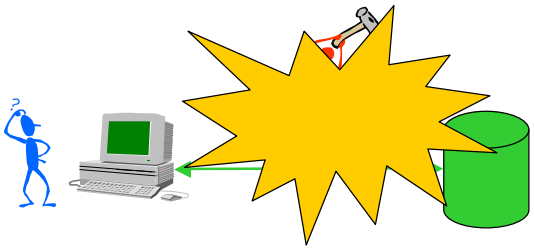
Authorization Schemes & Intrusion Tolerance for Internet Applications

Yves Deswarte
LAAS-CNRS

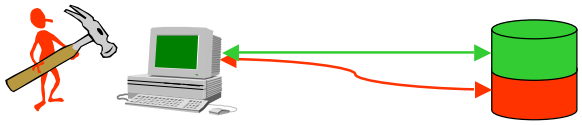
Who are the intruders?

1: "Outsider"



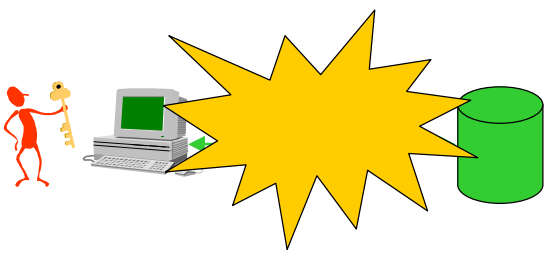
☺ Authentication
☺ Authorization

2: User



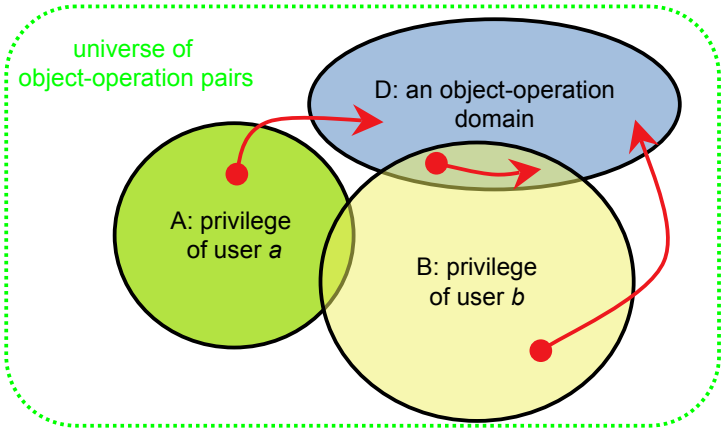
☹ Authentication
☺ Authorization

3: Privileged User



☹ Authentication
☹ Authorization

Outsiders or Insiders: Privilege

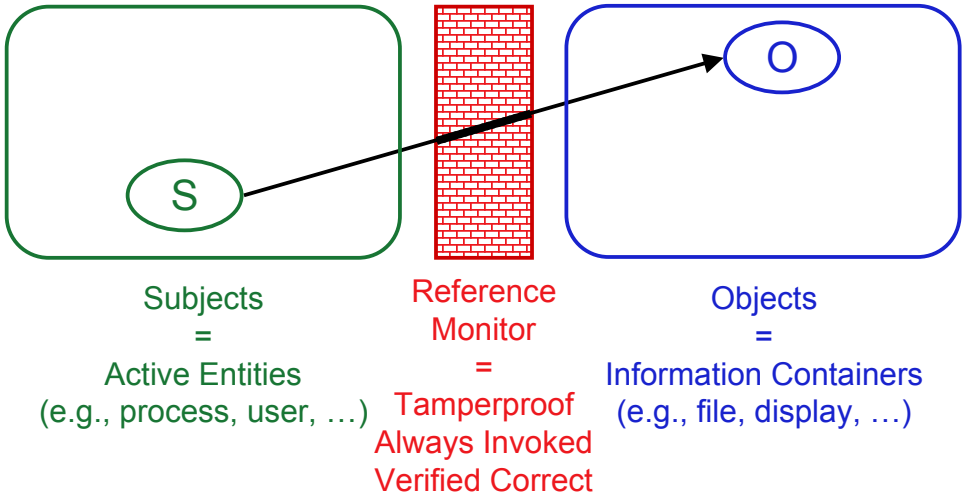


- Theft of privilege: unauthorized increase in privilege
- Abuse of privilege: improper use of authorized operations
- Outsider: current privilege does not intersect considered domain
- Insider: current privilege intersects considered domain

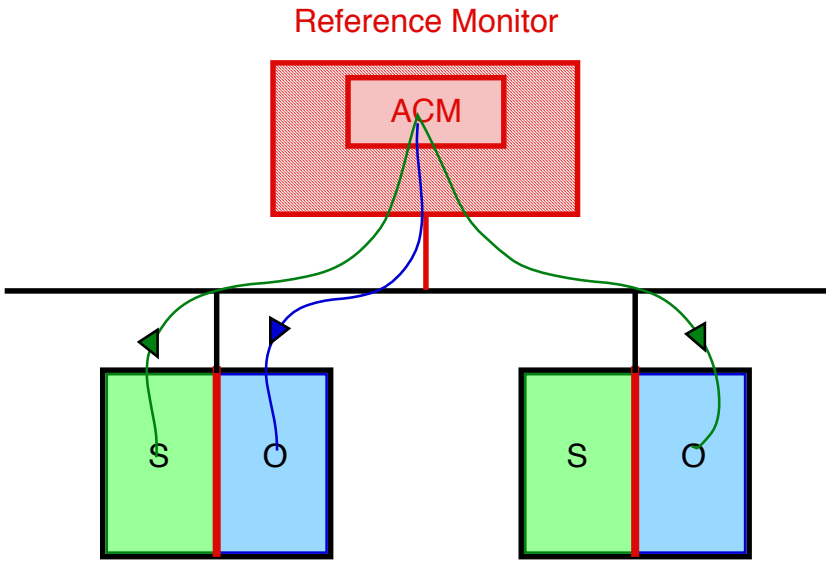
Authorization

- Contributes to protection:
  - Error detection/confinement
  - Intrusion prevention/confinement
- For Internet applications:
  - More flexible than "client-server" paradigm
  - Contributes to privacy: personal information is disclosed only on a "need-to-know" basis

Authorization: reference monitor

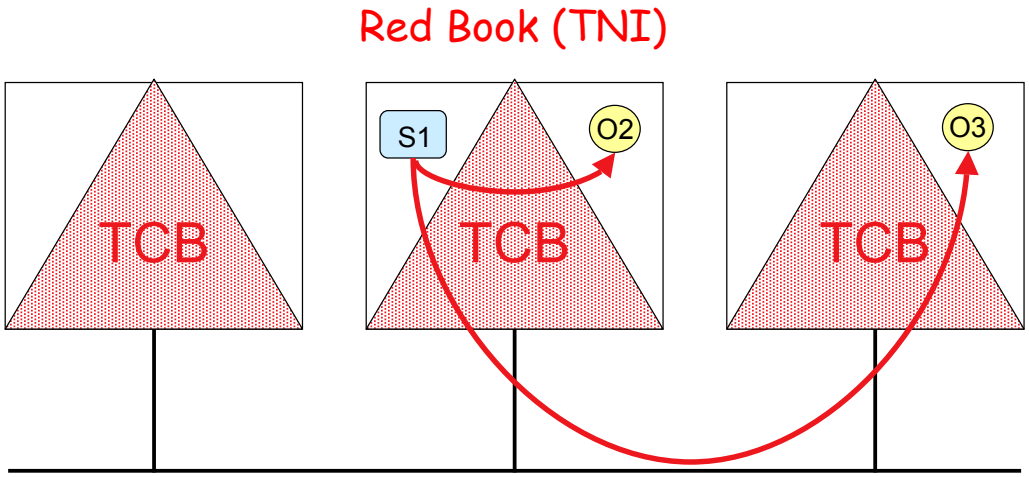


Distributed Authorization ? (1)



☺ small trusted area, easy administration
☹ bottleneck, single-point-of-failure

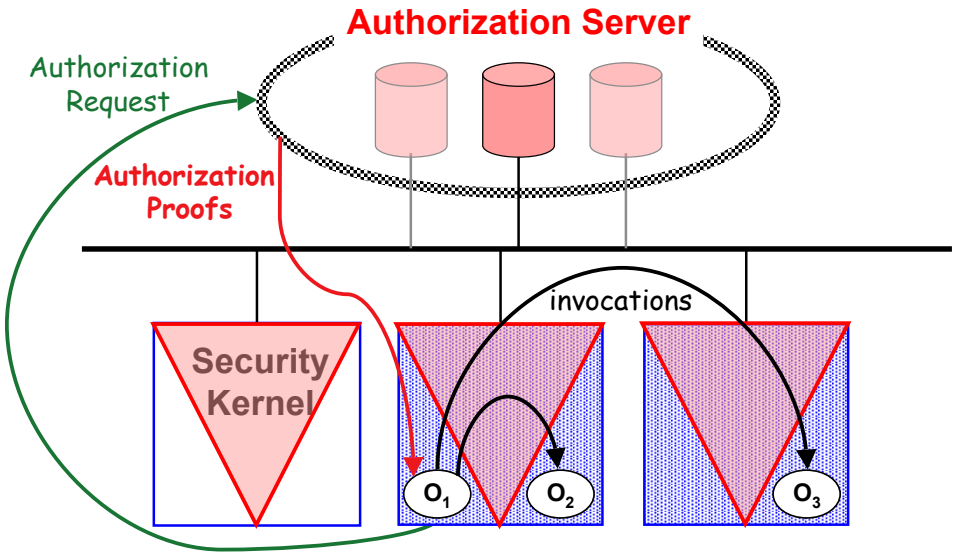
Distributed Authorisation ? (2)



☺ No bottleneck, no single-point-of-accidental-failure
☹ Mutual trust between TCBs, consistency?

Authorization Scheme for DOOS

[Nicomette & Deswarte, 1997]



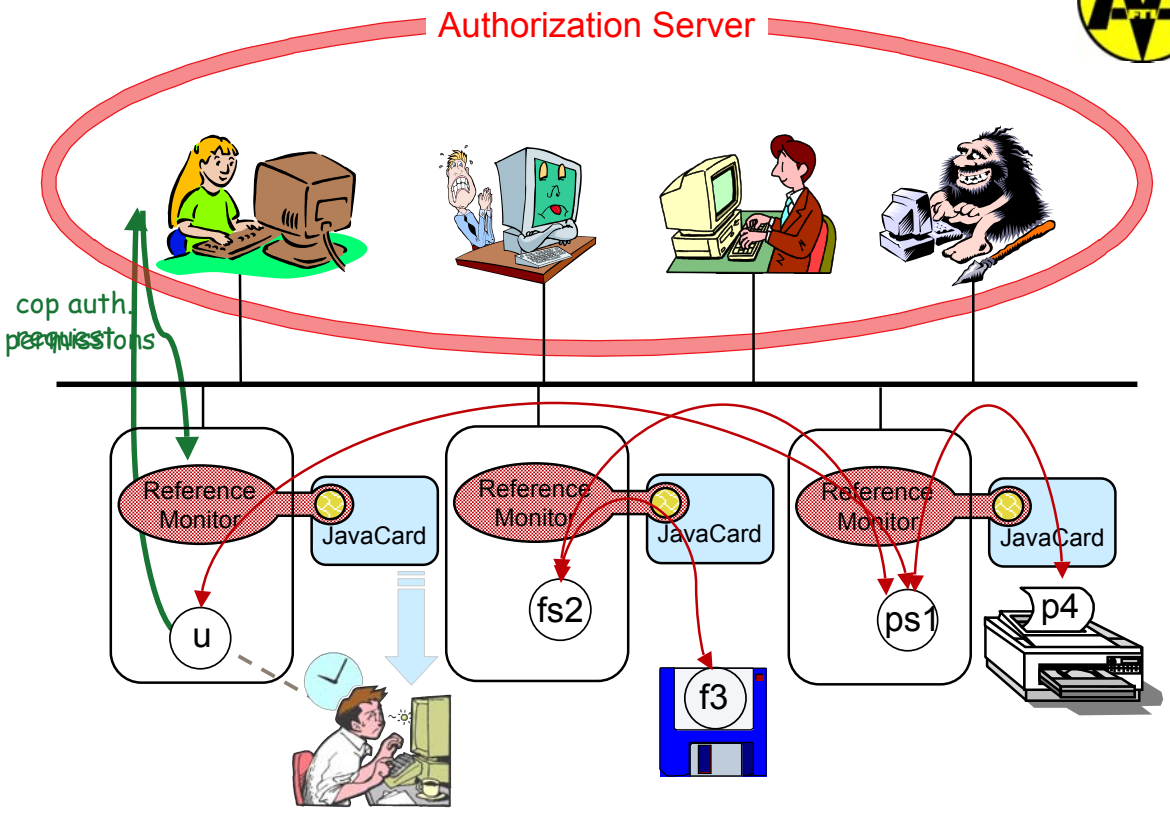
MAFTIA Authorization Server

Makes use of MAFTIA middleware:

- Non-confidential information is replicated (atomic multicast)
- Confidential information is shared securely (threshold crypto)
- Global consensus is achieved (majority voting / Byzantine agreement)
- Authorization proofs are distributed to local reference monitors (threshold signatures)

Local protection

- Internet applications: heterogeneous platforms => no modification of user workstations or even servers
- => no trusted security kernel, but JVM
- Local Reference Monitor =
  - A local dispatcher (not trusted)
  - A JavaCard -> security kernel

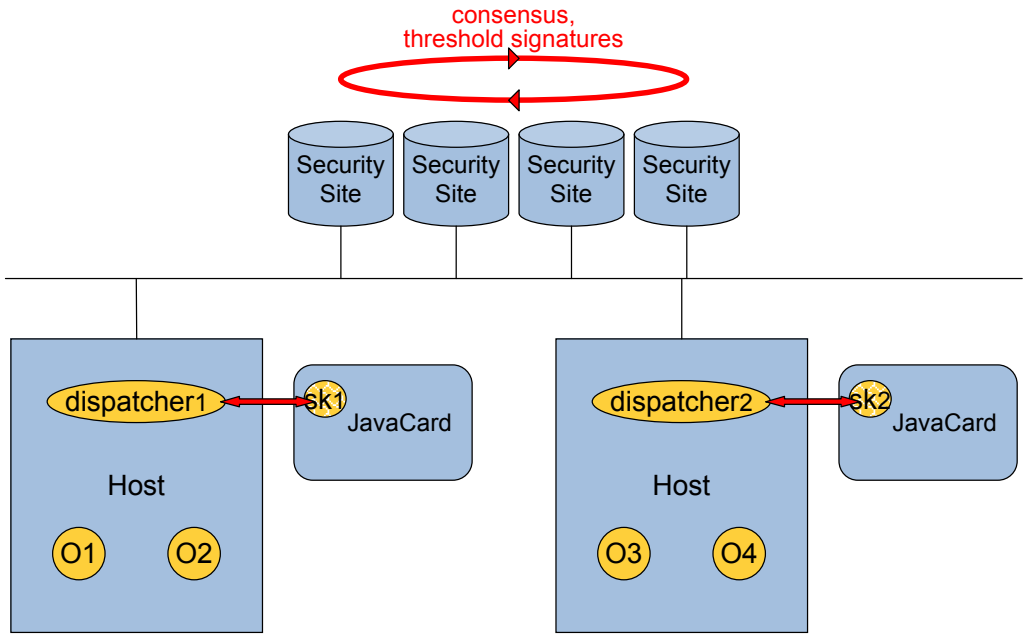


Security properties

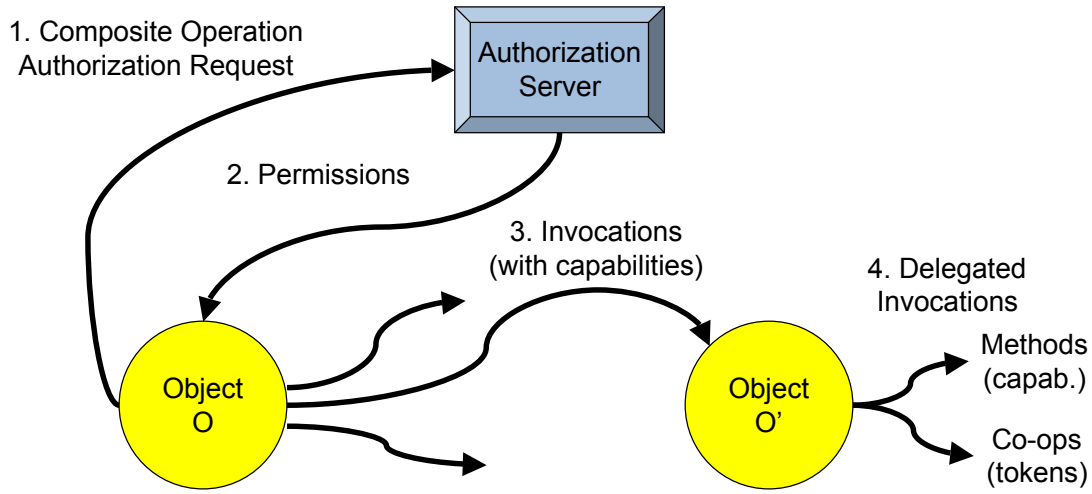
- Authorization server:
  - AS1: The AS generates only valid authorization proofs
  - AS2: It is not possible to prevent AS from generating valid authorization proofs
- Local reference monitors
  - RM1: Only valid operations will be executed on a non-faulty host
  - RM2: It is not possible to prevent non-faulty hosts from executing valid operations

- Assumption1: no network denial-of-service
- Assumption2: Java Card tamperproof

Architecture



Permissions



Permissions(O) = <{Perm(O,$O'.m$)*;Perm(O,cop)*}>$_{SKas}$
Perm(O, $O'.m$) = {O;$O'.m$($parC$);**cap**(O;$O'.m$($parC$))[;**vouch**($O'.m$)]}
Perm(O,cop) = {O;cop($parC$);**token**(O;cop($parC$))}
cap(O;$O'.m$($parC$)) = <{O;$O'.m$;$parC$;$nonce$}>$_{SKas,PKhost(O')}$
vouch($O'.m$) = <{Perm(O',$O''.m'$)*;Perm(O',cop)*}>$_{SKas}$
token(O;cop($parC$)) = <{O;cop;$parC$;$nonce$}>$_{SKas,PKas}$

Cop Authorization Checks

Symbolic rights: corresponding to the authorization for an object to execute composite operations

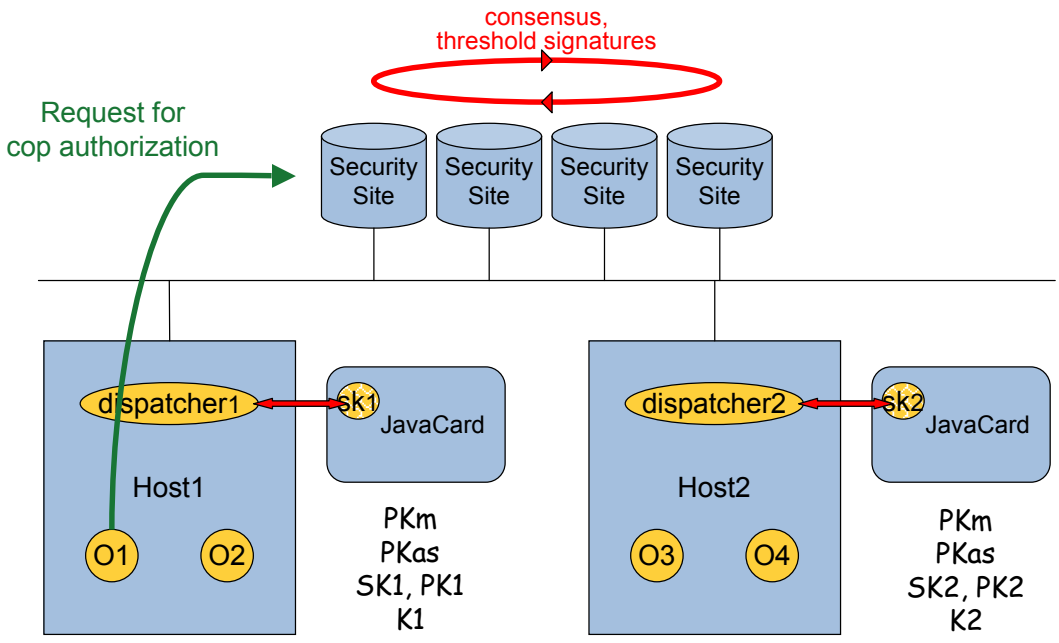
(a simple method execution is a particular case of cop)

	ps1	fs2	f 3	p 4
u			PF(this,PRINTER)	PF(FILE,this)
ps1				print
fs2				

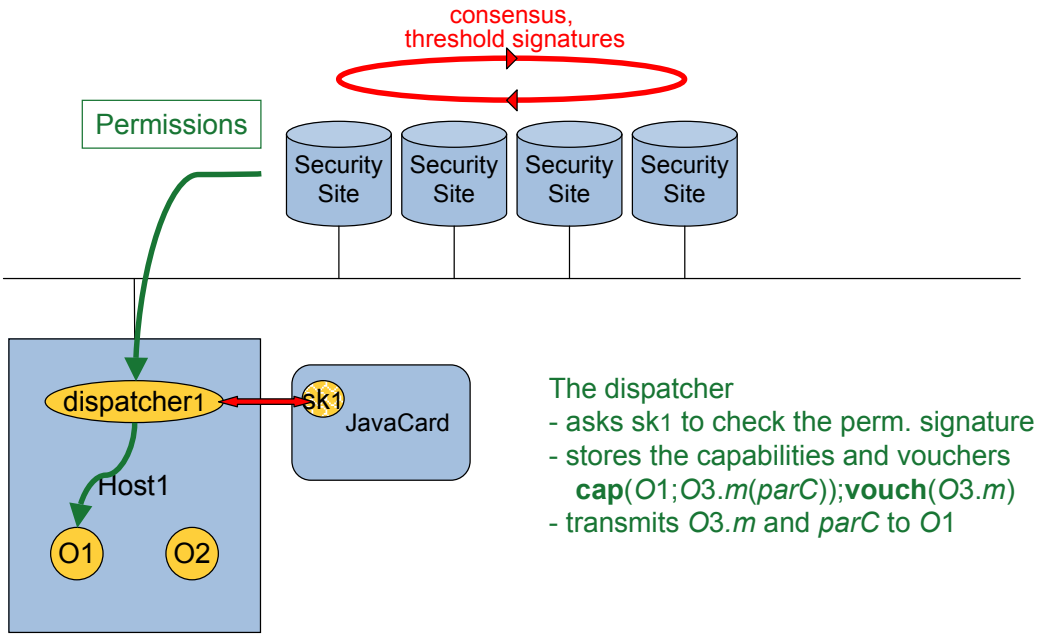
Symbolic right rules: to check authorization for composite operations

Permission creation rules: to generate permissions (capabilities and *vouchers*, tokens) to enable all methods executions

Architecture

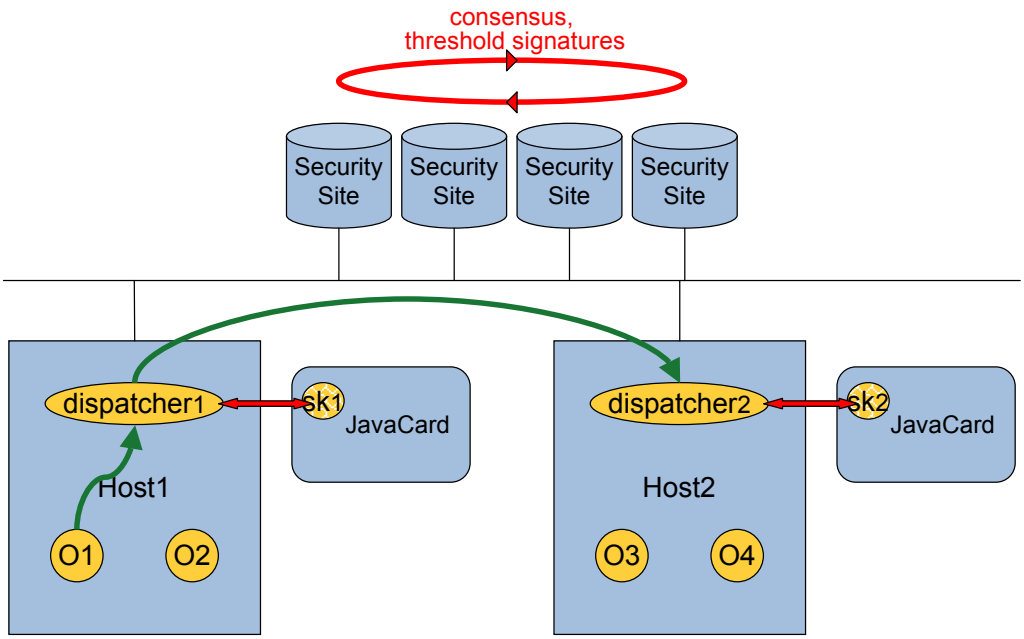


Architecture



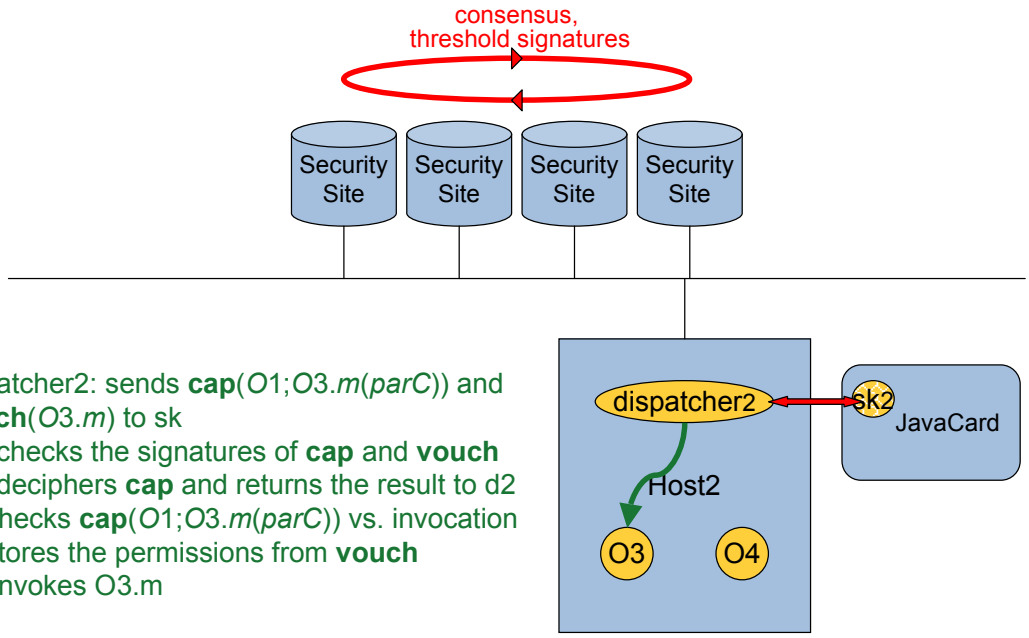
The dispatcher
- asks sk1 to check the perm. signature
- stores the capabilities and vouchers
  cap(*O*1;*O*3.*m*(*parC*));**vouch**(*O*3.*m*)
- transmits *O*3.*m* and *parC* to *O*1

Architecture



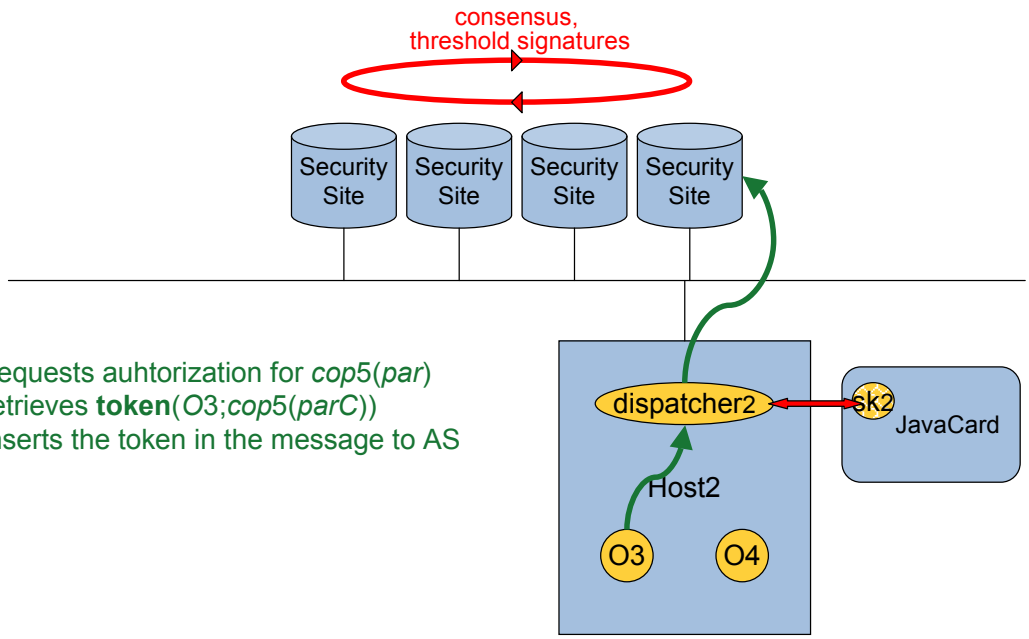
*O*1 invokes *O*3.*m* with {parameters}
dispatcher1 retrieves {**cap**(*O*1;*O*3.*m*(*parC*));**vouch**(*O*3.*m*)}, inserts it in the message to disp.2

Architecture



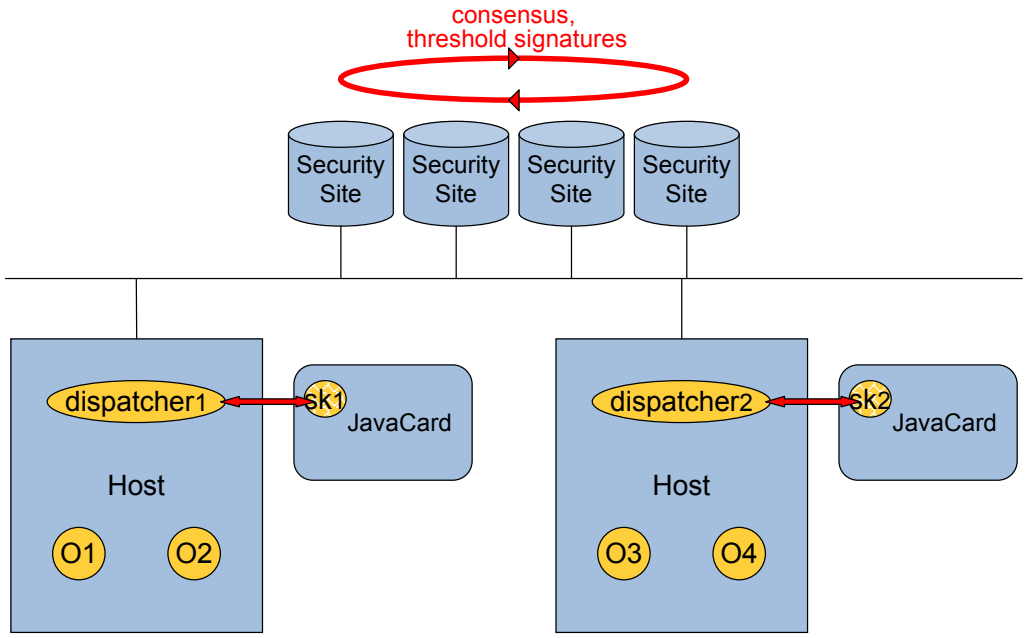
- dispatcher2: sends **cap**(*O*1;*O*3.*m*(*parC*)) and **vouch**(*O*3.*m*) to sk
- sk2 checks the signatures of **cap** and **vouch**
- sk2 deciphers **cap** and returns the result to d2
- d2 checks **cap**(*O*1;*O*3.*m*(*parC*)) vs. invocation
- d2 stores the permissions from **vouch**
- d2 invokes O3.m

Architecture

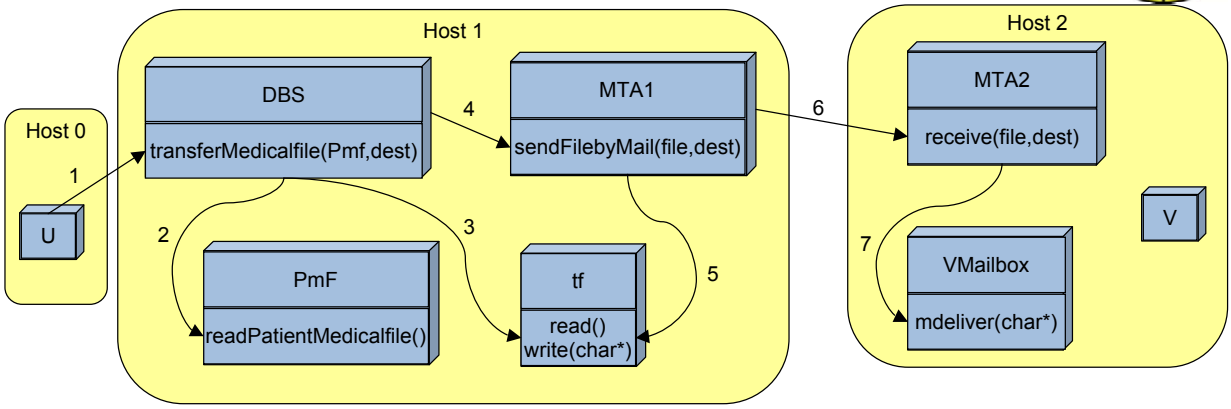


- *O*3 requests auhtorization for *cop*5(*par*)
- d2 retrieves **token**(*O*3;*cop*5(*parC*))
- d2 inserts the token in the message to AS

Architecture



Example:

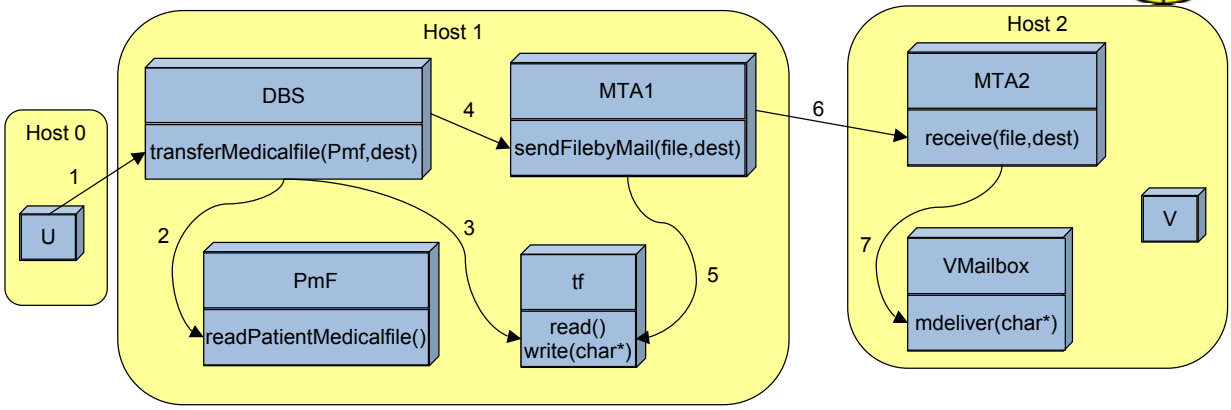


$$U \rightarrow AS : \text{Request}\left(SendPatientMedicalFile\left(Pmf_1, V\right)\right)$$

	…	HCP Role	{Pmf(U)}
U		$SPmf\left(\{Pmf(U)\}, this\right)$	$read, write(*), SPmf\left(this, HCPRole\right)$
DBS	…		
…			

$$AS \rightarrow D_0 : \left\{ \begin{array}{c} U; DBS.transferPatientMedicalfile\left(Pmf_1, V\right); \textbf{Cap}\left(U; DBS.transferPatientMedicalfile\left(Pmf_1, V\right)\right); \\ \textbf{Vouch}\left(DBS.transferPatientMedicalfile\right) \end{array} \right\}$$

Example:

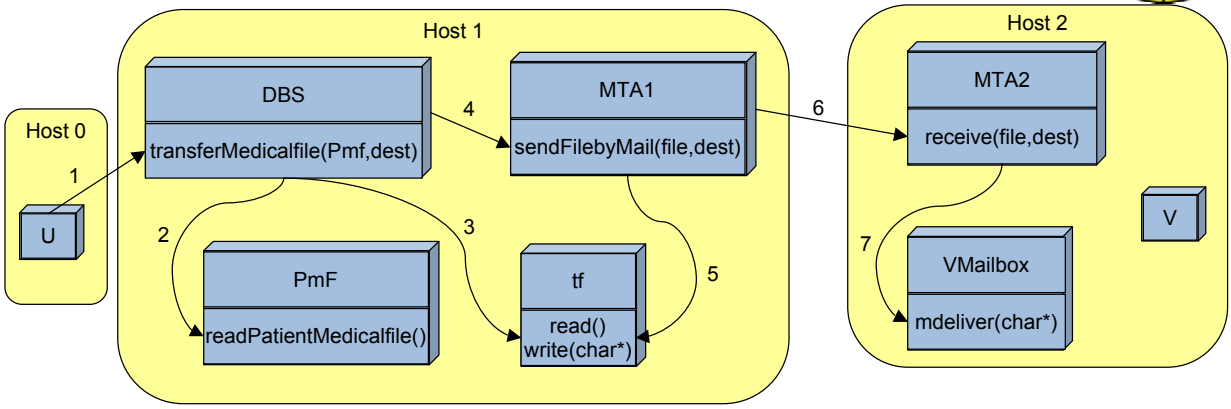


$$U \overset{1}{\mapsto} DBS : \left\{ \begin{array}{c} \mathbf{RMI}\left(DBS.transferPatientMedicalfile(Pmf_1, V) \right); \\ \mathbf{Cap}\left(U; DBS.transferPatientMedicalfile(Pmf_1, V) \right); \mathbf{Vouch}\left(DBS.transferPatientMedicalfile \right) \end{array} \right\}$$

$$D_1 \rightarrow JC_1 : \left\{ \mathbf{Cap}\left(U; DBS.transferPatientMedicalfile(Pmf_1, V) \right); \mathbf{Vouch}\left(DBS.transferPatientMedicalfile \right) \right\}$$

$$\mathbf{Vouch}(\text{DBS...}) = \left\{ \begin{array}{c} DBS; Pmf1.readPatientMedicalfile; \mathbf{Cap}\left(DBS; Pmf1.readPatientMedicalfile \right); \\ DBS; MTA1.sendFilebyMail(*V); \mathbf{Cap}\left(DBS; MTA1.sendFilebyMail(*V) \right); \\ \mathbf{Vouch}\left(MTA1.sendFilebyMail \right) \end{array} \right\}$$

Example:

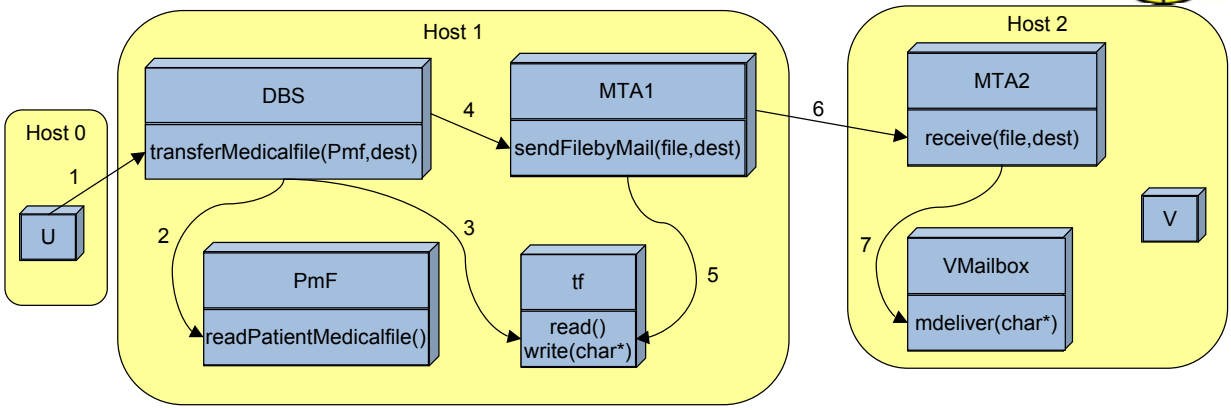


$$DBS \overset{2}{\mapsto} Pmf_1 : \left\{ \mathbf{RMI}\left(Pmf_1.readPatientMedicalfile(\ ) \right); \mathbf{Cap}\left(DBS; Pmf_1.readPatientMedicalfile \right) \right\}$$

$$DBS \overset{3}{\mapsto} tf : \left\{ \mathbf{RMI}\left(tf.write(\langle\langle \text{contentof } Pmf_1 \rangle\rangle) \right) \right\}$$

$$DBS \overset{4}{\mapsto} MTA1 : \left\{ \begin{array}{c} \mathbf{RMI}\left(DBS.sendFilebyMail(tf, V) \right); \\ \mathbf{Cap}\left(DBS; MTA1.sendFilebyMail(*V) \right); \mathbf{Vouch}\left(MTA1.sendFilebyMail \right) \end{array} \right\}$$

Example:

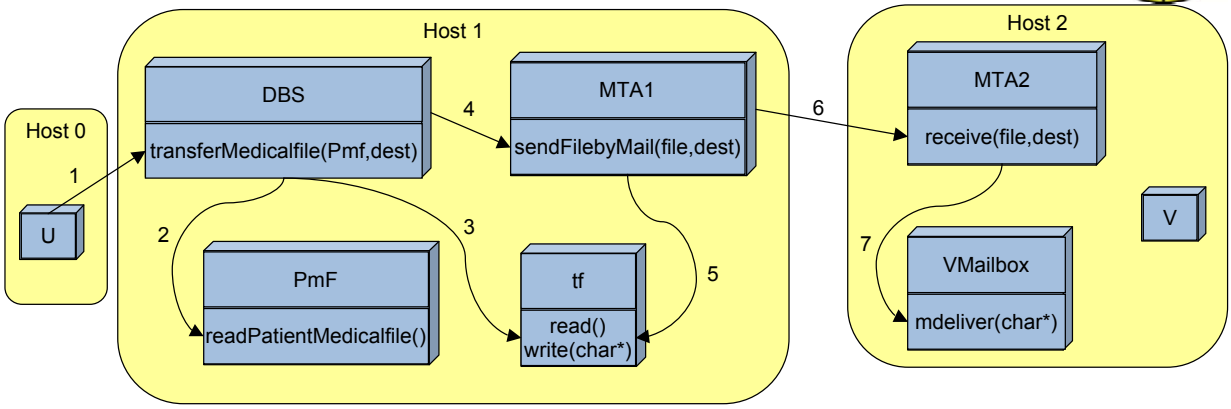


$$MTA1 \rightarrow AS: \text{Request}\left(DeliverFilebyMail(*,V)\right); \textbf{Token}(MTA1; DeliverFilebyMail(*,V))$$

$$AS \rightarrow D_1: \left\{MTA1; MTA2.receive(*V); \textbf{Cap}\left(MTA1; MTA2.receive(*V)\right); \textbf{Vouch}\left(MTA2.receive\right)\right\}$$

$$MTA1 \overset{6}{\mapsto} MTA2: \left\{\begin{array}{c} \textbf{RMI}\left(MTA2.receive\left(\langle \text{content of } tf \rangle, V\right)\right); \\ \textbf{Cap}\left(MTA1; MTA2.receive(*V)\right); \textbf{Vouch}\left(MTA2.receive\right) \end{array}\right\}$$

Example:



$$D_2 \rightarrow JC_2: \left\{\textbf{Cap}\left(MTA1; MTA2.receive(*V)\right); \textbf{Vouch}\left(MTA2.receive\right)\right\}$$

$$MTA2 \overset{7}{\mapsto} VMailbox: \left\{\textbf{RMI}\left(VMailBox.mdeliver\left(\langle \text{content of } tf \rangle\right)\right); \textbf{Cap}\left(MTA2; VMailbox.mdeliver(*)\right)\right\}$$

http://www.research.ec.org/maftia/