Sécurité informatique et tolérance aux fautes

Yves Deswarte

LAAS-CNRS & INRIA

Toulouse (France)

deswarte@laas.fr

Deux approches possibles

- Tolérance aux fautes pour la sécurité :

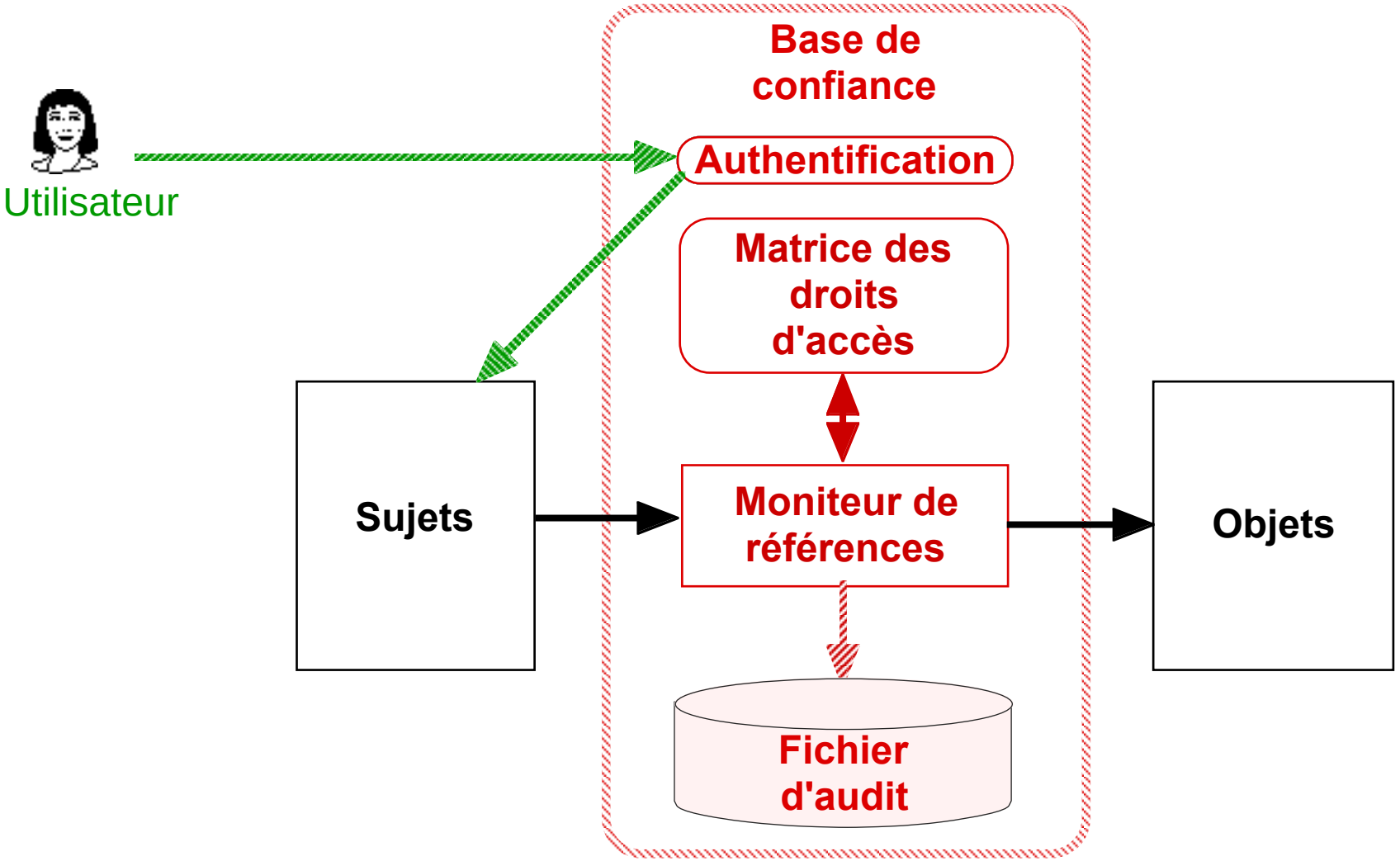
  tolérance aux malveillances : fautes malicieuses

- Appliquer une approche de sécurité informatique aux systèmes tolérants aux fautes

  politique d'intégrité pour des systèmes à plusieurs niveaux de criticité

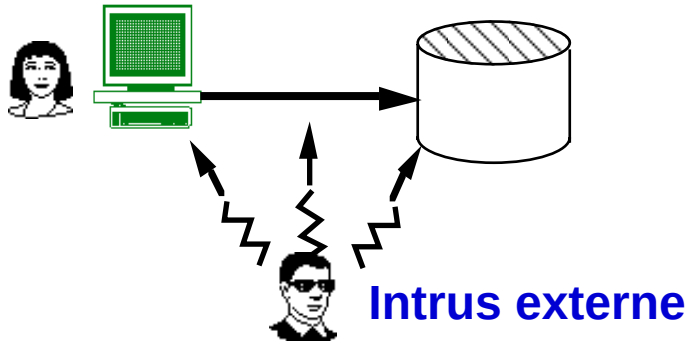
Vision classique de la sécurité : Prévention

Base de confiance (TCB) : incontournable, inviolable, prouvée correcte

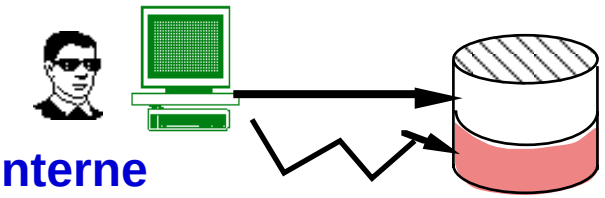


Qui sont les intrus ?

1)

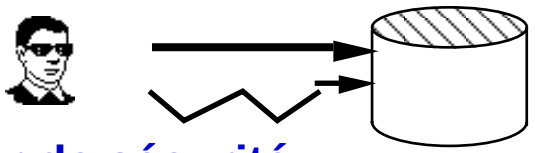


2)



Intrus interne (extension de privilèges)

3)

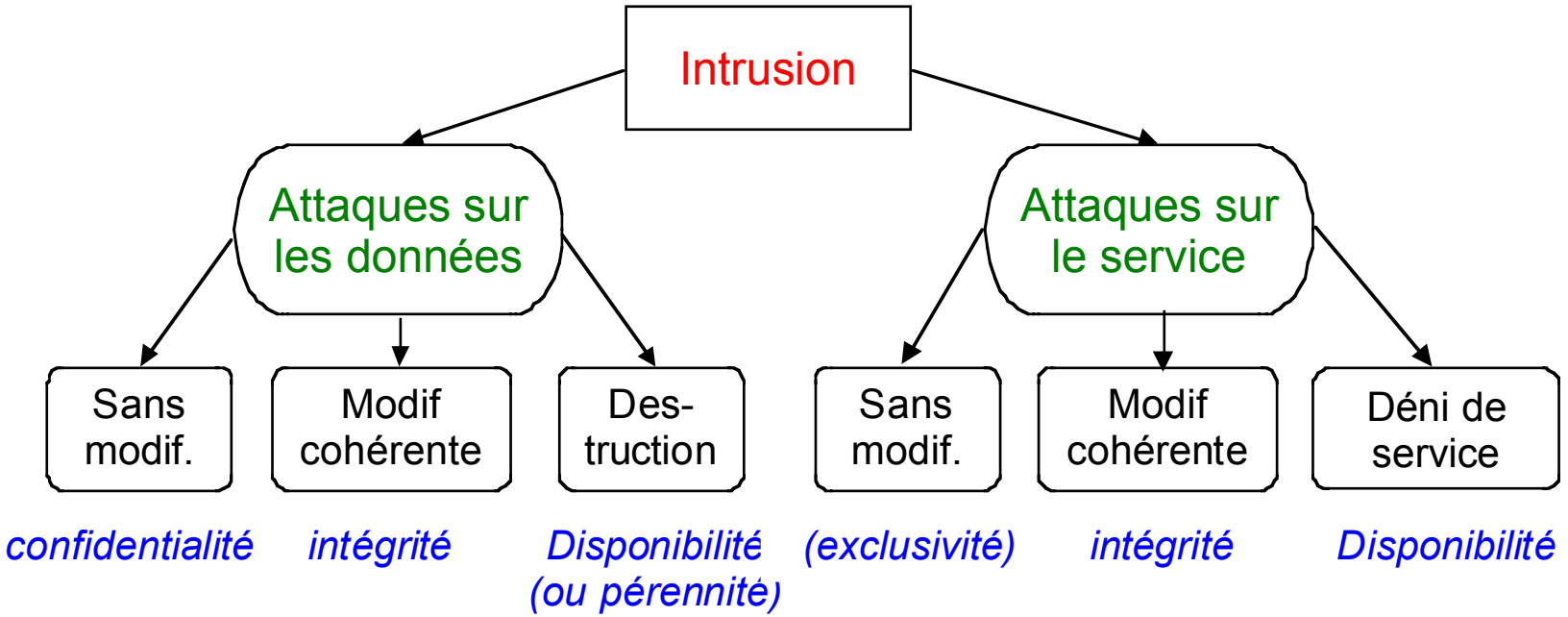


Administrateur de sécurité (abus de pouvoir)

80% des fraudes informatiques sont des abus de pouvoir

Effets des intrusions

Intrusions = *fautes d'interaction délibérées*



Tolérance aux intrusions ≈ Tolérance aux fautes accidentelles
(+ confidentialité + distribution non aléatoire)

Tolérance aux Fautes Intentionnelles

Tolérance aux fautes de conception intentionnelles :

porte dérobée, bombe logique, cheval de Troie, virus, vers...

Comme pour les fautes de conception accidentelles, la seule voie (peu explorée) semble être la diversification de la conception : production de N versions différentes avec ***vote*** sur les résultats)

Tolérance aux intrusions

☞ Chiffrement

➤ *confidentialité* (zone noire - zone rouge)

➤ *intégrité* : signature numérique pour détection des modifications : toute modification équivaut à une destruction

• mais inefficace pour la *disponibilité* : il faut utiliser des techniques de redondance analogues à celles de la tolérance aux fautes accidentelles.

☞ Réplication

• la réplication permet de se protéger contre les modifications et les destructions non-autorisées (comme pour les fautes d'origine accidentelle)

➤ l'intrus doit modifier/détruire la majorité, voire l'ensemble des copies avant de compromettre l'intégrité ou la disponibilité.

• Cependant, la réplication est néfaste pour la confidentialité.

☞ Détection et recouvrement des intrusions :

Détecter, ralentir, intervenir (+ enregistrer, corriger)

- **Détection :**
  - ➤ par les mécanismes de contrôle d'accès : authentification et autorisation
  - ➤ par discrimination entre les comportements normaux (utilisateurs autorisés) et les comportement anormaux (intrus)
    (systèmes experts, analyse statistique)
  - ➤ autres détections : audits de sécurité, détection d'erreurs, défaillances
- **Ralentissement :**
  - ➤ résilier les droits
  - ➤ tromper l'attaquant (intoxication)
- **Intervention :**
  - ➤ identifier et localiser l'attaquant
  - ➤ poursuites judiciaires
  - ➤ restauration des informations détruites

☞ Brouillage

- Ajouter des informations superflues (*bourrage*), p.ex. des messages
- Augmentation de l'incertitude dans les réponses aux requêtes statistiques dans les bases de données (tolérance aux attaques de déduction par inférences)

Fragmentation-Redondance-Dissémination (FRD)

Objectif :

Faire en sorte que l'intrusion d'une partie du système ne donne accès qu'à des informations non significatives : *confidentialité, intégrité, disponibilité*

Principe :

- ➤ Découper l'information en fragments de telle sorte que des fragments isolés ne fournissent pas d'information significative
- ➤ Ajouter de la redondance aux fragments pour tolérer les attaques en modification, destruction ou déni de service
- ➤ Isoler les fragments en les disséminant :

= *Dissémination spatiale* : utiliser différents sites de stockage ou voies de communication

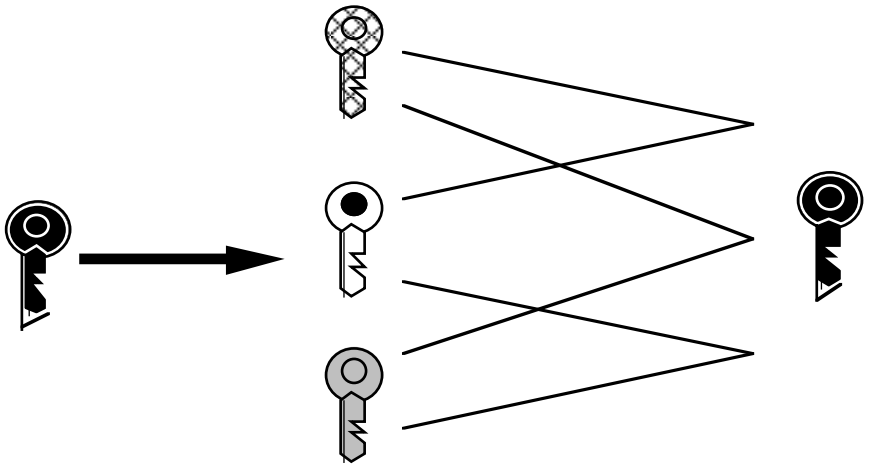
= *Dissémination temporelle* : mélanger les fragments envoyés avec d'autres fragments, ou les transmettre dans un ordre aléatoire

= *Dissémination fréquentielle* : envoyer les fragments sur des fréquences différentes (communications à large bande)

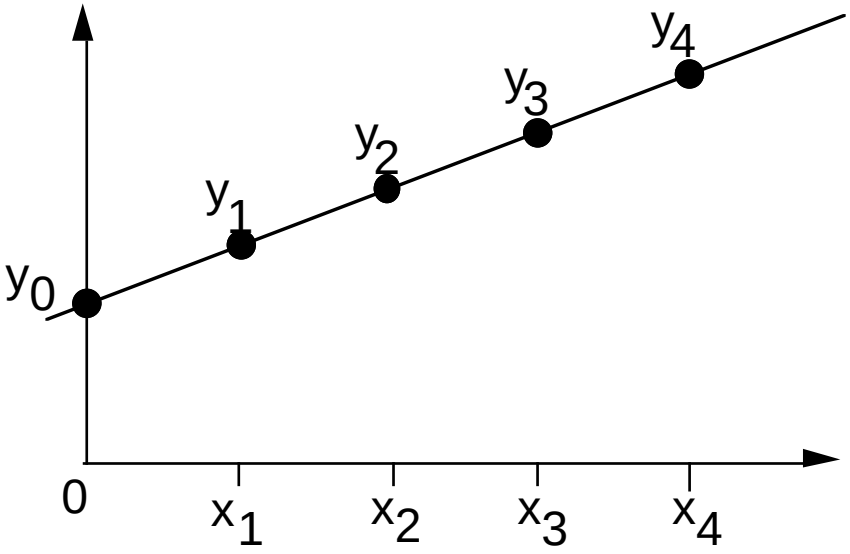
= *Dissémination des privilèges* : exiger l'accord de plusieurs utilisateurs ou entités avant d'exécuter une opération (secret partagé, ou séparation des pouvoirs)

Exemple : Schémas à seuil

- Stocker K sous la forme d'un ensemble de valeurs K_i (*images*), telles que:
  - S images permettent de reconstruire le secret (S = seuil)
  - S-1 images n'apportent aucune information
- Si l'on sait générer N images (avec N>S), alors jusqu'à (N-S) images peuvent être détruites



Exemple simple : Seuil = 2



$P(x) = \{y = a\,x + b\}$

$y_0 = a\,x_0 + b = b$
$y_1 = a\,x_1 + b$
$y_2 = a\,x_2 + b$
•••
$y_m = a\,x_m + b$

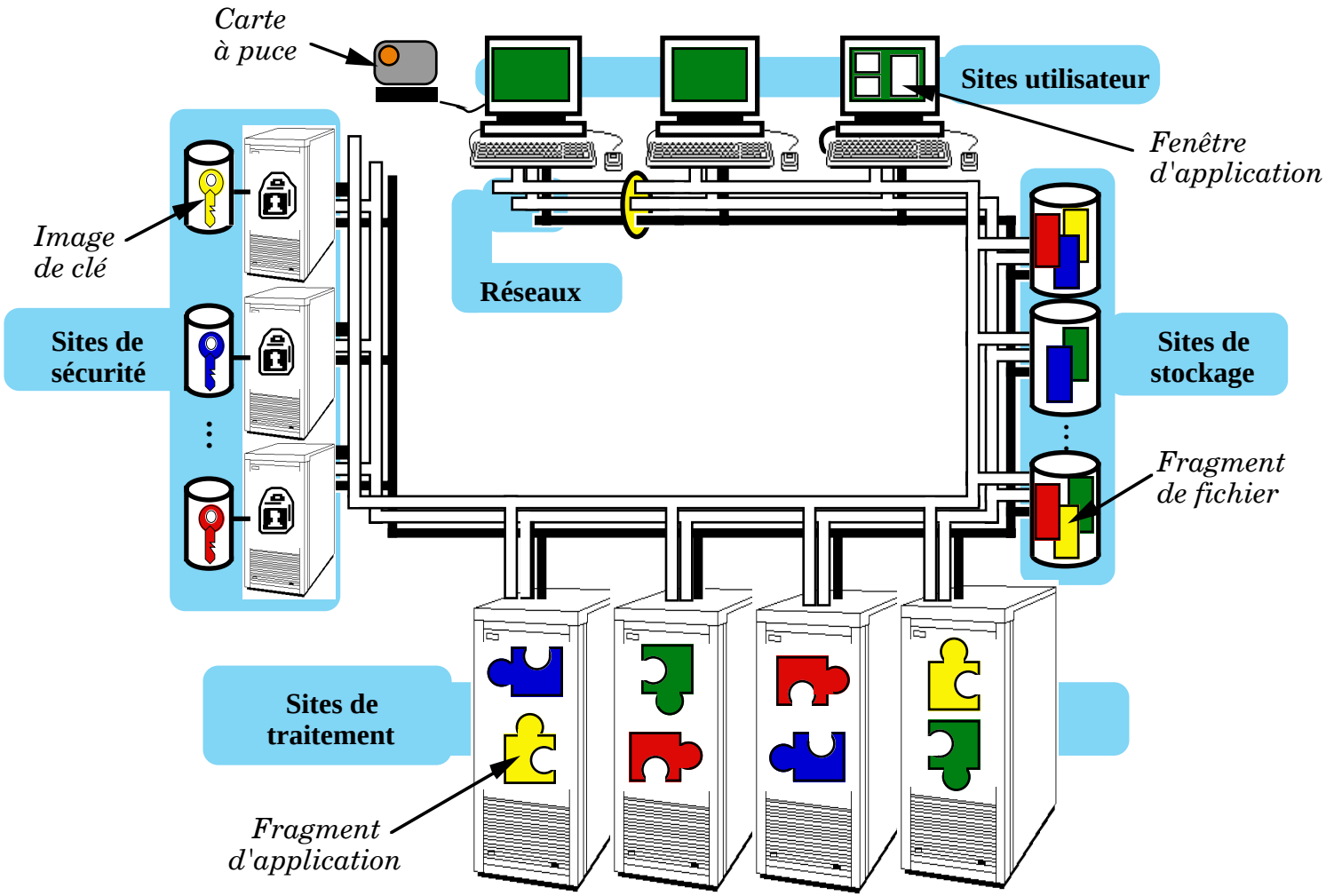
a, b secrets
x_1, x_2, ••• x_m fixés

=> à partir de deux points quelconque on sait calculer a et b

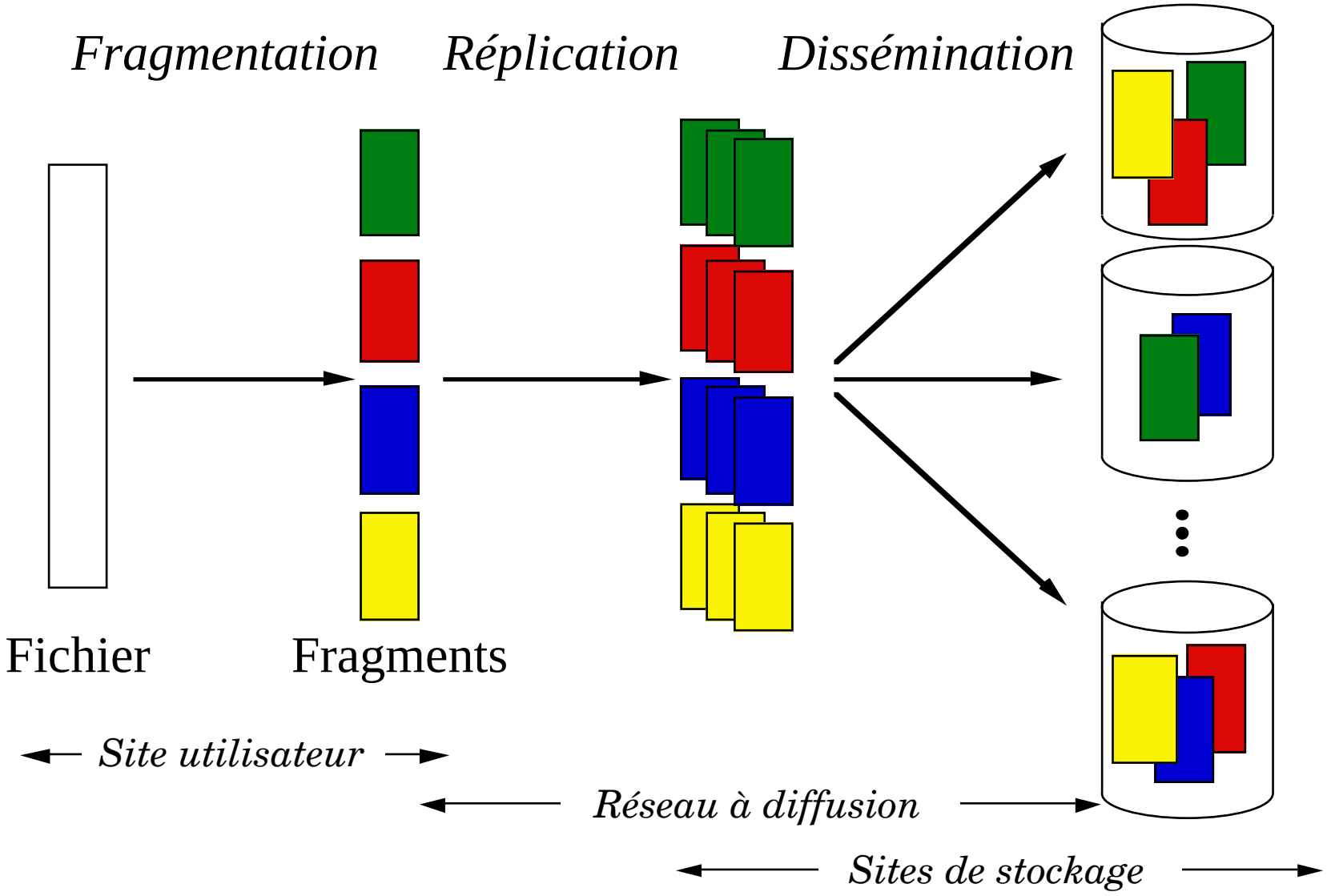
- avec 2 images quelconques, on reconstruit le secret
- avec une seule image, on n'a rien

=> Seuil = 2

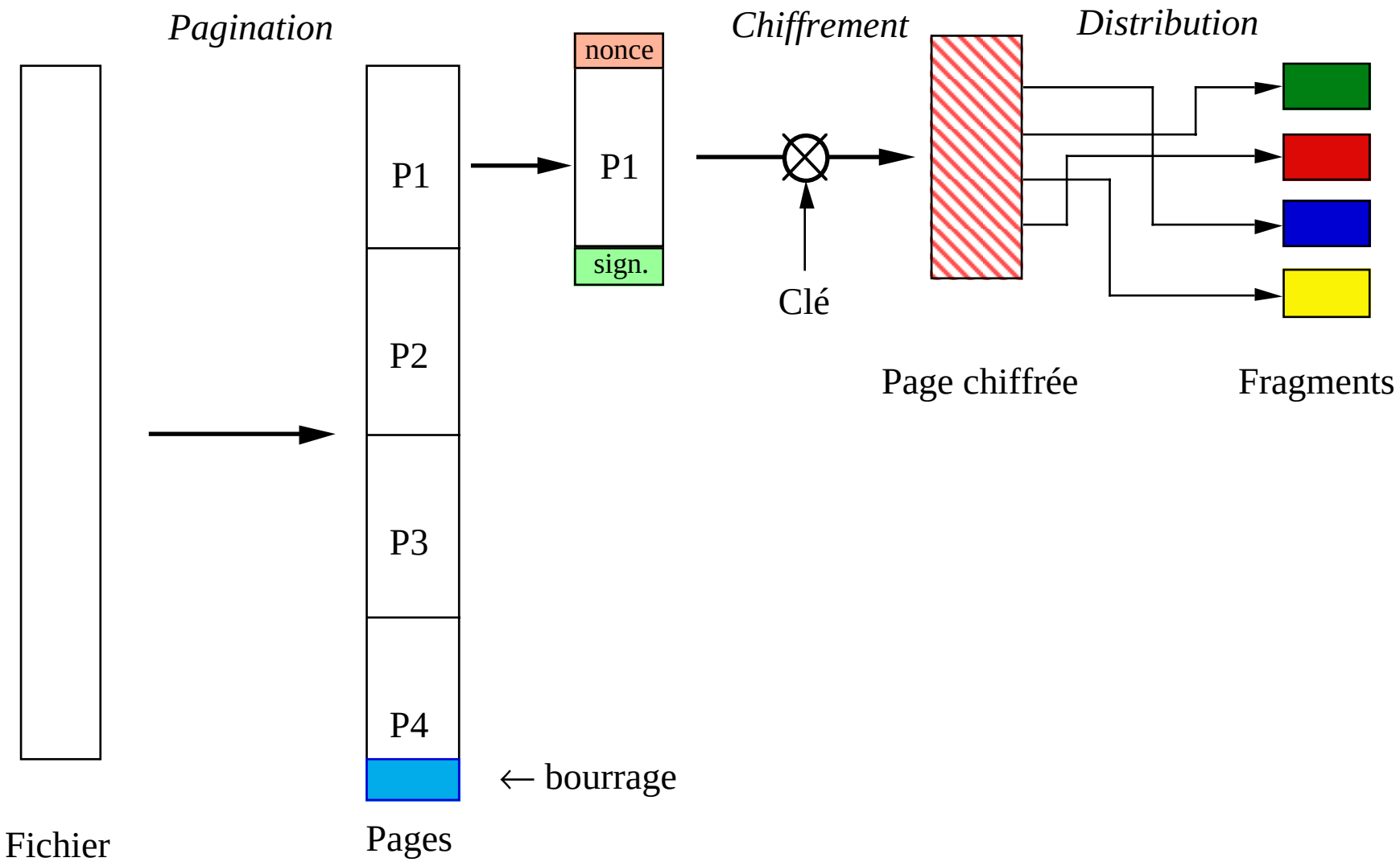
Prototype de système basé sur la FRD



FRD appliquée à la gestion de fichiers

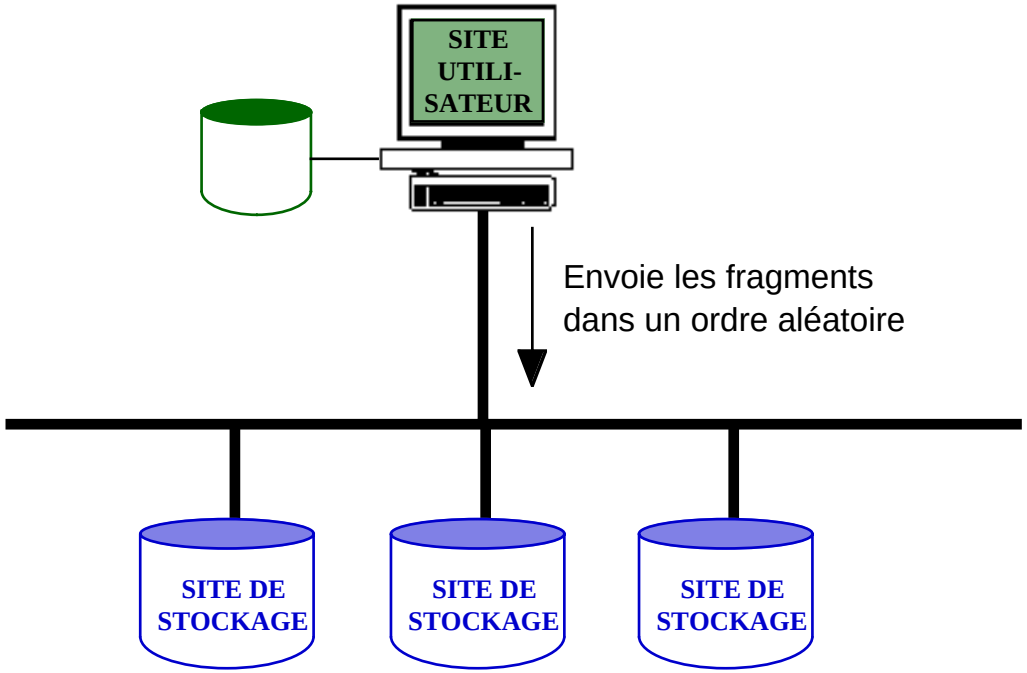


Fragmentation



Écriture

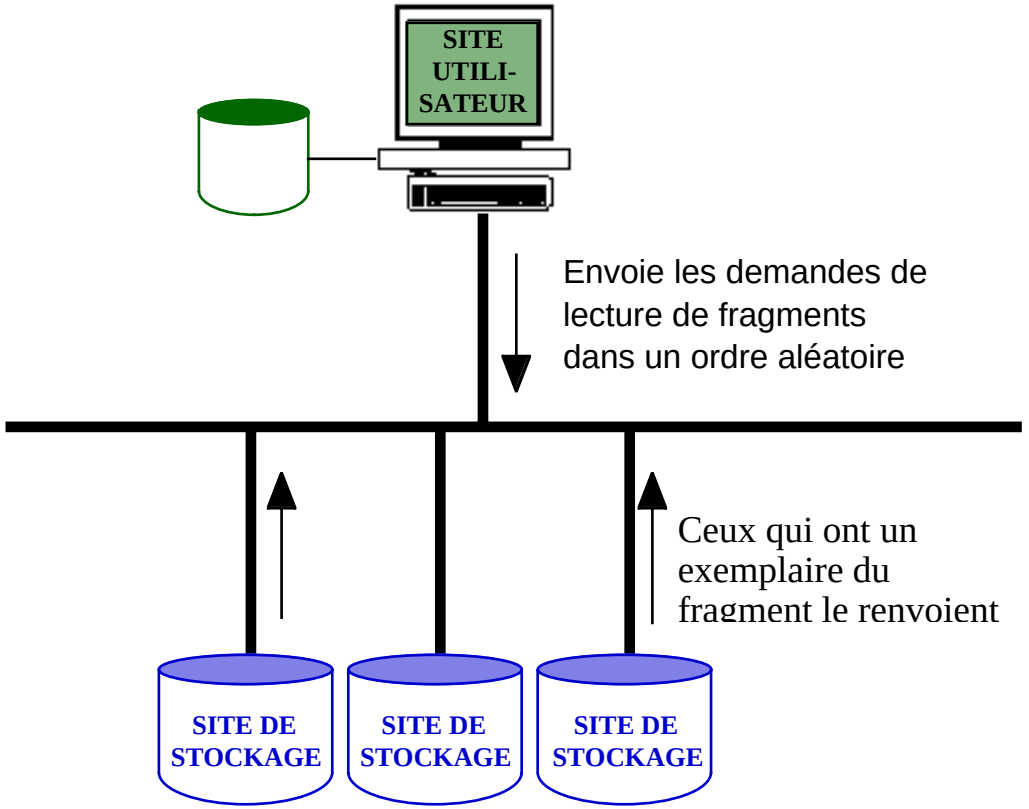
(nom du fichier, n°page, n°fragment, clé) —**Fonction à sens unique**→ **nom de fragment**



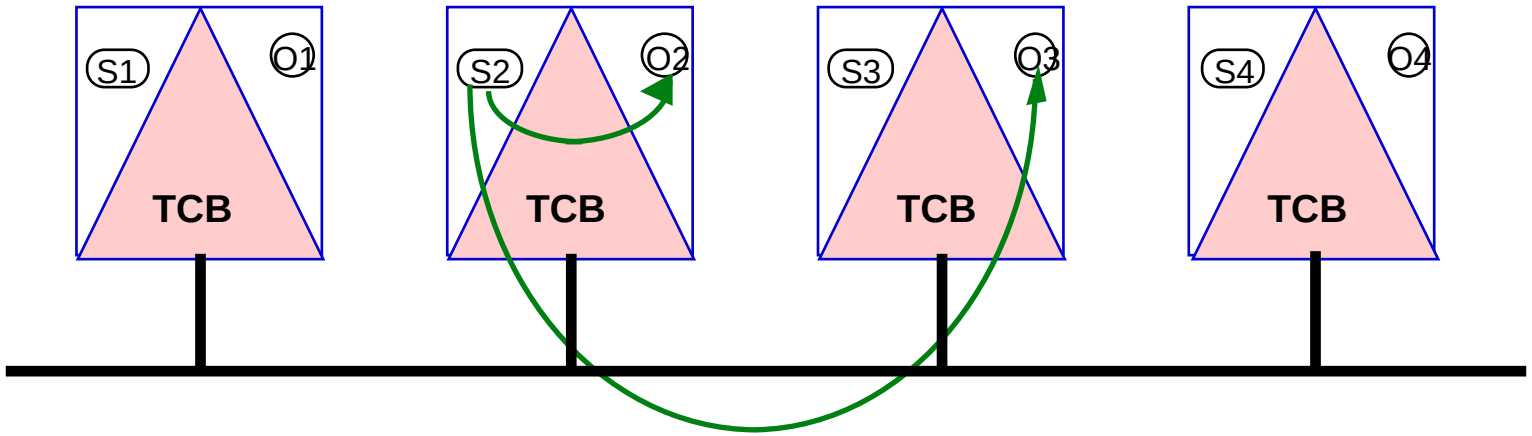
Décident quel site stocke quel

Lecture

(nom du fichier, n°page, n°fragment, clé) —Fonction à sens unique→ nom de fragment

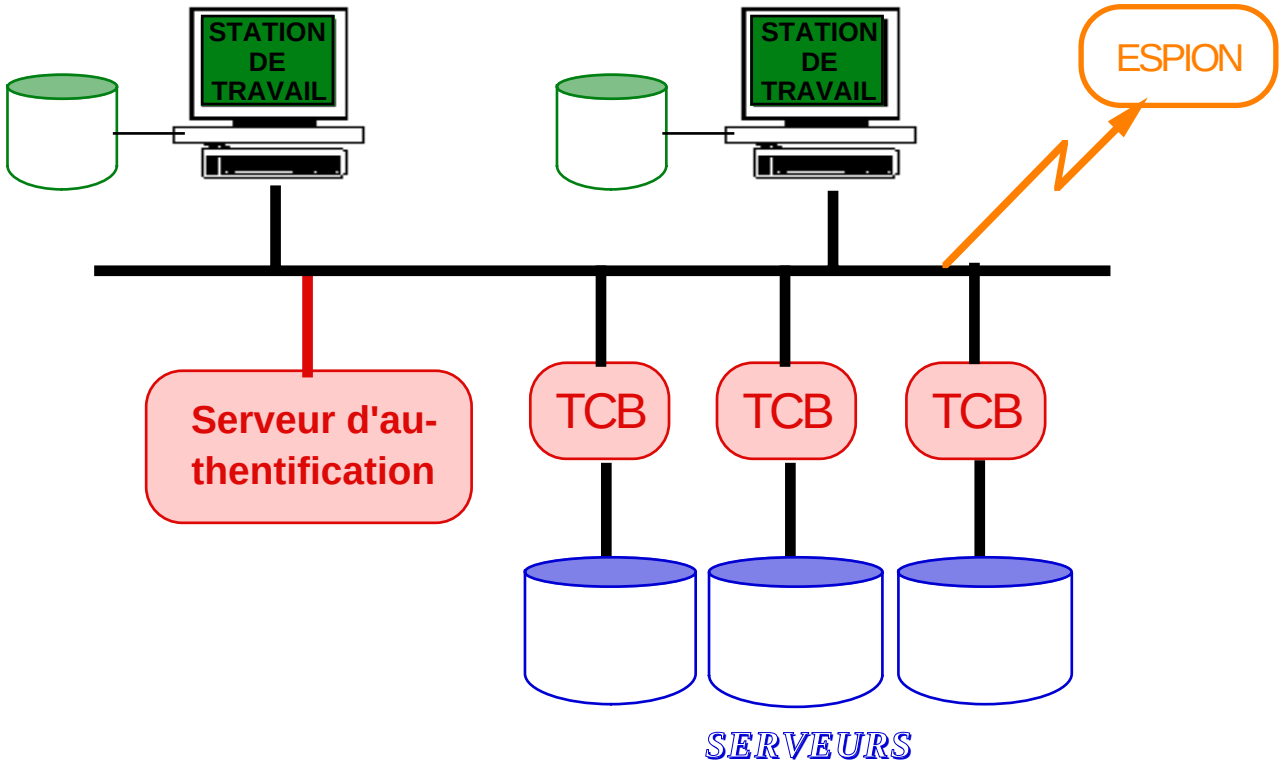


Vision classique : TNI (*Livre Rouge*)



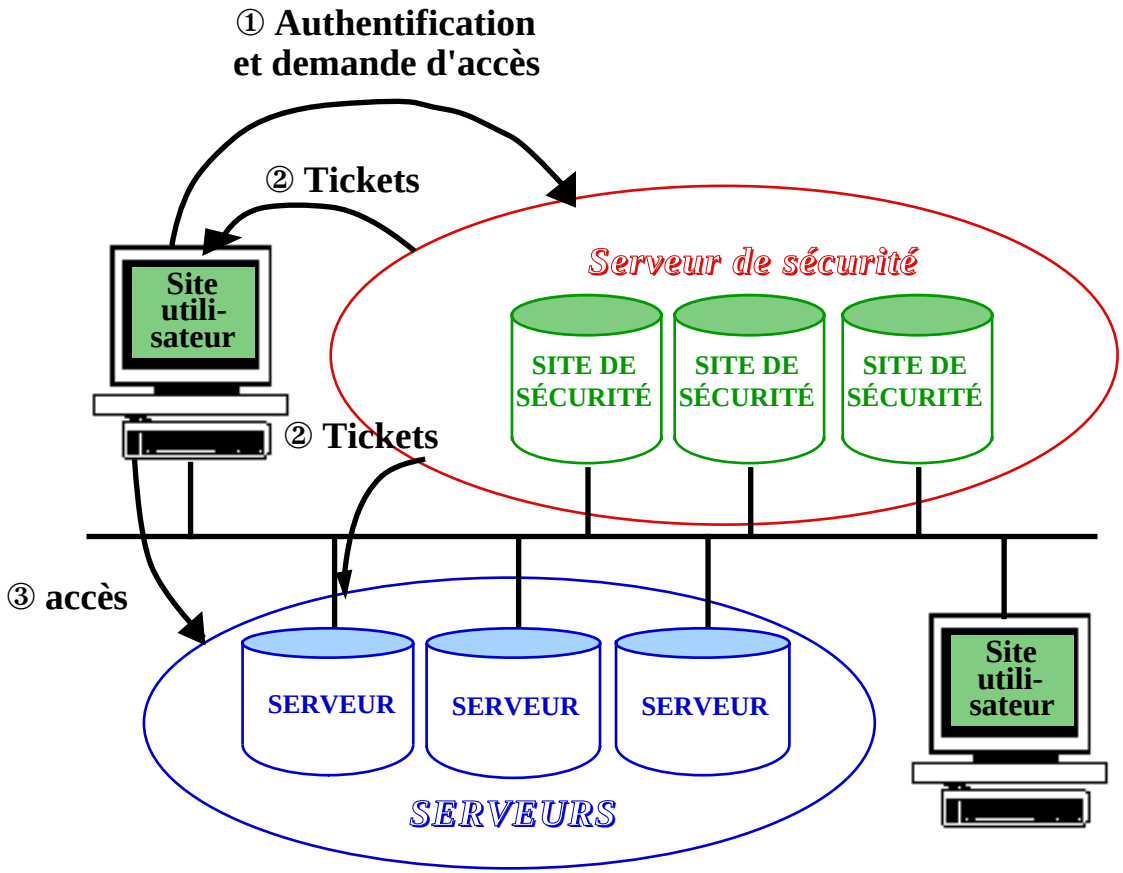
- une TCB par site
- confiance mutuelle entre les TCB (pour des niveaux de sécurité donnés)
- confiance dans les administrateurs (ou officiers de sécurité) de chaque site

Approche de Kerberos , SESAME :



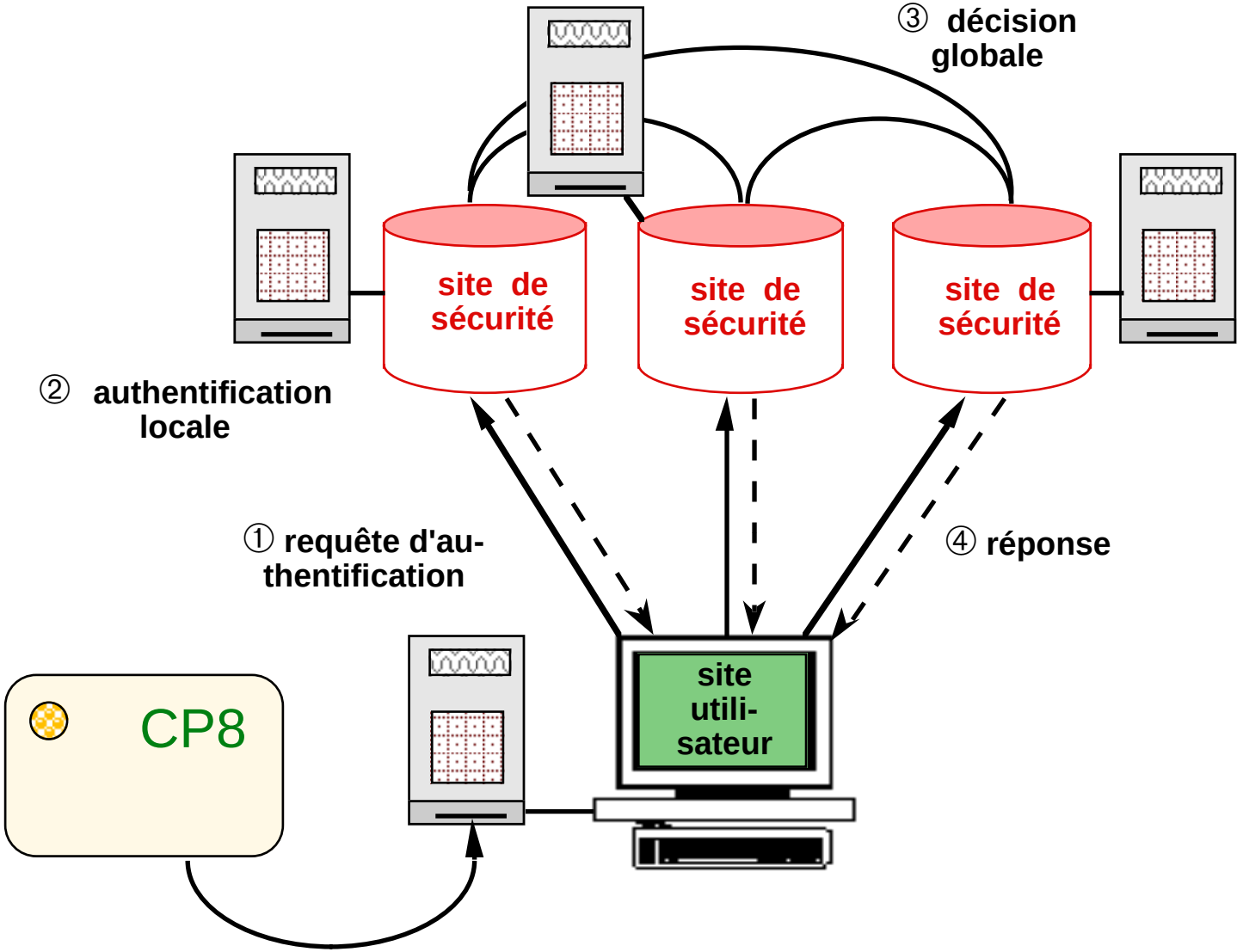
- ≈ pas de confiance dans les stations de travail
- confiance dans les serveurs et leurs administrateurs (ou officiers de sécurité)
- ≈ incompatible avec les politiques "obligatoires" (Mandatory Access Control).

FRD appliquée à la gestion de la sécurité

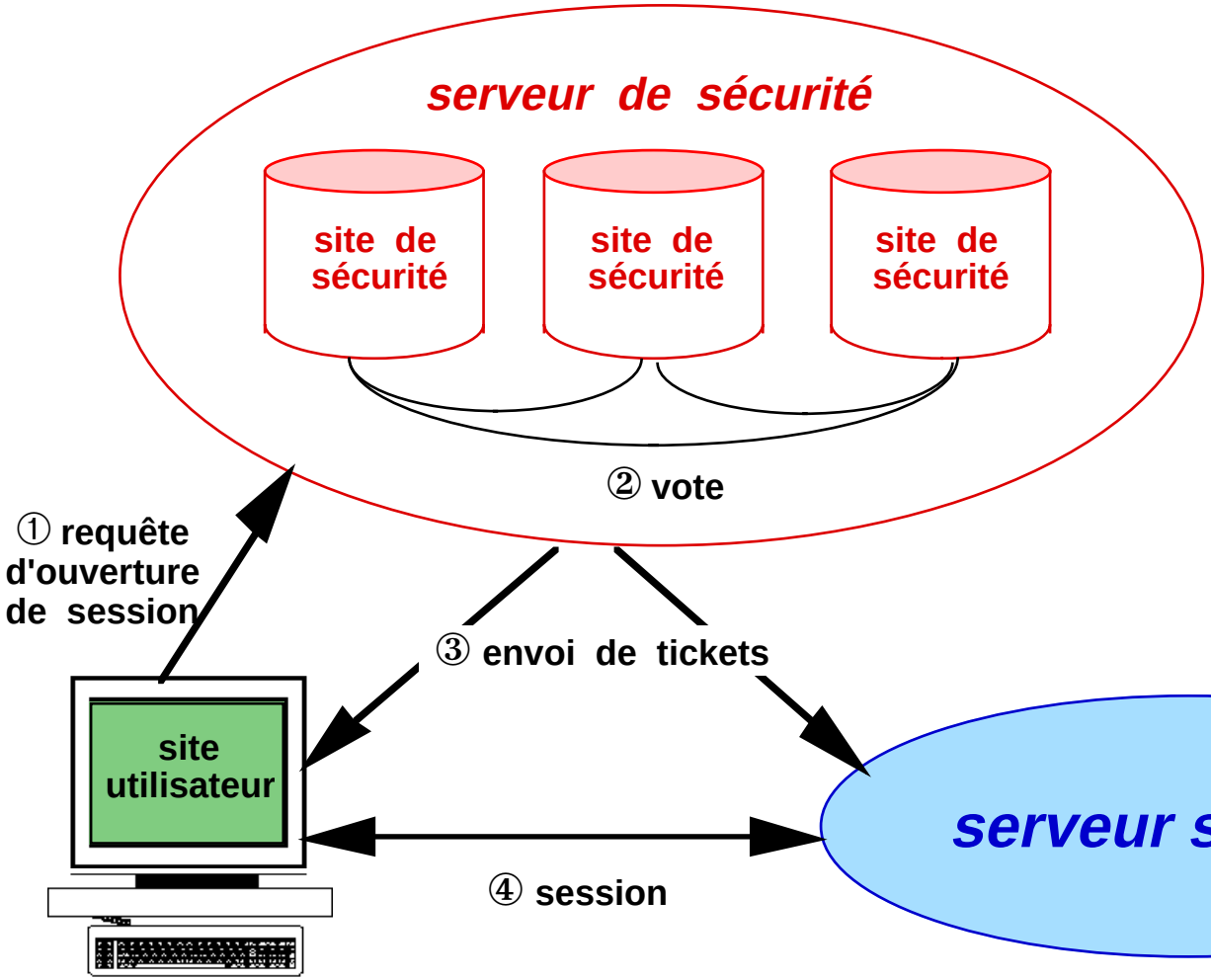


- service de sécurité réparti, *tolérant les intrusions*, responsable de l'authentification et de l'autorisation
- ≈ pas de confiance dans les stations ni dans les serveurs (individuellement), ni dans leurs administrateurs (ou officiers de sécurité)

Authentification



Autorisation



fichier1	droits	i(k)
fichier2		
fichier3		
fichier4		
...		

FRD pour le traitement fiable de données confidentielles

Problème :

*Comment faire exécuter des traitements **redondants***
*sur des données **confidentielles***
*par des calculateurs **non dignes de confiance** ?*

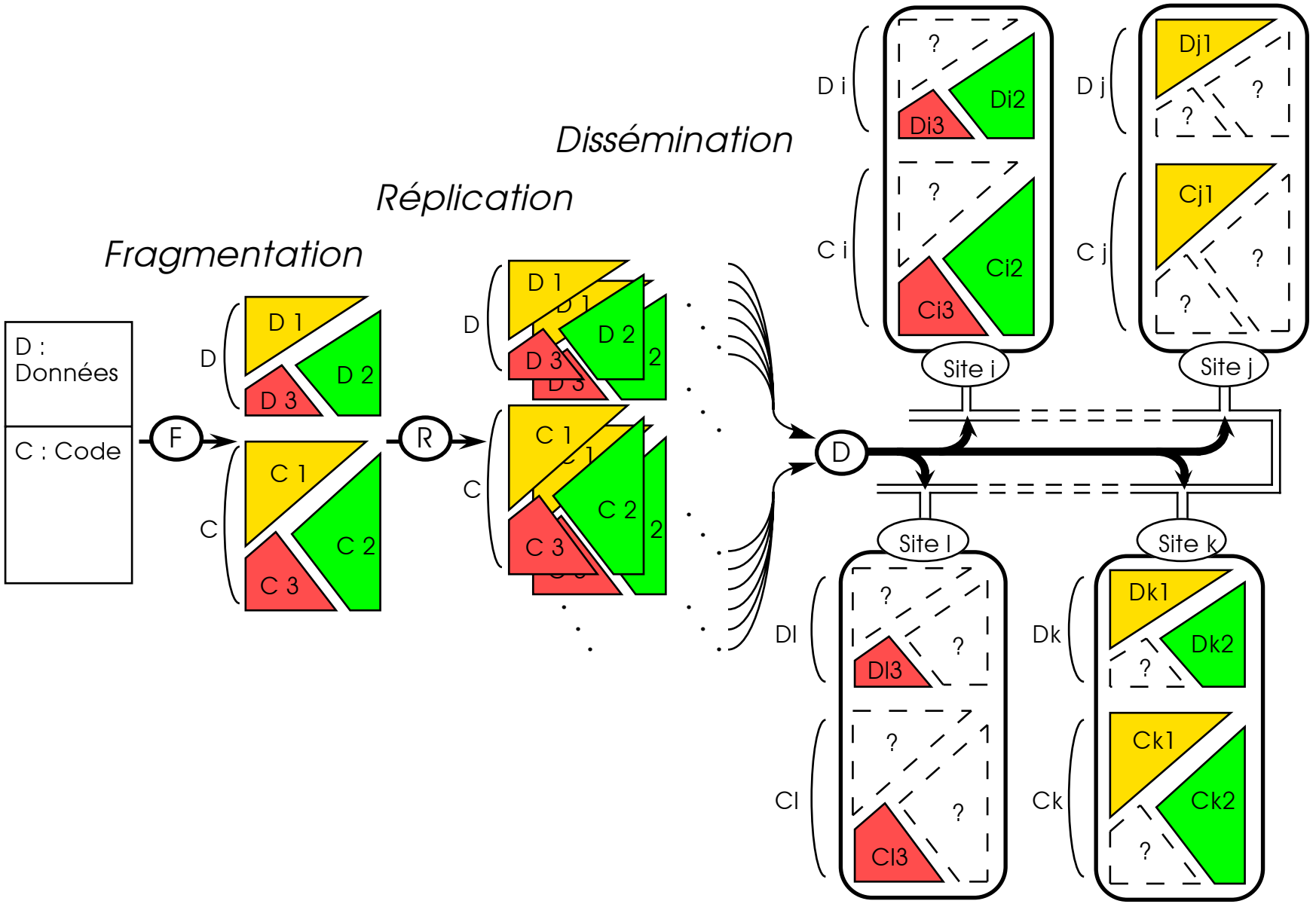
Idée :

Découper l'application en *activités coopérantes* telles que :

- les données transmises ou traitées soient des *fragments*
- les activités puissent être *répliquées*

*Le parallélisme et la répartition sont utilisés pour **cacher** l'information plutôt que pour réduire le temps de réponse.*

Traitement fragmenté



Intégrité pour les systèmes tolérant aux fautes avec plusieurs niveaux de criticité

Exemples : avions, trains, automobiles, véhicules spatiaux…

Intégration croissante ➟ le même système exécute des tâches très critiques et peu critiques

➟ une tâche peut être critique pendant certaines phases, et non critiques pendant d'autres

2 solutions:

- ➤ valider l'ensemble des tâches au plus haut niveau de criticité : … très coûteux
- ➤ construire des pare-feux pour éviter la contamination des tâches critiques par des tâches moins critiques : … effort de validation adapté au niveau de criticité

Idée :

- affecter différents niveaux d'intégrité aux différentes tâches (programmes + données), en fonction de leur criticité
- appliquer les règles de la politique de Biba :
  - observation d'un objet par un sujet : $(s_i, o_j, observer) \Rightarrow il(s_i) \leq il(o_j)$
  - modification d'un objet par un sujet : $(s_i, o_j, modifier) \Rightarrow il(o_j) \leq il(s_i)$
  - invocation d'un sujet par un autre sujet : $(s_i, s_j, invoquer) \Rightarrow il(s_j) \leq il(s_i)$

➢ aucun flot d'information fonctionnel ne devrait être interdit par les règles

➢ structuration de l'application

Exceptions : capteurs non fiables, interfaces H/M, dégradation de l'information...

Politique proposée (GUARDS)

- Basée sur la politique de Biba, adaptée au modèle “orienté-objet”
- appels de méthodes = messages de requête
- modes des méthodes : read (= flot d’information de l’appelé vers l’appelant),
  write (= flot d’information de l’appelant vers l’appelé)
  read-write (= flots dans les deux sens)
- 2 sortes d’objets (pour l’application comme pour l’OS):
  - single-level objects (SLOs): à chaque SLO est affecté un *niveau d’intégrité* en fonction de sa criticité et le SLO doit être validé en conséquence
  - multi-level objects (MLOs): un MLO peut s’exécuter à différents niveaux d’intégrité, mais il doit être validé à son plus haut niveau d’exécution :
    + un MLO ne peut pas provoquer de flot d’information entre différentes invocations
    + un MLO possède trois niveaux : max, min et courant
    + le niveau courant est initialisé à l’invocation, et il peut baisser en fonction des informations lues pendant l’exécution
    + le niveau min est initialisé à l’invocation, en fonction du niveau de l’appelant (méthodes read ou read-write)

Règles

un objet *O* appelle une méthode d'un objet O''

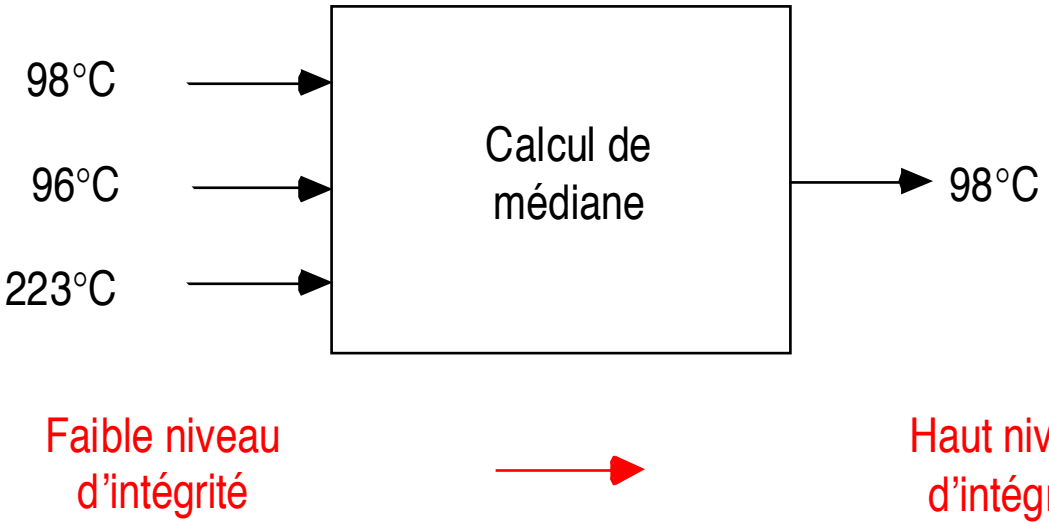
$l(O) = il(O)$ si *O* est un SLO

$l(O) = maxl(O)$ si *O* est un MLO

- *O'* est un SLO:
  - ◊ mode = read: interdit si $il(O') < l(O)$
  - ◊ mode = write: interdit si $l(O) < il(O')$
  - ◊ mode = read/write: interdit si $l(O) \neq il(O')$

- *O'* est un MLO:
  - ◊ mode = read: interdit si $maxl(O') < l(O)$
  - ◊ mode = write: interdit si $l(O) < minl(O')$
  - ◊ mode = read/write: interdit si $maxl(O') < l(O)$ ou $l(O) < minl(O')$

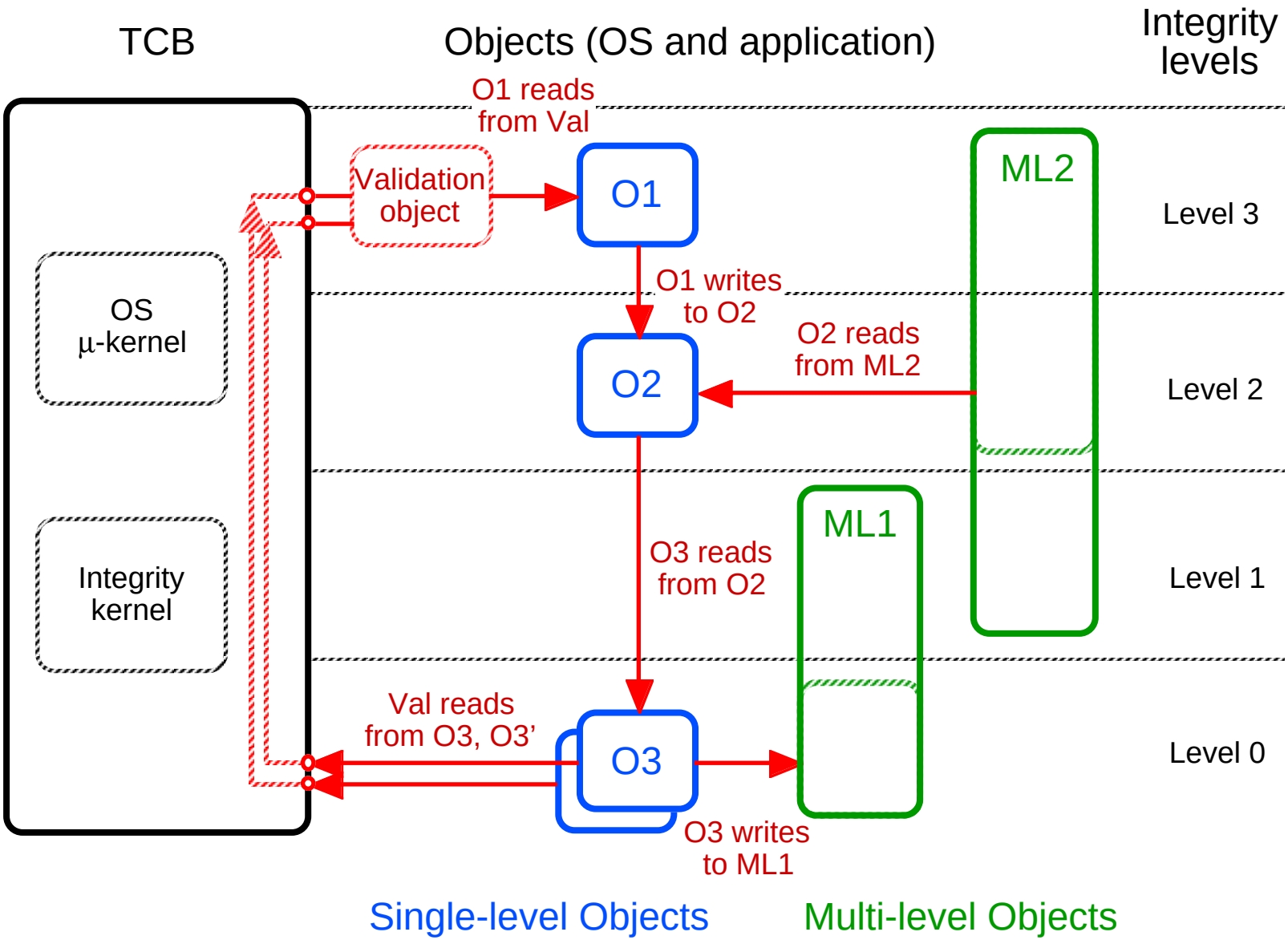
Exemple : trois capteurs de température non fiables



Validation Objects (VO) :

- pour lire des informations de niveaux d'intégrité inférieurs et y appliquer des mécanismes de tolérance aux fautes
- les VO sont validés au niveau de leurs sorties

Exemple


TCB
Objects (OS and application)
Integrity levels
O1 reads from Val
Validation object
O1
ML2
Level 3
OS µ-kernel
O1 writes to O2
O2 reads from ML2
O2
Level 2
Integrity kernel
ML1
O3 reads from O2
Level 1
Val reads from O3, O3'
O3
Level 0
O3 writes to ML1
Single-level Objects
Multi-level Objects

Conclusion

La tolérance aux fautes peut apporter de nouvelles solutions aux nouveaux problèmes de sécurité :

Exemples de problèmes:

- Ver de Robert T. Morris Jr. (novembre 1988) : 10 % des machines d'Internet contaminées, 98 % d'Internet bloqué pendant plusieurs jours
- défaillance d'InterNIC (juillet 97) : domaines .com, .net, .edu… inaccessibles pendant des durées de 4 à 24 h.
- possibilité d'attaque contre des tiers de confiance : commerce électronique, autorités de certifications…

Exemples de solutions :

- suppression des points durs par FRD
- projet ETERNITY (Ross Anderson, Cambridge, UK)
- sauver la civilisation face aux barbares (Yvo Desmedt, Un. Wisconsin)