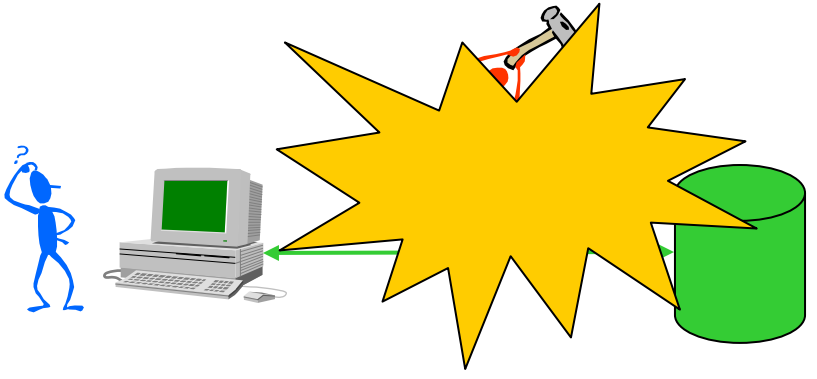
MAFTIA's Authorization: WP5 Overview

Presented by Yves Deswarte, LAAS-CNRS

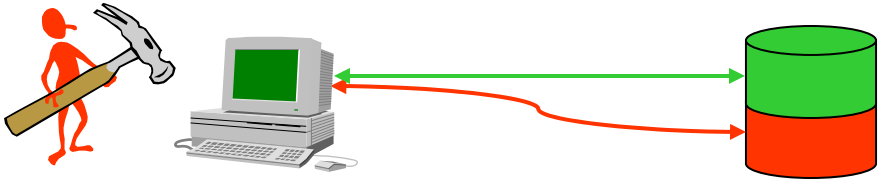
Who are the intruders?

1: “Outsider”



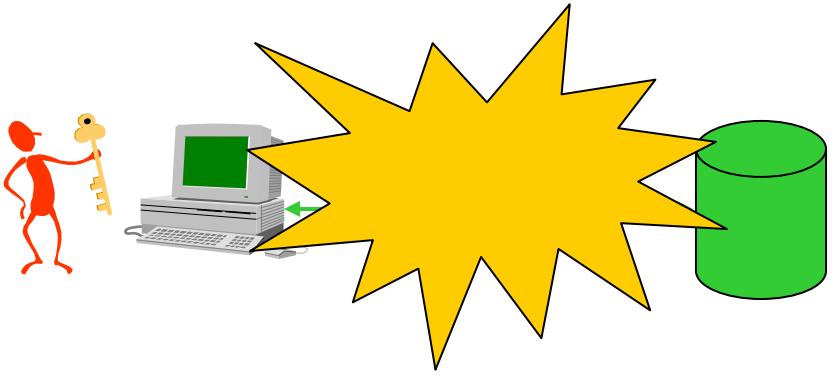
☺ Authentication
☺ Authorization

2: User



☹ Authentication
☺ Authorization

3: Privileged User



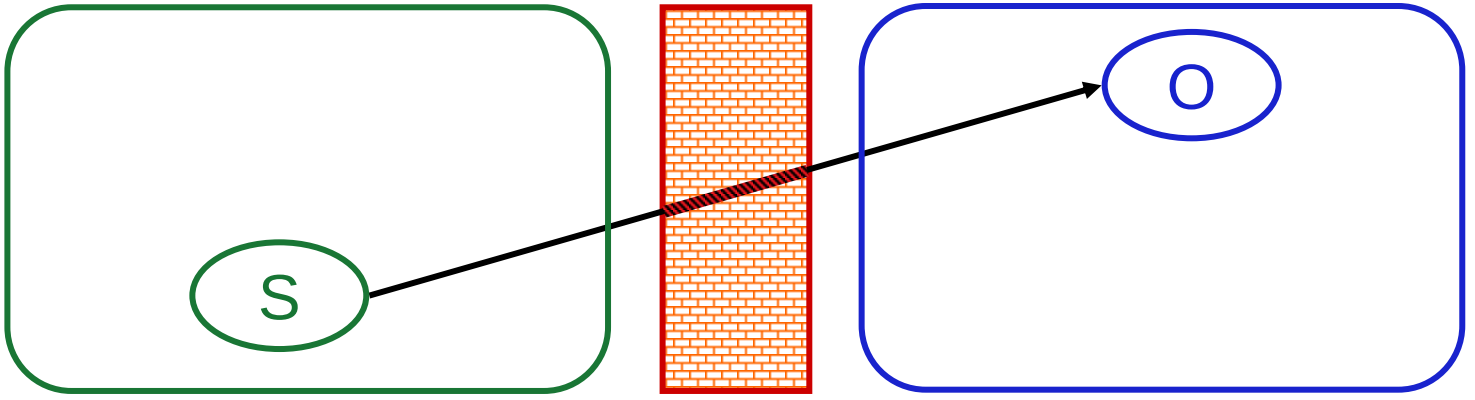
☹ Authentication
☹ Authorization

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

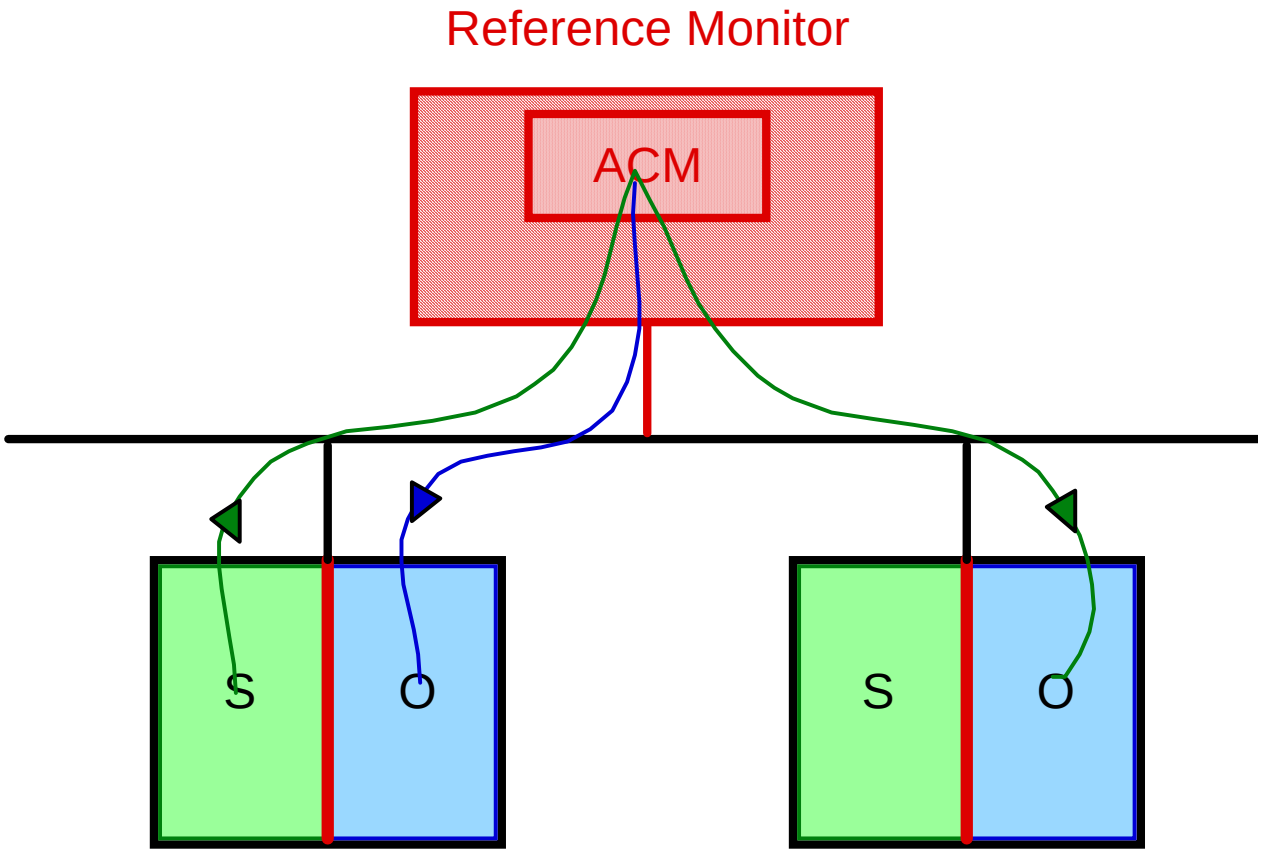


Subjects
=
Active Entities
(e.g., process, user, …)

Reference
Monitor
=
Tamperproof
Always Invoked
Verified Correct

Objects
=
Information Containers
(e.g., file, display, …)

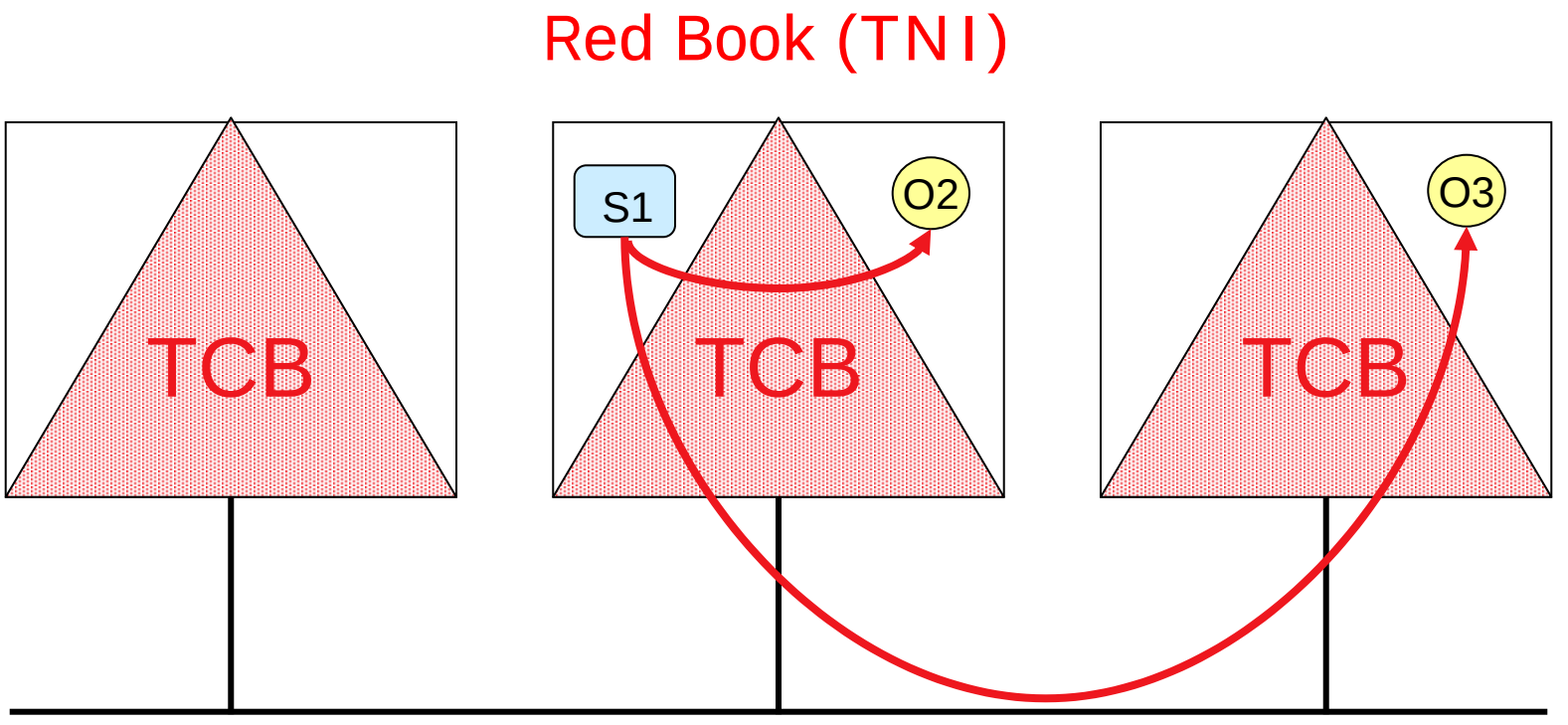
Distributed Authorization ? (1)



☺ small trusted area, easy administration

☹ bottleneck, single-point-of-failure

Distributed Authorisation ? (2)

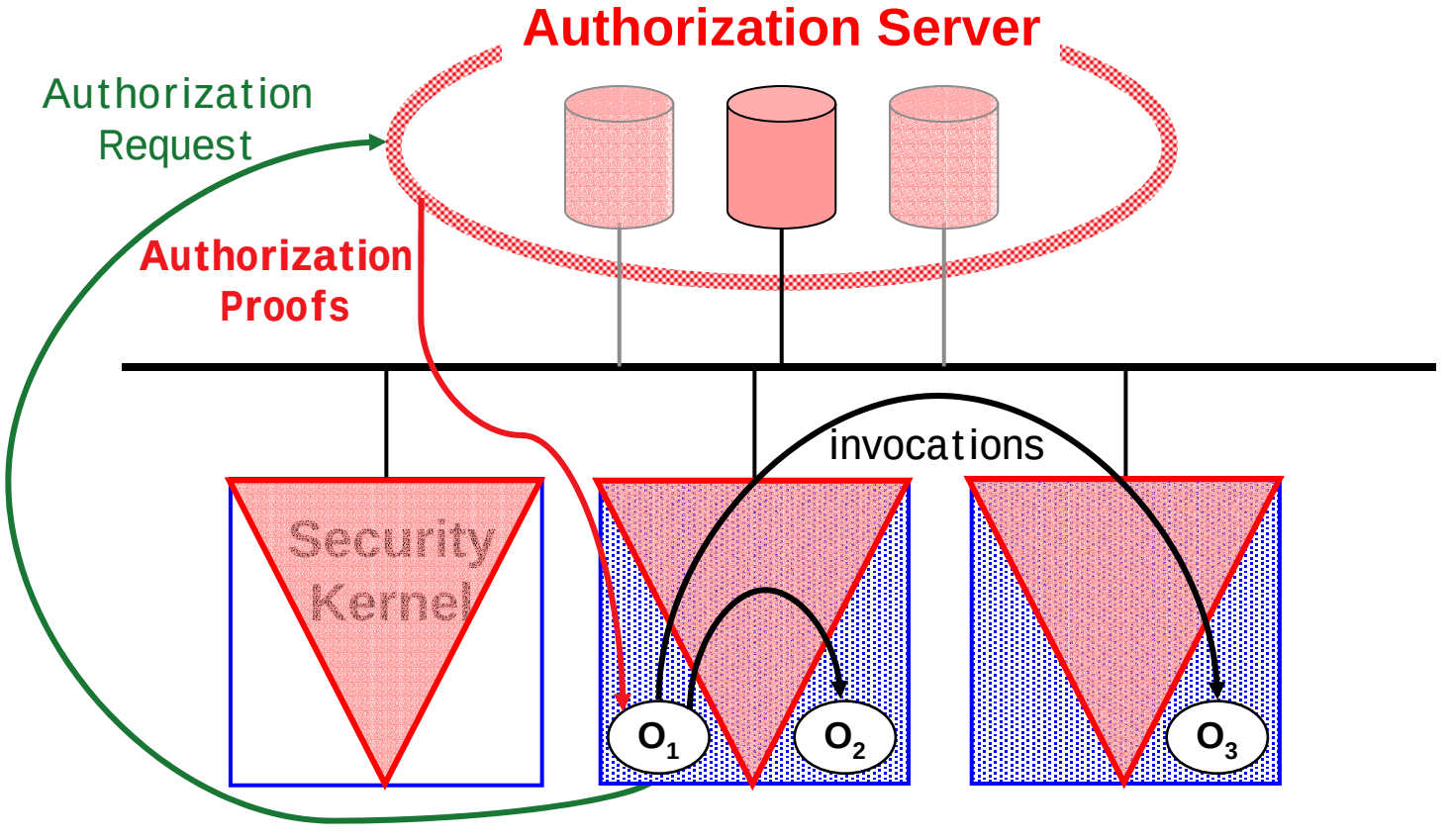


☺ No bottleneck, no single-point-of-accidental-failure

☹ Mutual trust between TCBs = multiple SPoFs
consistency?

Authorization Scheme for DOOS

[Nicomette & Deswarte, 1997]



IT Authorization Servers

Similar to WP4 TTPs

- Non-confidential information is replicated (atomic multicast WP2)
- Confidential information is shared securely (threshold crypto WP2)
- Global consensus is achieved (majority voting / Byzantine agreement WP2)
- Authorization proofs are distributed to local reference monitors (threshold signatures WP2)

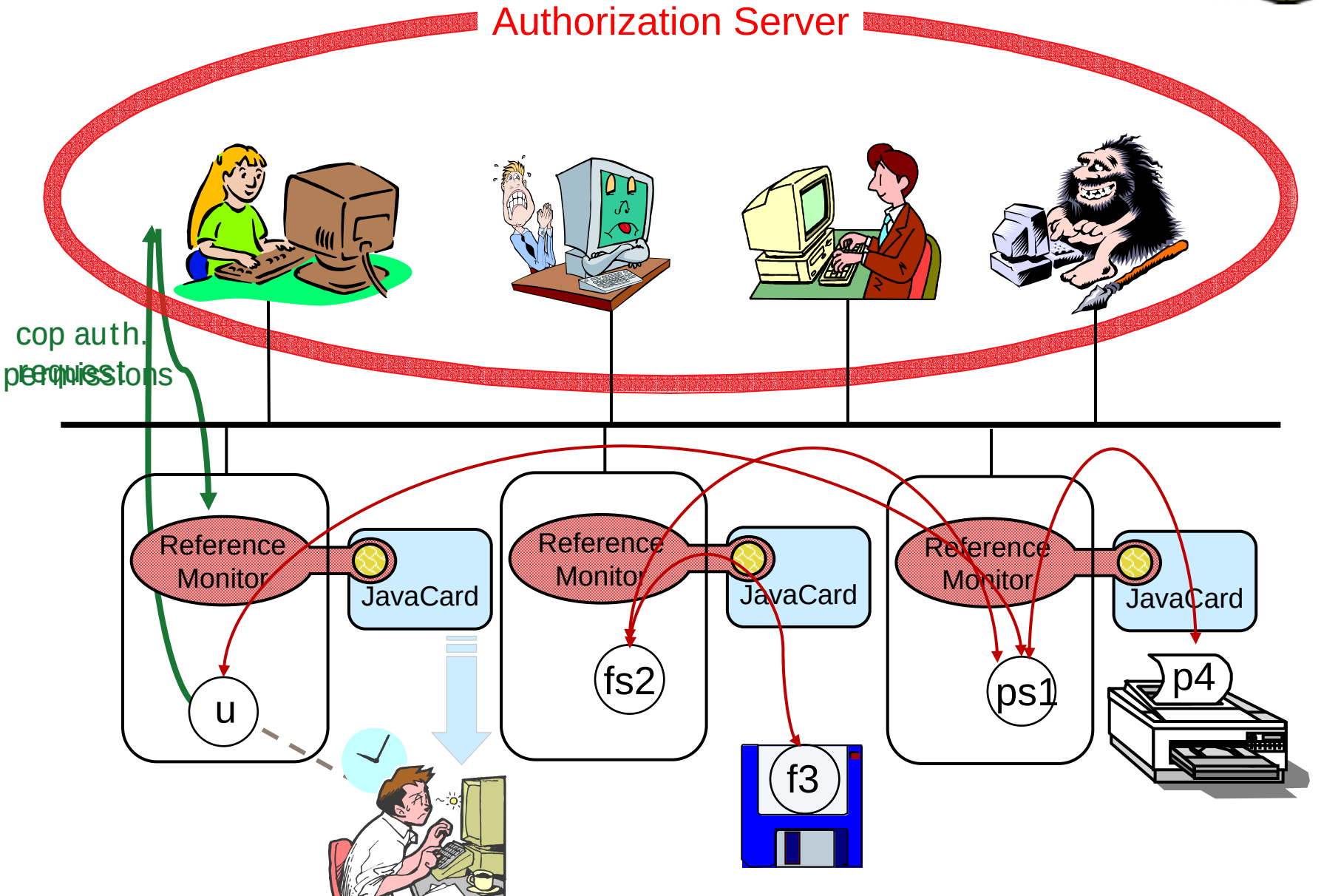
Local protection

- Internet applications: heterogeneous platforms ⇒ no modification of user workstations or even servers

  ⇒ no trusted security kernel, but JVM

- Local Reference Monitor =
  - Dispatcher (non-trusted Java object)
  - JavaCard (key storage + crypto computation)

Authorization Framework



Security properties (1)

- Authorization server:
  - AS1: The AS generates only valid authorization proofs (capabilities, tokens)
  - AS2: It is not possible to prevent AS from generating valid authorization proofs

  - Assumptions:
    - less than t faulty sites, with 3t +1 sites
    - no denial of service on communication
  - ✓ Byzantine agreement, threshold signatures

Security properties

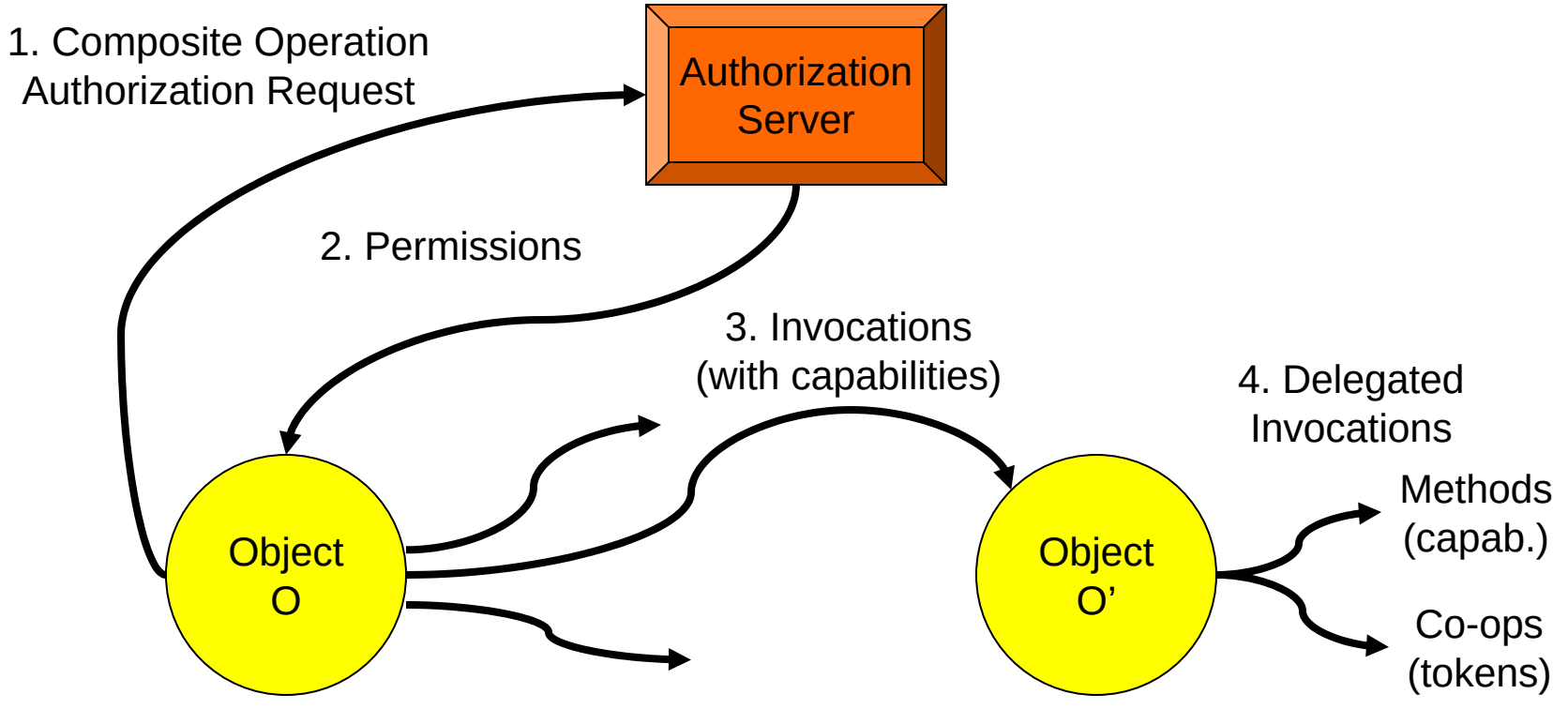
- Local Reference Monitors
  - RM1: Only valid operations will be executed on a non-faulty host
  - RM2: It is not possible to prevent the execution of an authorized operation on a non-faulty host

  - Assumptions:
    - no denial of service on communication
    - JavaCards cannot be cloned (the RM's private key is unique)
  - ✓ Verification of capabilities by RM, signed acknowledgements & time-outs

Functional Characteristics

- Composite operations:
  - Correspond to coordinated invocations of several objects, to realize a common distributed task, e.g., to print a file
  - Facilitate security management:
    - Automatic generation of capabilities (and *tokens*)
    - Automatic name resolution
- Vouchers:
  - New delegation scheme satisfying the least-privilege principle (as much as possible?)

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**($O'.m$)]}

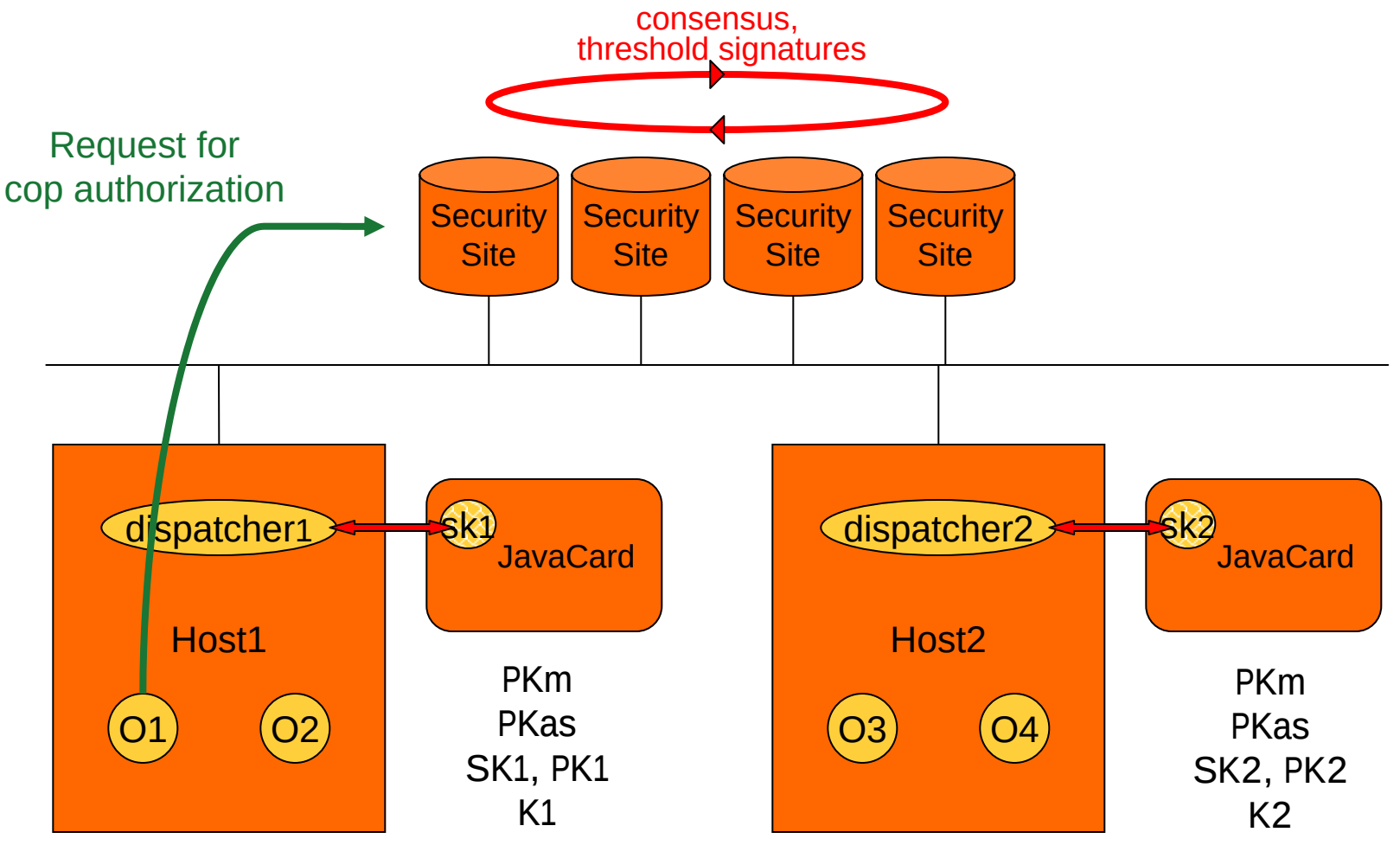
Perm(O,*cop*) = {O;*cop*($parC$);**token**(O;*cop*($parC$))}

cap(O;$O'.m$($parC$)) = <{O;$O'.m$;$parC$;*nonce*}>$_{SKas,PKhost(O')}$

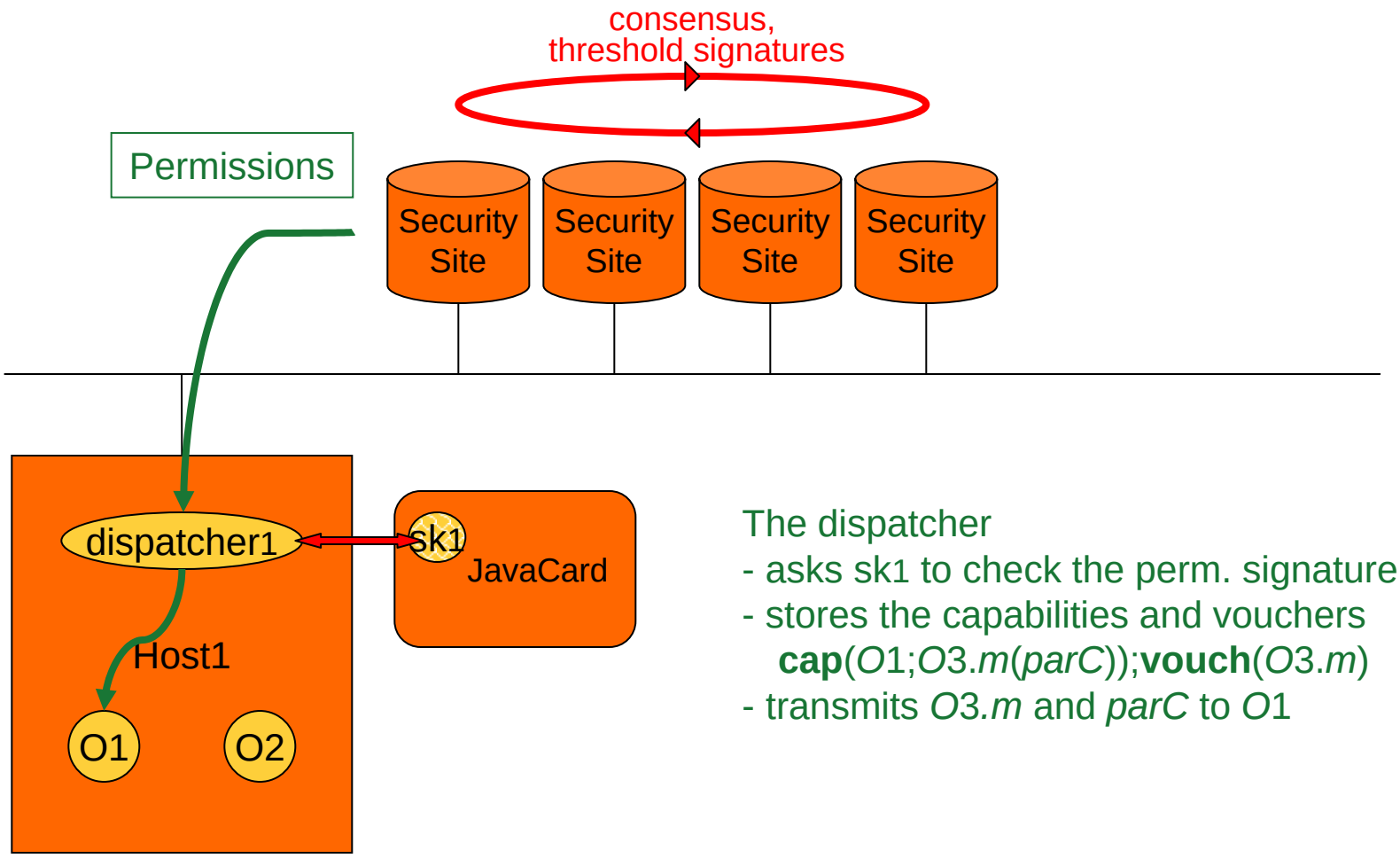
vouch($O'.m$) = <{Perm(O',$O''.m'$)*;Perm(O',*cop*)*}>$_{SKas}$

token(O;*cop*($parC$)) = <{O;*cop*;$parC$;*nonce*}>$_{SKas,PKas}$

Architecture

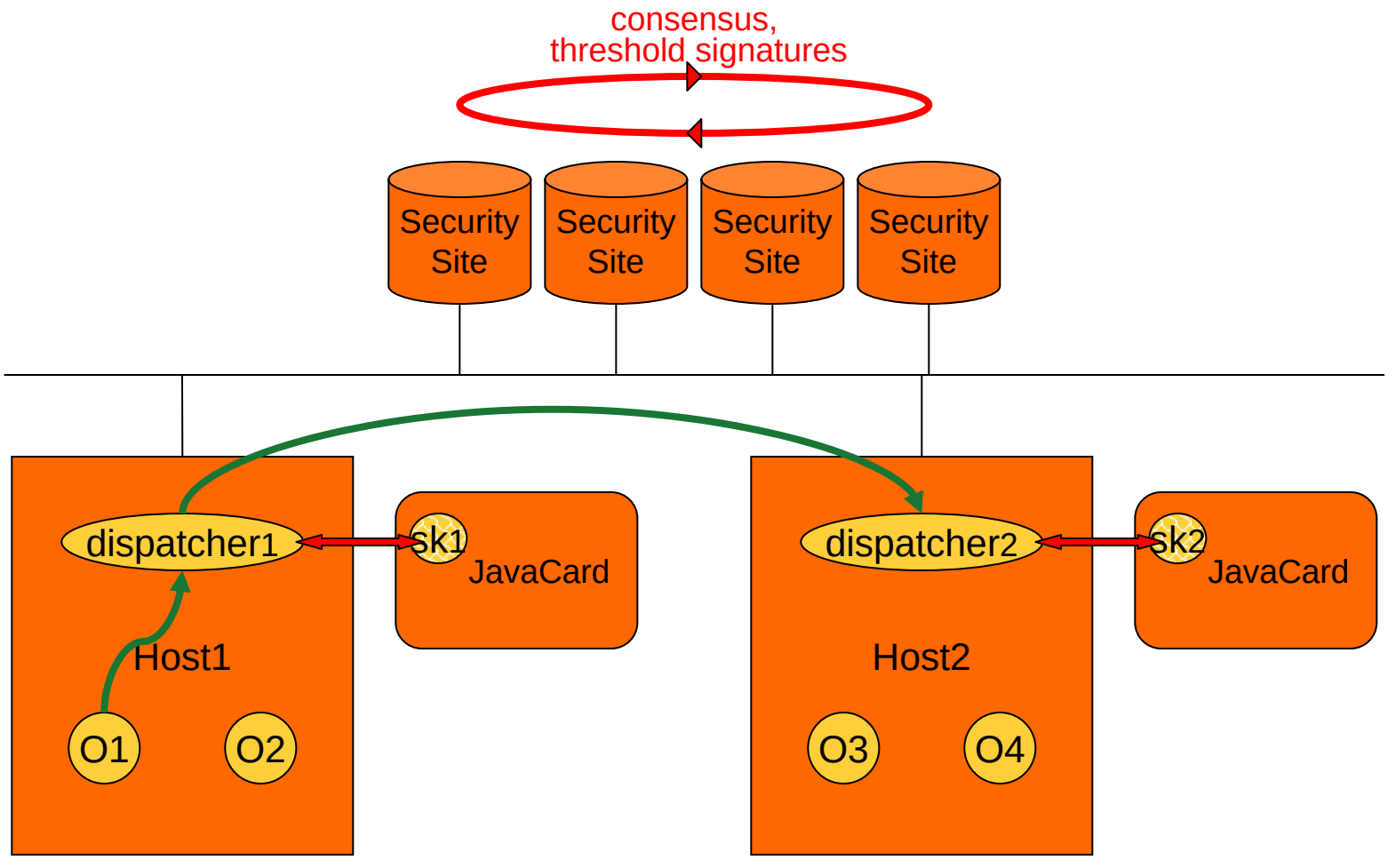


Architecture



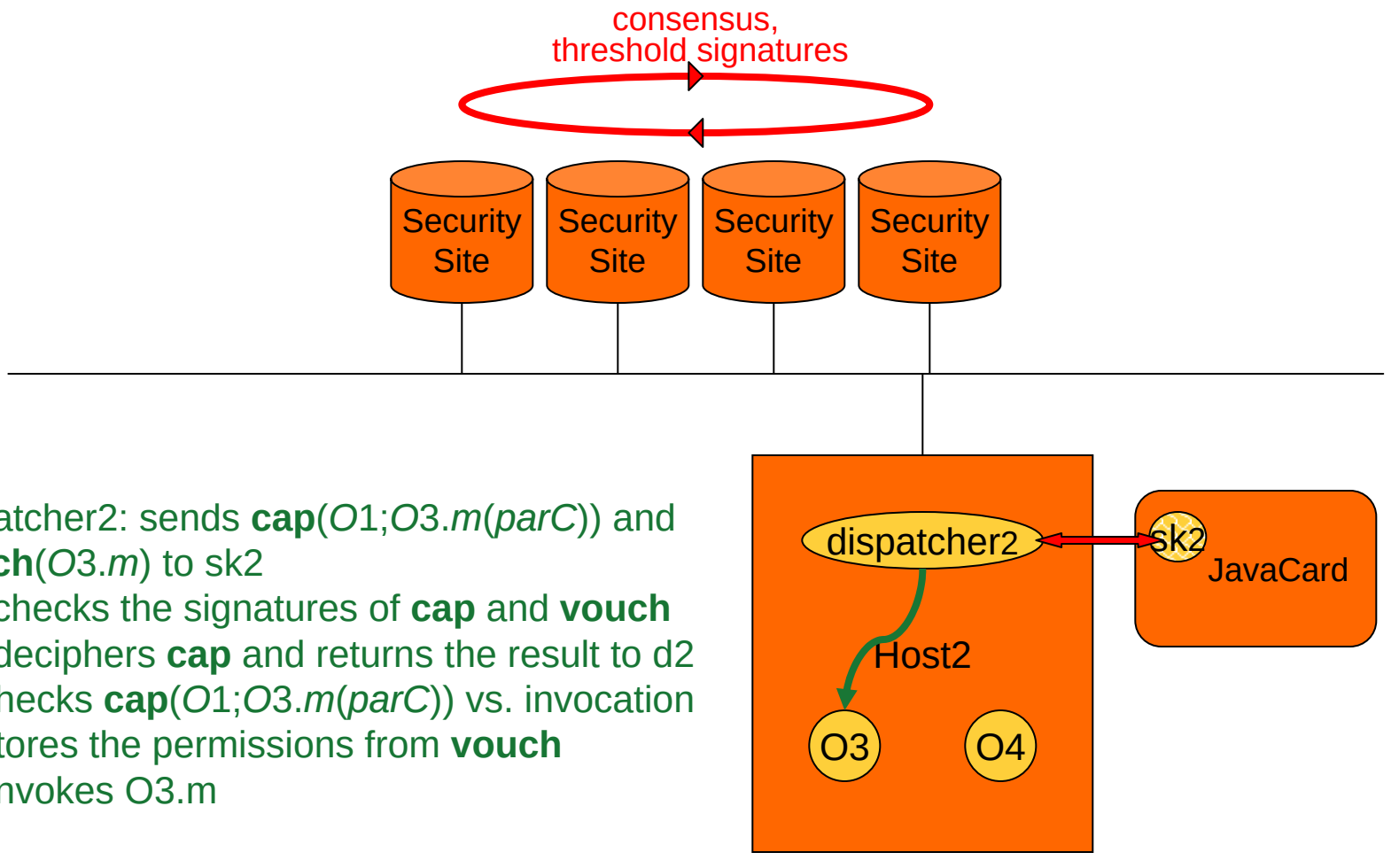
The dispatcher
- asks sk1 to check the perm. signature
- stores the capabilities and vouchers
  cap(*O*1;*O*3.*m*(*parC*));**vouch**(*O*3.*m*)
- transmits *O*3.*m* and *parC* to *O*1

Architecture



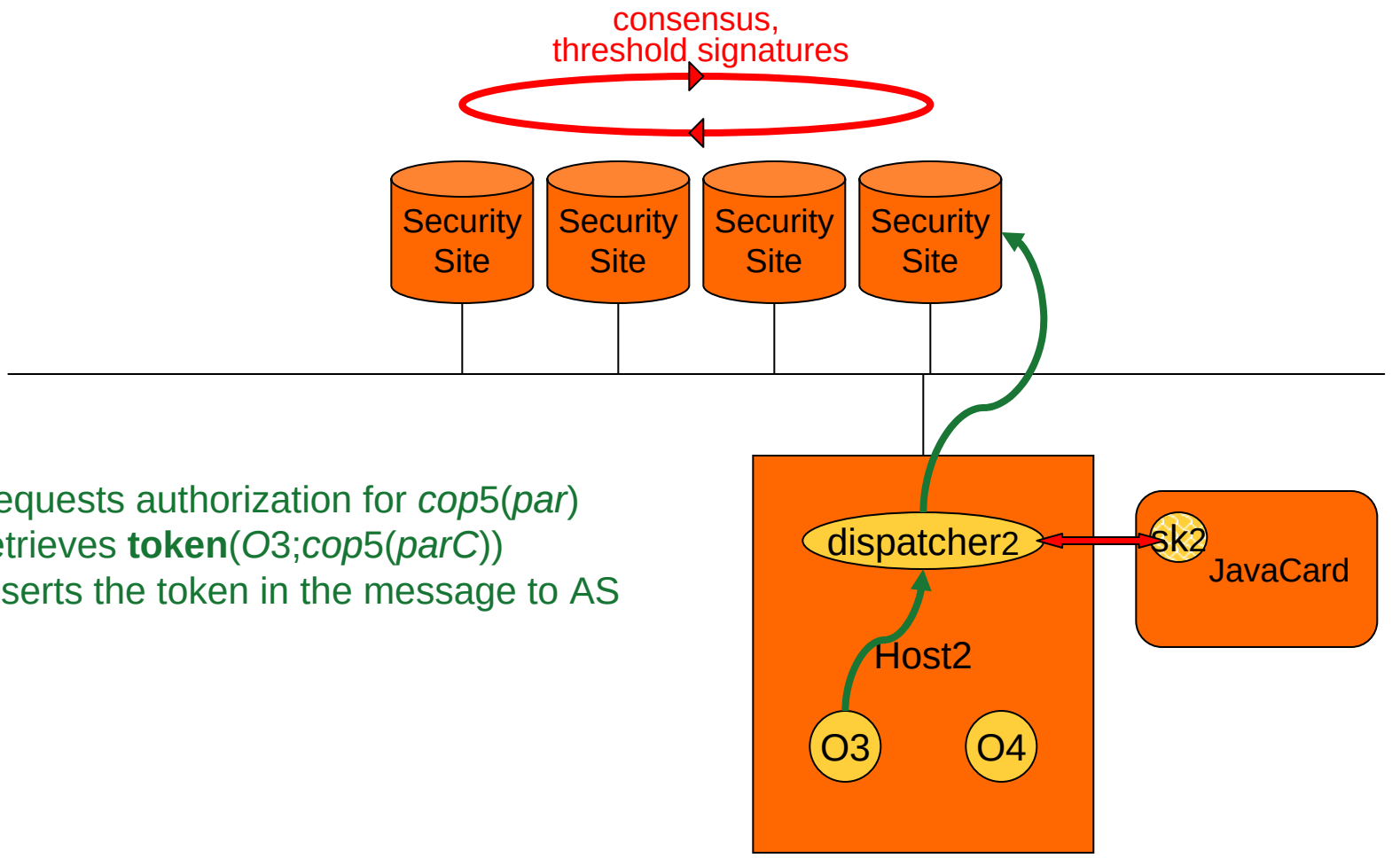
*O*1 invokes *O*3.*m* with {parameters}
dispatcher1 retrieves {**cap**(*O*1;*O*3.*m*(*parC*));**vouch**(*O*3.*m*)}, inserts it in the message to disp.2

Architecture



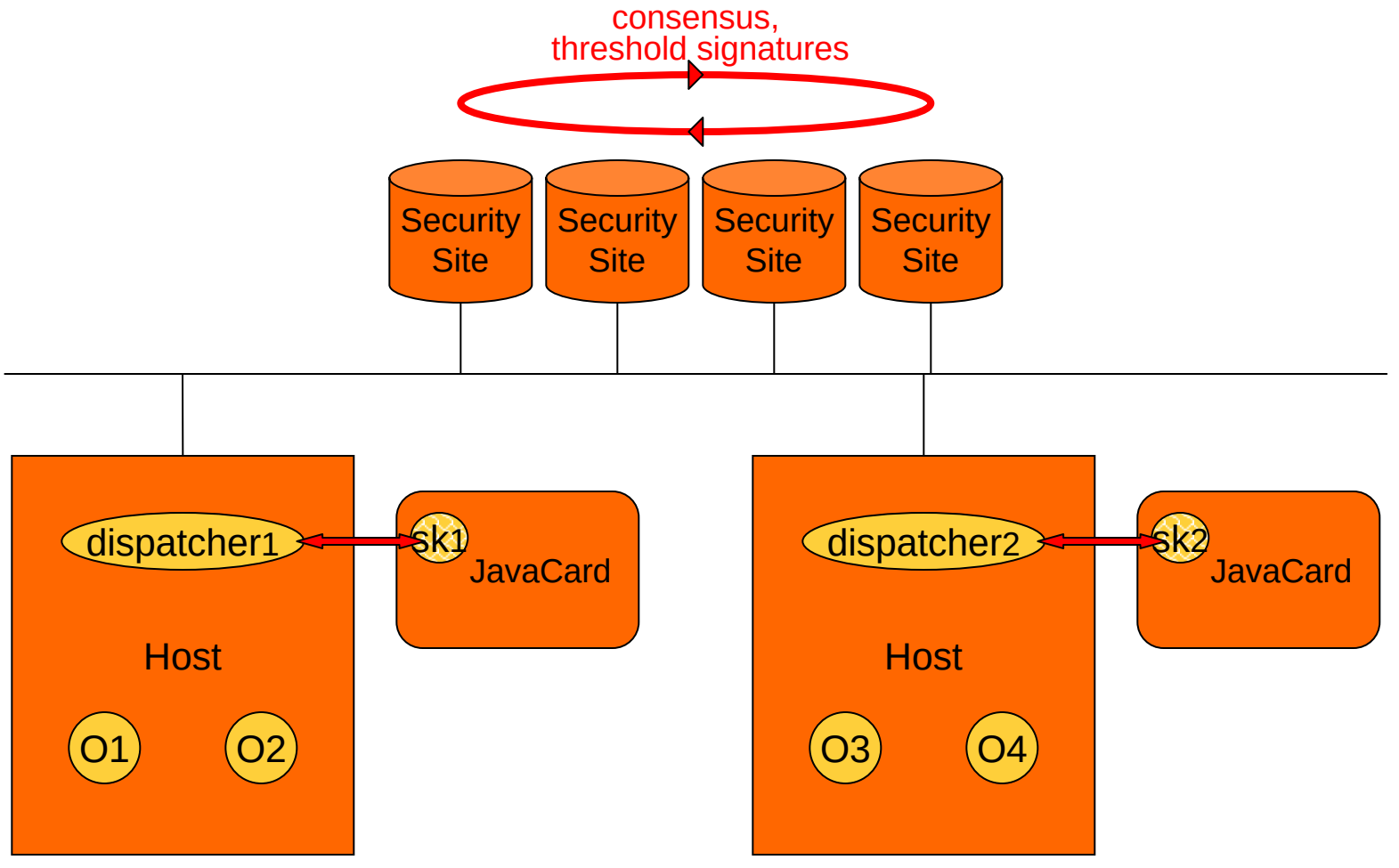
- dispatcher2: sends **cap**(*O*1;*O*3.*m*(*parC*)) and **vouch**(*O*3.*m*) to sk2
- sk2 checks the signatures of **cap** and **vouch**
- sk2 deciphers **cap** and returns the result to d2
- d2 checks **cap**(*O*1;*O*3.*m*(*parC*)) vs. invocation
- d2 stores the permissions from **vouch**
- d2 invokes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

Architecture



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

Architecture



WP5 Achievements

- 1st Year:
  - Scenario analysis
  - Authorization requirements
  - Definition of the authorization architecture
  - Deliverable D27
- 2nd Year:
  - Design & implementation of the reference monitor
  - Design of the non-FT authorization server
  - Deliverable D6
- 3rd Year
  - Implementation of the FT authorization server
  - Full Demonstrator (deliverable D14)