility of the system, the intrusion-tolerant security techniques will not be integrated into the basic system design, but will be provided only as tools and applications.

Secondly, each host is considered to be under the local control of either the local administrator (time-sharing system) or the user (workstation). Local security is ensured by these persons and only global security for remote access can be ensured by specific security components. No additional security components need to be added to every new host. If and only if this host needs to do remote protected accesses or needs to be protected from remote accesses, new software and hardware components will be incorporated.

Thirdly, the client-server model must be applied to security too. The objects to be protected are servers and so the requests/accesses from clients to server must be verified. Two solutions can be adopted in order to secure the servers. The first one is to provide tools permitting construction of a secured server with authentication, authorization, audit.... The problem of this solution is that of the cost of building a new protection system for each server and all the interfaces with clients. The second solution is to provide a set of security services available for all servers which need security and which become clients of these security servers. These services are also available for clients which want to access secured servers. The advantage of this solution is that the part which is implemented on the secured servers is relatively small. The major part of security is ensured by the security servers. This is equivalent to the Kerberos approach [MILL 87]. Kerberos provides only identification, authentication and audit, our approach includes sensitive data management and recovery.

The security services which are presented concern only distributed applications for which the location of application entities is transparent. In this sense, the security services can be viewed as complementary to network security services [ISO 7498-2] which are focused on communication confidentiality and integrity, rather than authorization to access application servers.

I. DISTRIBUTED SECURITY ARCHITECTURES

Our intrusion tolerant security service is a new concept with regard to existing architectures such as those implied by Orange Book [DOD 85], Red Book [NCSC 87a] and Kerberos [MILL 87, STEIN 88].

I.1 The Orange Book and Red Book architectures

The Trusted Computer System Evaluation Criteria (TCSEC), often called Orange Book, describes a multilevel mandatory policy and Bell-LaPadula model and their implementation

requirements. This architecture is based on two components: the **Trusted Computing Base (TCB)** and the **Security Kernel**.

• The TCB consists of all the of protection mechanisms within a computer system -- including hardware, firmware and software -- the combination of which is responsible for enforcing an authorization policy. The ability of a TCB to correctly enforce a authorization policy depends solely on the mechanisms within the TCB and on the correct input by system administrative personnel of parameters related to the authorization policy.

• The Security Kernel is the set of hardware, firmware and software elements of the TCB that implement the reference monitor concept. It is *trusted*, *tamperproof*, *always invoked* and *verifiable as correct*.

The Orange Book provides only for a centralized system. The National Computer Security Centre also provided a Trusted Network Interpretation (Red book) of the TCSEC; the same requirements are then applied in the context of distributed system. The Red Book extends the TCB and Security Kernel concepts to distributed configurations. The idea is to connect distributed components together so as to form a global secure network. The security functionalities may be distributed.

There exists a **Network Trusted Computing Base (NTCB)** for the network. The NTCB is partitioned in such a way that the set of the partitions constitutes a global TCB. But all local accesses must be mediated locally by local mechanisms. For components which are not hosts, functionalities can be reduced since such components are not concerned by all the rules of the network security policy. This is not so for a host. The local security mechanisms must perform local security policy, because there exist subjects and objects within the host.

Thus each host possesses (Fig I.1):

- subjects and objects which are local on the host (i.e. not distributed),
- a TCB as defined above,
- a NTCB partition which is the local part of the global TCB and which mediates all accesses from remote subjects to local objects or from local subjects to remote objects (the NTCB partition and the TCB are thus often merged),
- certain functionalities of the NTCB partition which can be used by other hosts,
- a Security Kernel which mediates *all* accesses, be they remote or local,
- a **Trusted Interface Unit (TIU)** managing communication security between hosts.

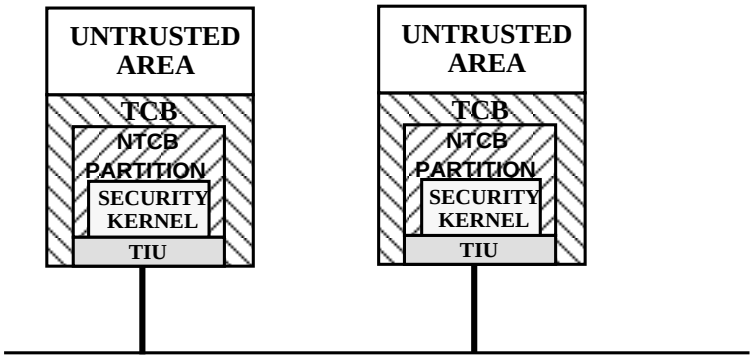


Fig. I.1. TNI Architecture.

This approach requires large trusted parts on all computers on the network. It is not a very open approach since it is difficult to add new computers to an existing network. The computer has to be evaluated in stand-alone mode and then in the network. Moreover, "trusted" is a very subjective property that is hard to verify and implies important software and hardware protections.

I.2 The Kerberos architecture

Kerberos was developed at MIT [STEIN 88, MILL 87] within the framework of the Athena project. The architecture of the Kerberos is based on the Client-Server model. Kerberos provides authentication and authorization services. In the Athena project, there exist different kinds of sites: unprotected Public workstations, private workstations, protected servers... Kerberos must authenticate legitimate users who want to use remote secured services. The workstations are not considered like time-sharing systems. They are totally under the control of the user, and local security services can be easily by-passed by the local user. Consequently, the network administration cannot trust any authentication carried out on a workstation. While a user accesses only local services, such local authentication is sufficient. But when the user wants to access remote services, the authentication must be carried out by a trusted service. This is the role of the Kerberos service, split in several security servers.

The network is seen as in Fig. I.2. Servers are trusted and so they are physically protected. Clients are not trusted. However anything occuring on the client host is under the responsibility of the user. And the server is under the responsibility of its administrator.

When a client logs in and wants to access a protected server:

- He asks the Key Distribution Server (KDS) for a session key.
- The KDS authenticates the client and gives him the session key.
- The client now asks the Ticket-Granting Server (TGS) for a ticket to access the protected server.
- The TGS provides a ticket to communicate with the server.
- The client uses this ticket to open a session with the server.
- It is the server itself which finally authorizes access.

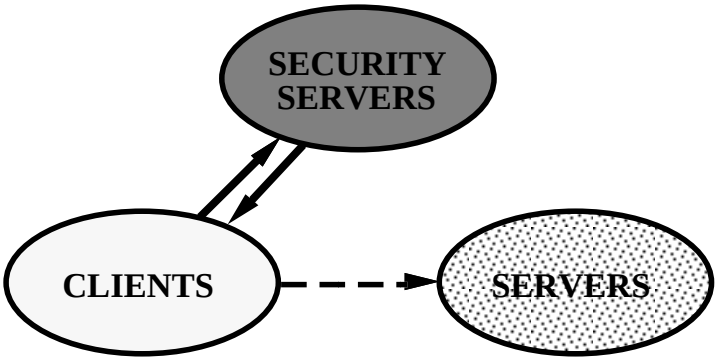


Fig. I.2. Kerberos Architecture.

The NTCB is in part on the Kerberos server (authentication) and in part on the secured server (authorization).

An essential characteristic of the Kerberos architecture is the centralization of servers. Indeed, the security services are not distributed. The Kerberos master server is replicated on passive slaves, which can replace the master when it fails. This replication is carried out by dumping the master Kerberos database every hour. This architecture has several potential weaknesses.

- The security administrator who manages the master server can misuse his priviledges to perform unauthorized actions. The security administrator must be trusted. Furthermore, if an intruder succeeds in penetrating the master server, global security is no longer ensured.
- The slaves can also give enough information to intruders who can read them to use it later in order to act as authorized users. Each slave must be very well protected.
- If Kerberos server fails, the last database changes are lost.

There exists a single point of failure, from the viewpoint of both accidental faults and intrusions.

The openness of the Kerberos architecture induces another drawback with respect to the Red Book proposals; it is not possible to enforce a mandatory access control policy by the Kerberos servers; nothing is provided to prevent "covert" channels which can be easily implemented by communications between workstations or by a memory channel within a workstation.

II. THE DELTA-4 INTRUSION TOLERANT APPROACH

An **intrusion** can be defined as a *deliberate interaction fault*. The definition of fault is given by [LAP 89]. Intrusions can be treated with the same means as for other faults (fault-avoidance and fault-tolerance). The means used to provide security in architectures described above is *intrusion-avoidance*. On the contrary, the means used in our approach is *intrusion-tolerance*. It is based on the fragmentation-scattering technique [FRAG 85, FRAY 86] in order to implement an intrusion-tolerant archive server and an intrusion-tolerant security server.

II.1 The fragmented-scattered archive service

In Delta-4, fragmentation-scattering was to implement a secure distributed archive service [RAN 88]. Data security is provided by intrusion-tolerance and more precisely by geographical fragmentation-scattering. The basis of the fragmentation and scattering technique in this case is to cut every sensitive file into several fragments in such a way that one or more fragments (but not all) are insufficient to reconstitute the file. These fragments are then stored in geographically distributed archive sites. An intruder who could access some sites cannot obtain all the fragments of the same file unless he has almost overall control of the complete distributed system. On the other hand, in order to ensure availability, several copies of each fragment are stored on different archive sites. This service thus answers all three security requirements: confidentiality, integrity and availability. Confidentiality and integrity are directly provided by fragmentation-scattering while availability is obtained by replication of fragmented data.

The archiving steps can be described as follows:

- Enciphering of the file with the fragmentation key (private to file's owner).
- Splitting of the file into fixed-length pages.
- Fragmentation of these pages, and fragment naming using the fragmentation key.
- Fragment replication.
- Transmission of fragment replicates to the archive sites.
- Agreement between sites according random parameters in order to decide where fragment replicates will be stored.

The intrusion-tolerant security server described in this paper provides the services that are needed to complete this secure file archive service; namely, user authentication, access-rights verification, protection of fragmentation keys, etc. The same basic services can of course be used in a more general context.

II.2 Requirements and assumptions for an intrusion-tolerant security server

The objectives of the intrusion tolerant approach are:

- openness and compatibility (for Delta-4 requirements), no specific hardware.
- reduction of TCB (by intrusion tolerance).
- modular security requirements.

The Security view of a Delta-4 network defines three kinds of sites (Fig. II.1):

- **User sites** are untrusted computers where users can log in. The local security is ensured by users.
- **Security sites** are computers providing security services: registration, identification-authentication, authorization, sensitive information management, audit and recovery service.
- **Particular servers** whose access needs to be secured. In the current system, this is the case of the Archive service located on Archive sites.

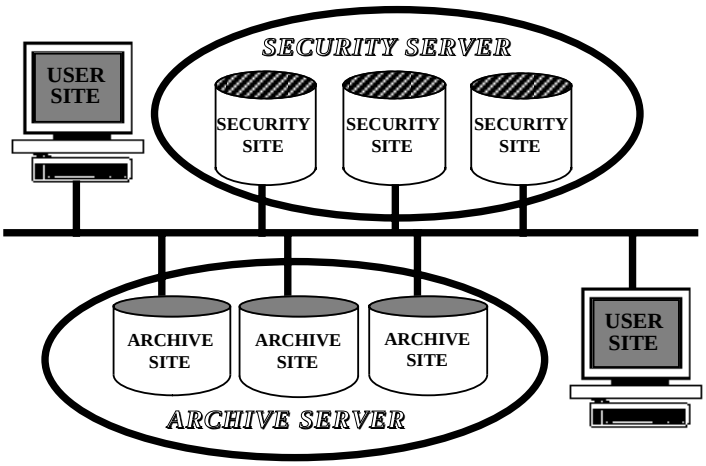


Fig. II.1. The different types of sites of the network.

The fault assumptions for security sites are:

- The probability of more than one intrusion before detection and recovery is small.
- The probability of an intrusion into a site is independent of the previous intrusion(s) in other site(s).

The characteristics of the intrusion-tolerant security approach are:

- An intrusion or a misuse on one security site is immediately masked and has no consequence on the service and on its properties.
- If errors occur on some security sites before recovery, the number which will be masked depends on the services and their properties (confidentiality, integrity, availability). The service performances can be degraded.

II.3 Intrusion tolerance for a security server

The different types of intrusion depend on who makes an intrusion.

- It can be somebody outside the system who tries to access it. This is the most well known kind of intruder, but not the most important. In this case the intruder has to *by-pass* physical, procedural and logical protections.
- The second kind of intruder is a user of the system who tries to access information or services without access rights. The intruder tries to *extend his priviledges*. This the most common intrusion. The intruder has "only" to by-pass the logical protection.
- The third - and the most dangerous - type of intruder is a security administrator who *uses his rights* to perform illegitimate actions. In this later case, the administrator has enough access-rights to do these actions but, according to the security policy, is not supposed to do them.

The architectures described in section I are intrusion avoidance architectures. A user/intruder **is trusted not to misuse his rights**. All the protection mechanisms must prevent the unauthorized actions. If an intruder succeeds in by-passing these protections, the security of the system is no longer ensured. If a security administrator decides to carry out illegal actions, there is no logical protection to prevent him from so doing. He could only be detected by using an intrusion-detection model [DENN 86] which is able to detect intrusions by monitoring system's audit records for abnormal patterns of system usage.

The principles of the intrusion tolerance are different. The system tolerates a bounded number of misuses. If one or more intruders by-pass the protection mechanisms and if the number of misuses they do is less than a given threshold, the security properties of the system (confidentiality, integrity and availability) are always ensured.

Three types of intrusion tolerance can be formulated:

- for **confidentiality**: read access to a subset of confidential data gives no information about the data.
- for **integrity**: the change of a subset of data does not change the data perceived by legitimate users.
- for **availability**: the change or deletion of a subset of data or of a server does not produce a denial of service to legitimate users.

For each property, a tolerance threshold is defined. If the reading, modification or destruction is done on a part D' of data/server D such that |D'| (size of D') is less than the threshold, the properties are always verified (Fig. II.2).

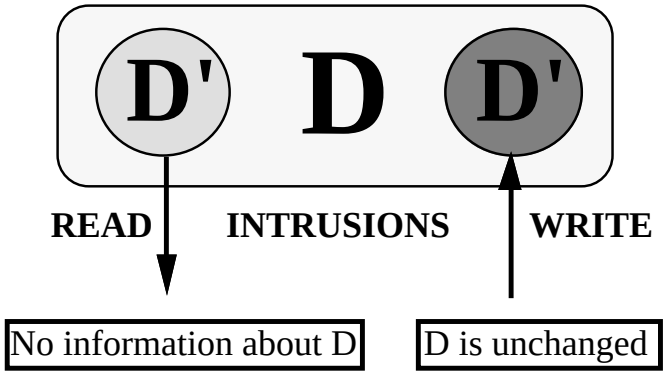


Fig. II.2. Intrusion tolerance.

The implementation of this concept is done on the security sites. The security sites are particular sites in that they together offer security services in a way such that security is always ensured in spite of intrusions on a bounded number of sites. The security services are *distributed* and *intrusion tolerant*. The implementation has the essential property of requiring no local TCB.

II.3.1 INTRUSION TOLERANCE BY DISTRIBUTION

Intrusion tolerance for data is relatively easy to implement. Data intrusion-tolerance techniques have existed for a long time. Confidentiality can be ensured by cryptographic tools like threshold scheme [SHA 79, BLAK 79]. The data is shared in "shadows", each shadow being stored on one security site. To build the data you only need a sufficient number of shadows called the threshold. If you do not have enough shadows, you cannot build the initial data. The same scheme can ensure availability and integrity.

Some data is not confidential, so it is also possible to replicate such data on each security site. Data is *shared* or *replicated* according to its confidentiality.

The most important point and the hardest one is the prevention of denial of service. In this case, it is not just data but a service that has to be protected. The server thus has to be replicated on each site in order to prevent denial of service in case of a security site unavailability. However the different sites cannot take certain decisions independently. They must agree by communicating data and local decisions. To ensure the last property, the servers must obey to two implementation principles: *replication* and *agreement* (Fig. II.3).

The distribution of the security service permits a geographic distribution of the security sites. This makes the intruder's task more difficult since even if he succeeds in accessing one site, it will be more difficult for him to access other sites if they are not in the same place. It would indeed be a pity to distribute programs and data to perform logical intrusion tolerance and to not provide geographic distribution to assist physical intrusion tolerance.

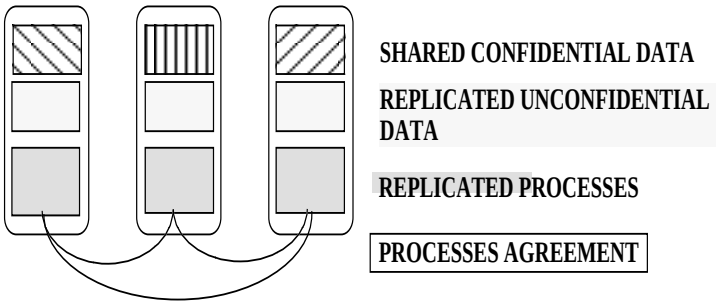


Fig. II.3. Distributed Security Services on three security sites.

II.3.2 DISTRIBUTED TCB

In the Red Book architecture, there exists on each computer a part called the Trusted Computing Base including the NTCB partition (cf. I.1.). The TIU assumes only communications security whereas the NTCB partition implements the local part of the network authorization policy. This NTCB partition must have a very high physical and logical protection. Moreover all sites have to trust one another. On each computer, accesses are mediated by the local TCB which is firstly the implementation of the **local** authorization policy. The **global** authorization policy is implemented in the set of NTCB partitions. When all computers are connected by a network and a subject wants to access a remote object, the NTCB partitions communicate with each other. In this case, a subject on a given site must trust all sites he wants to access. If one site has been penetrated by an intruder, the security of the network cannot be ensured.

In the Kerberos architecture, the most important part of the TCB is within the security server. There also exists on each server an important TCB which has to carry out authorization operations. However if an intruder succeeds in penetrating a server, he cannot access the other ones. The consequences of an intrusion are then limited. The only site you really have to trust to ensure global security is the security server site. However, in this case there exists a single point of failure.

In the intrusion tolerant approach, there is no local "Trusted" Computing Base on the security sites. Only the set of security sites is globally trusted (Fig. II.4). There is also a small local TCB on the user sites and the secured servers. The servers themselves, for instance the Archive Service, can have a distributed TCB. In this case, the only single point of failure is on the user site. This can be minimized if the user site is considered as a one-user computer when a user accesses a secured service. If this were not so, there would always exist trapdoors on the local protections between users.

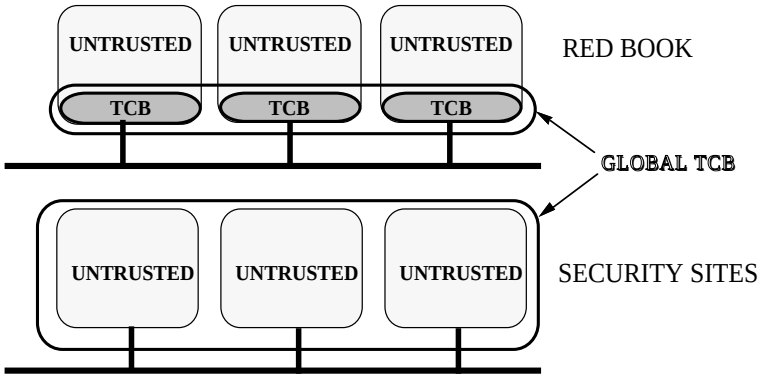


Fig. II.4. Red Book versus Security Sites TCB.

The security server is trusted, but not the different computers. The security service is considered as one global server. If an intrusion in one computer is successful in a classical architecture with a local TCB, the security of the full system is no longer verified. On the contrary, if protections of one security site in the intrusion tolerant approach are by-passed, the security of the global system is

maintained. These differences come from three different points of view about security and networks:

- In the Red Book, a network is seen as only a communication channel (low layers) between centralized systems. A subject, or an object, is located on one site and cannot be shared between several sites. In our architecture, the network is a support for distributed applications. A subject or an object can be shared on several sites.
- In the Red Book, sites are time-sharing systems with several users working on the same host. These computers are controlled by a system administrator. In Kerberos, sites are workstations under the control of one local user. In our system, whenever possible, the use of one host by several users during access to a secured service will be prohibited.
- The Red Book architecture is a support for DoD policy, a multilevel mandatory policy where confidentiality is the most important property to be ensured. All sites thus need to have an important trusted part. It is not possible to implement this policy in our system (cf. II.6). Trusted paths between all user sites would be needed.

II.4 The provided services

The different services which must be provided by the security sites are registration, authentication, authorization, sensitive data management, audit and recovery service.

II.4.1 THE REGISTRATION SERVICE

The registration service permits a user to be regis tered by the system for future access to secured services. This operation must be carried out independently on each security site to prevent a single site from using information to impersonate the user. The operation is done under control of the security administrator of each site.

II.4.2 THE IDENTIFICATION-AUTHENTICATION SERVICE

The role of this service is to verify the claimed identity of a subject. When a user, or a process acting for a user wants to access a secured service, he must first be authenticated. The authentication service verifies that the subject is really who he claims to be. To do this, both logical or physical techniques are available. These can be based on passwords, zero-knowledge authentication [FIAT 86, GUIL 88], smartcards or chip-cards. All these techniques use the same protocol principle: the subject must prove its identity to the authentication server by showing that he possesses a secret information, its authenticator.

In a distributed system with several authentication servers, each server must independently authenticate the subject (Fig. II.5). Indeed, the security sites are untrusted and one site could try to use authentication information given by the user and the data stored on the site to impersonate him in another authentication phase. The independence of the authentications on each site must be complete (different authentication objects for each site or zero-knowledge authentication). The second phase of the protocol is the agreement. The servers communicate their decision to each other and they take a global decision. If a majority of servers succeed in authenticating the subject, the subject can be considered as authenticated by all the servers.

The servers will send some specific data to the subject (session key, identificator, ...) with a time-out. If the time-out expires, the subject must repeat the authentication phase.

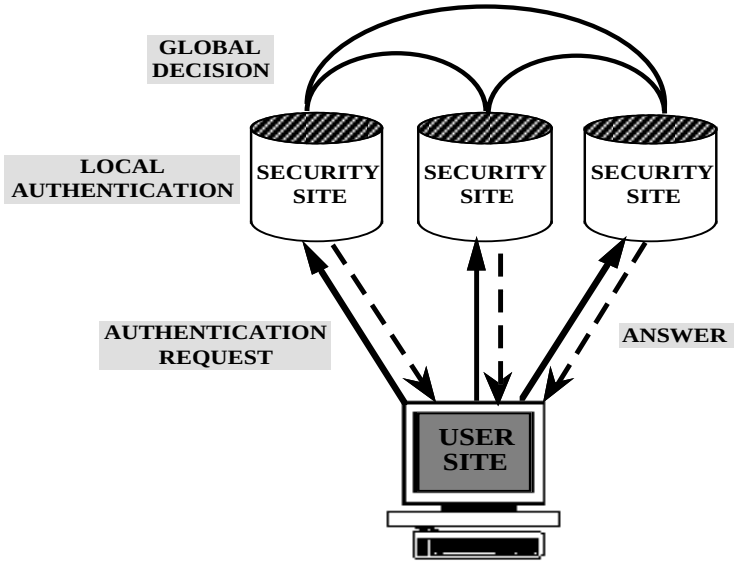


Fig.II.5. The Authentication protocol.

II.4.3 THE AUTHORIZATION SERVICE

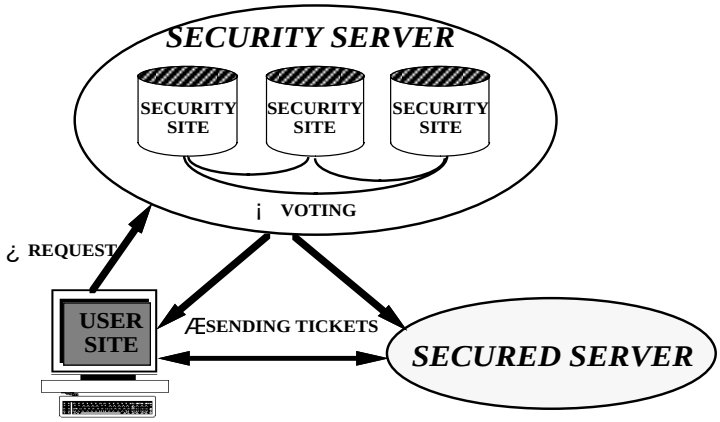
The role of this service is to check that the access to a secured service by a subject is authorized according to its access-rights. In Kerberos, this service is provided by the servers themselves because of the large size of the managed database and of the difficulty of modelling different access-rights for different services. However, these two problems can be solved.

It is possible to implement access-rights for different services using standard UNIX rights. In UNIX, all objects are viewed as files. The read, write and execute rights are thus applied to all objects of the system. The accesses to the services can be authorized using the same access-rights model. In this case, access-rights can be implemented for all kinds of services thus leading to a reduction in the cost of the access-rights database management.

The authorization service is made intrusion tolerant when it is implemented on security servers. The rights must be changed by all security administrators. One of them cannot access service if he is not authorized by others. An intruder cannot modify access rights as easily as he could do if they were located on the server. But it is obvious that if an intruder controls the service, he can do what he wants. The only solution is to build an intrusion tolerant service such as the archive service, but this is not always possible (e.g. consider a printer service).

The different phases of authorization are (Fig. II.6):

- The subject asks the security servers for permissions to access a secured service by sending its identificator (received in the authentication phase) (1).
- The access-rights stored on the security sites enable the latter to verify that the subject is authorized to access the requested service.
- The security sites vote to decide if the access is authorized using the same protocol as that defined for authentication (2).
- If the sites agree to permit access, they send a ticket to the subject and another ticket to the secured server (3).
- With the ticket, the subject can open a session with the server.



II.6. Authorization protocol.

A subject may also want to access secured data stored on security sites, like access-rights or authentication keys. In this case, the access control protocol is the same as the one described above for the phases 1 and 2 and then, if and, when the security sites accept the access, they perform it on each site and send the information or the affirmative response to the user site.

II.4.4 THE SENSITIVE DATA MANAGEMENT SERVICE

The role of this service is to store, manage and retrieve the sensitive information on the security servers so that their protection verifies the hypothesis made in §II.2. This information consists of short data items needed to achieve security services.

The data management service must enforce the three main security properties (confidentiality, integrity and availability). The integrity property is provided by modification detection mechanism such as cryptographic signatures. According to the sensitivity of the security data, it can be important to preserve both the confidentiality and availability of this data, or only the availability. For this, two storage techniques can be applied: replication (for availability) or threshold schemes (for confidentiality and availability). In function of this, the security administrators (data for a service access) or a subject (data for authentication) will store data with one of the two algorithms (Fig. II.7).

If a data item is replicated on N security sites, it is assumed:

- with respect to availability, that N-1 replicates can be lost (modified or destroyed),
- with respect to confidentiality, that one replicate is sufficient to observe data.

If one data is shared on N security sites (in this case we speak of shadows) using a threshold T, it is assumed:

- with respect to availability, that N-T shadows can be lost,
- with respect to confidentiality, that T shadows are necessary and sufficient to observe data (less than T shadows gives **no information** about the data).

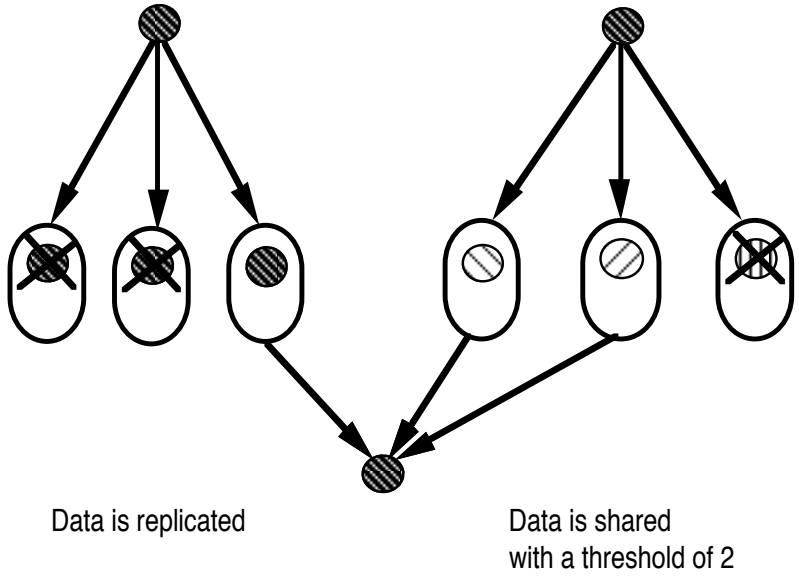


Fig.II.7. Replication and Threshold Scheme.

II.4.5 THE AUDIT SERVICE

The role of this service is to record all information related to security. Such information is sent by the services defined above. There exists two kinds of information, authorized operations performed by authorized users (registration, access, rights change, ...) and attempted or successful intrusions or misuses. It is not the role of the services to determine what is an intrusion or misuse by an authorized user. This is the function of an audit trail analysis.

The audit information is sent not only by security sites but also by secured servers and user sites. For the former, it will be access-requests, and for the latter it will be, for instance, information about correct or incorrect shared data sent by security sites (bad shadows received from certain security sites).

The audit trails are stored on each security site. The information received on one site is not sufficient to compromise the security of the system.

The analysis of this audit information will be done off-line by security administrators. As one intrusion is masked, it is not necessary to detect intrusion on-line.

II.4.6 THE RECOVERY SERVICE

It acts as an error recovery mechanism to correct certain modified data (e.g. shadows of the threshold scheme). Other recovery functions can be performed manually by security administrators using audit trails.

II.5. Implementation of an authorization policy

The security policy of an enterprise strongly influences the design of the system intended to support applications for this enterprise. In our case, the architecture was defined without a particular security policy in mind. All the policies are therefore not necessarily compatible with the security architecture and the hypotheses. In our case, mandatory policies cannot be implemented for the same reasons as Kerberos (cf. I.2): security is not ensured within a user site and between user sites. We have thus chosen a discretionary authorization policy.

II.5.1 IMPLEMENTATION OF DISCRETIONARY POLICY

Discretionary policy is an authorization policy where some authorized subjects have right-modify right on an object. **Right-modify right** on an object is the right to modify the access-rights

on the object. Discretionary policy means that subjects which have right-modify rights must evaluate by themselves the trustworthiness of other subjects. Discretionary policy is very close to our distributed security ideas. In our system, if a subject can read information he can disclose this information to whoever he wants. Only some paths are trusted. Subjects can be identified as all processes acting on user sites. If this site can reassemble an archive, it can forward it to other user sites without any control. The unauthorized disclosure of information is only limited by the trust a subject puts in another subject. If integrity is ensured by security sites, confidentiality is in part under the responsibility of users.

The propagation of other rights (execute, write, ...) can be limited by using hierarchical authorization (right-modify right is given to the subjects at the top of hierarchy) or centralized authorization (right-modify right is given to one security administrator) [NCSC 87b]. The centralized scheme can only be used in a rigid system where most subjects or objects are created during an initialization phase. Changes have to be infrequent. It will certainly not be the case in Delta-4 applications. The hierarchical scheme is more flexible and is simple to implement.

II.5.1.1 Access lists and access tokens

The implementation of access-rights can be made in two ways, either using **access-control lists** or **capabilities** [FABR 74, SALT 75].

In our system, access control is done by security sites and then by archive sites. Security sites must authenticate subjects and authorize them to access archives. When they grant access, two solutions are possible. Firstly, security sites could be the mandatory gateway between a user site and the archive sites during a session (Fig. II.8). In this case performance and security decrease since:

- security sites become a fragment-transfer bottleneck,
- security sites know information that they should not have to know (fragment, fragment name).

The second solution is that security sites supply user sites and archive sites with elements that enable the creation of a direct session between them (Fig. II.9). These elements form a ticket with cryptographic keys and access tokens (enciphered fragment name and access-right). The access tokens will be used as capabilities and will be sent to archive sites so as to authorize users to access fragments.

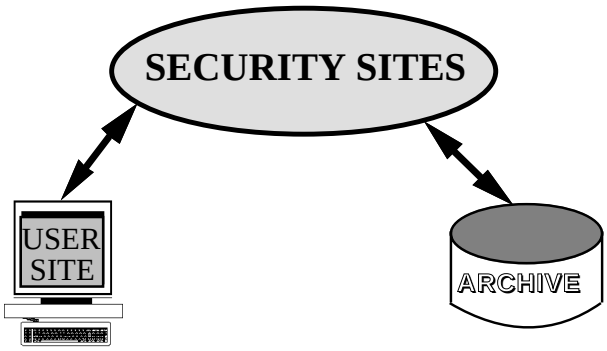


Fig.II.8. Security sites are gateway.

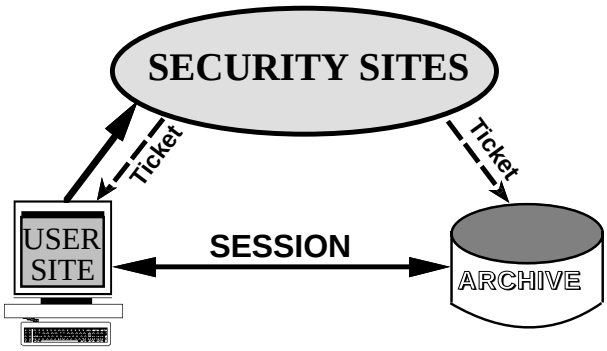


Fig.II.9. Opening session with tickets.

The security sites have to verify if a subject is authorized to access an archive. The fastest way to check authorization is the use of an access-control list which can be replicated on all sites.

II.5.1.2 Access rights

Since access-control lists are to be used, it appears interesting to use the existing UNIX file protection scheme. Under this scheme, the file authorization list is made up as follows:

- Access-right types are read, write and execute.
- The rights are defined for the owner, the members of the file group and the other users.

A fourth field called access-list is added. This ac cess-list contains a set of groups or users with added or restricted rights (Fig. II.10). These groups or users have more or less rights than ones defined by precedent fields. For instance, if the object group right is execute only; in order to give write right to a particular member of the group, an entry for this subject in the access-list with added write rights is created. This is the same thing for groups. Rights for some subjects can also be reduced. The advantages of this dynamic implementation is to be close to complete access-control lists granularity without excessive data overhead.

FIXED LIST	
OWNER	**RIGHTS**
GROUP	**RIGHTS**
OTHERS	**RIGHTS**
DYNAMIC ACCESS-LIST	
USER OR GROUP	**EXTENDED RIGHTS**
USER OR GROUP	**RESTRICTED RIGHTS**

Fig. II.10. Archive access-list.

II.5.2 DISCRETIONARY MULTI-CATEGORIES SOLUTION

Groups can be defined in a such a way that security categories can be built. Only the security administrators will be authorized to create and modify groups.

We have seen that it is impossible to implement a mandatory policy, but it is possible to make categories such as Bell-LaPadula categories. These categories represent interest classes. In military or commercial domains, it is easy to partition objects into categories. A subject can access to more than one category whereas an object can only belongs to one category (this is different from the Bell-LaPadula model). In our system, categories are built with groups. These groups should be defined by security administrators when they implement procedural part of the security policy. A group can also be added or subtracted when the system runs.

To implement the categories, UNIX group concepts are used with some differences. Firstly, a subject can belong to several groups. This is already implemented in UNIX System V, however its use is not really easy. The file access-control does not use the groups the user belongs to, it uses the "current shell group" only. It is first defined in a login configuration (passwords file) and when a user wants to have other group permissions, he has to change his "current shell group" by a command (newgrp).

Our idea is to verify group members during each access try. If a user belongs to several groups, he must have access-rights of these groups all the time. These rights have to be defined. It is the object and subject creation rules which do this.

- *GROUPS.*

Groups are subject sets and each group specifies one category. To partition categories, one category cannot include an object which belongs to another group, but a subject can belong to several groups (Fig. II.11). A subject has all the rights of all the groups to which it belongs.

When the groups are specified by security administrators, and subjects are distributed over them, it is necessary to write the object creation rules which give rights to subjects.

The rules are:

- when a new object is created, the object owner is the object creator and has U-rights, the object group is the group of the owner and group members have G-rights, the other subjects have O-rights (often no rights).
- if a subject belongs to several groups, it may decide to which group the object is to belong.

These rules are modular enough to build different policies with different constraints. Security administrators have to choose the various U, G and O rights.

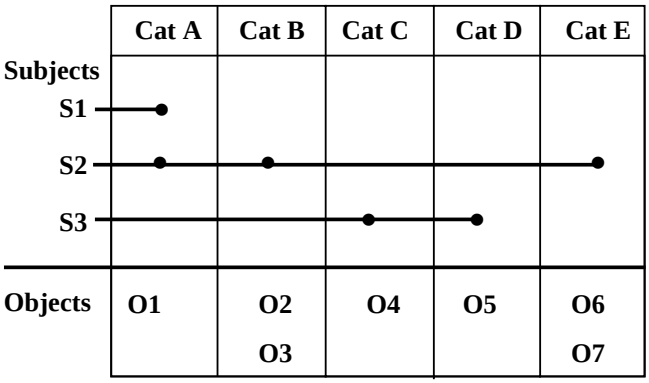


Fig. II.11. Example of groups/categories.

In this example:

- S1 belongs to the category A and has G(A) rights on A-objects and O-rights on B, C, D and E-objects.
- S2 belongs to the categories A, B and E, and has G(A), G(B) and G(E) rights on the A, B and E-objects and O-rights on C and D-objects.
- S3 belongs to the categories C and D and has G(C) and G(D) rights on the C and D-objects and O-rights on the A,B and E-objects.

- *NO DISCRETIONARY CONTROL BY OWNER.*

The discretionary control on the objects is not carried out by the owner. The owner has particular rights but does not have the right-modify right on the objects. A user must not be able to give rights on an object in a given category to users in other categories.

The owner of the object will be the creator of the object, but the name of the owner cannot be changed by any user.

- *CATEGORY MANAGERS.*

There exists a super user, called category man ager, for each category. He is a member of the corresponding group. This user has right-modify right on all objects which belong to the category. He can change access-rights of all these objects. He does not have right-modify right on objects of other categories. A category manager cannot change the owner or the group of an object. This can only be done by security administrators. This allows separation of duties. Users, category managers and security administrators all have different roles.

III. CONCLUSION

In this paper, a new approach for security in an open distributed system has been described. It is based on a client-server model. Several security services are provided in order to secure the access to remote servers. These services are intrusion tolerant and reside on several security sites.

The intrusion tolerant technique is based on fragmentation-scattering to ensure confidentiality and integrity of data and availability of services. The security site architecture uses replication of data and processes, threshold schemes for confidential data and agreement protocols between processes. The different services built on this architecture are registration, identification, authentication, authorization, management of sensitive data, audit and recovery. With these services, an access-control scheme based on a discretionary multilevel authorization policy will be implemented.

ACKNOWLEDGEMENTS

The authors wish to thank Jean-Claude Laprie and David Powell for their contributions to the main concepts presented in this paper. The Delta-4 project is supported by the European ESPRIT program. This work is also partially supported by the Conseil Régional de Midi-Pyrénées and by the Programme de Recherche Concertée C^3.

REFERENCES

[BELL 74] BELL D. E. and LAPADULA L. J., "Secure Computer Systems: Mathematical Foundations and Model", M74-244, MITRE Co., October 1974.

[BIBA 77] BIBA K. J., "Integrity considerations for Secure computer Systems", ESD-TR 76-372, MITRE Co., April 1977, 64 p.

[BLAK 79] BLAKLEY G. R., "Safeguarding Cryptographic Keys", Proc. NCC, Vol. 48, AFIPS Press, Montvale N. J., 1979, pp. 313-317.

[CLARK 87] CLARK D.D. and WILSON D.R., "A Comparison of Commercial and Military Security Policies", Proc. of IEEE Symp. on Security and Privacy, April 1987, pp.184-194.

[DENN 86] DENNING D. E., "An Intrusion-Detection Model", Proc. of IEEE Symp. on Security and Privacy, April 1986, pp. 118-132.

[DOD 85] DoD, "Department of Defense Trusted Computer System Evaluation Criteria", DOD 5200.28-STD, December 1985.

[FABR 74] FABRY R. S., "Capability-Based Addressing", Communications of ACM, Vol. 17, N° 7, July 1974, pp. 403-412.

[FIAT 86] FIAT A., SHAMIR A., "How to Prove Yourself: Practical Solutions of Identification and Signature Problems", Advances in Cryptology - CRYPTO'86, Santa Barbara, August 1986, Lecture Notes in computer Science, Vol. 263, ISBN 0-387-18047-8, pp.186-194.

[FRAG 85] FRAGA J., POWELL D., "A Fault and Intrusion-Tolerant file System", Proc. 3rd Int. Cong. on Comp. Security (IFIP/SEC'85), Dublin, Ireland, August 1985, ISBN 0-444-87801-7, pp. 203-218.

[FRAY 86] FRAY J.M., DESWARTE Y., POWELL D., "Intrusion-Tolerance using Fine-Grain Fragmentation-Scattering", Proc. on the 1986 IEEE Symp. on Security and Privacy, Oakland, April 1986, pp. 194-201.

[GUIL 88] GUILLOU L.C., QUISQUATER J.J., "A Practical Zero-knowledge Protocol Fitted to Security Microprocessor Minimizing both Transmission and Memory", Advances in Cryptology - Eurocrypt 88, Davos, Switzerland, May 88, Lecture Notes in Computer Science, Vol. 330, Springer Verlag, ISBN 0-387-50251-3, pp. 123-128.

[HARR 76] HARRISON M. A., RUZZO W. L. and ULLMAN J. D., "Protection in Operating Systems", Comm. of ACM, Vol. 19, n° 8, August 1976, pp.461-471.

[ISO 7498-2] I.S.O., International Standard 7498-2: Information processing systems - OSI Reference Model - Part 2: Security Architecture, Tech. Rept. n°2890, ISO/IEC/JTC1/SC21, July 1988.

[LAP 89] LAPRIE J.C., "Dependability: A Unifying Concept for Reliable Computing and Fault Tolerance", in Dependability of Resilient Computing Systems, Blackwell Scientific Publications, T. Anderson editor, 1989, pp. 1-28.

[MILL 87] MILLER S.P., NEUMAN B.C., SCHILLER J.I. and SALTZER J.H., "Kerberos Authentication and authorization System", MIT Proj. Athena Technical Plan, Sect. E.2.1, December 1987.

[NCSC 87a] NCSC, "Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria", NCSC-TG-005, July 1987.

[NCSC 87b] NCSC, "A Guide to Understanding Discretionary Access Control in Trusted Systems", September 1987.

[RAN 88] RANEA P.G., DESWARTE Y., FRAY J.M., POWELL D., "The Security Approach in Delta-4", EUTECO'88, Vienna, Austria, April 1988, ISBN 0-444-70428-0, pp. 455-466.

[SALT 75] SALTZER J. H. and SCHROEDER M. D., "The Protection of Information in Computer Systems", Proc IEEE, Vol. 63, September 1975, pp. 1278-1308.

[SHA 79] SHAMIR A., "How to Share a Secret", Comm. of ACM, Vol. 22, n°11, November 1979, pp. 612-613.

[STEIN 88] STEINER J. G., NEUMAN C. and SCHILLER J.I., "Kerberos: An Authentication Service for Open Network Systems", USENIX Winter Conf., Dallas, February 1988.