L'horodatage sécurisé: état de l'art et applications

Axelle Apvrille
Axelle_Apvrille@storagetek.com
Toulouse Research & Development Center

Juin 2002

Plan de la présentation

- Introduction
  - Qu'est-ce que l'horodatage ? Comment sécuriser ?
  - A quoi ça sert ?
- Les modèles d'horodatage existants
  - Principe général de réalisation d'horodatage
  - Modèles liés, distribués, par arbre binaire
- Implémenter un protocole d'horodatage
  - TSP: fonctionnement, problèmes soulevés
- Application: V-WORM
  - Sécurisation de l'archivage électronique
  - Intégration de l'horodatage

Définition et problème

« ***A token that binds information about time with the bitstring*** » - Meelis Roos (1999)

- **Initialement, pas de notion de sécurité :**
  - **Rien ne prouve que l'horloge est correcte,**
  - **On peut forger un jeton d'horodatage.**
- **L'horodatage sécurisé, c'est :**
  - **Un jeton d'horodatage sécurisé par signature digitale,**
  - **Une horloge fiable fournissant l'heure.**

Cette présentation supposera qu'on dispose d'une horloge fiable

Horodater, quelle utilité ?

Les objectifs:

- Impossibilité d'antidater les documents
- Prouver sa bonne foi

Quelques exemples d'applications :

- Documents contractuels: banques, assurances, partenariats…
  - Le document a-t-il pu être fabriqué de toutes pièces ?
  - La date de signature a-t-elle été modifiée ?
  - Le document a-t-il été modifié depuis sa signature ?
- Horodatage des ordres de Bourse:
  - Connaître avec certitude la date des ordres d'achat / vente
- Médecine:
  - Horodatage des radios, résultats d'analyse
- Informatique:
  - Logs, mails…

Réaliser un jeton d'horodatage (1)

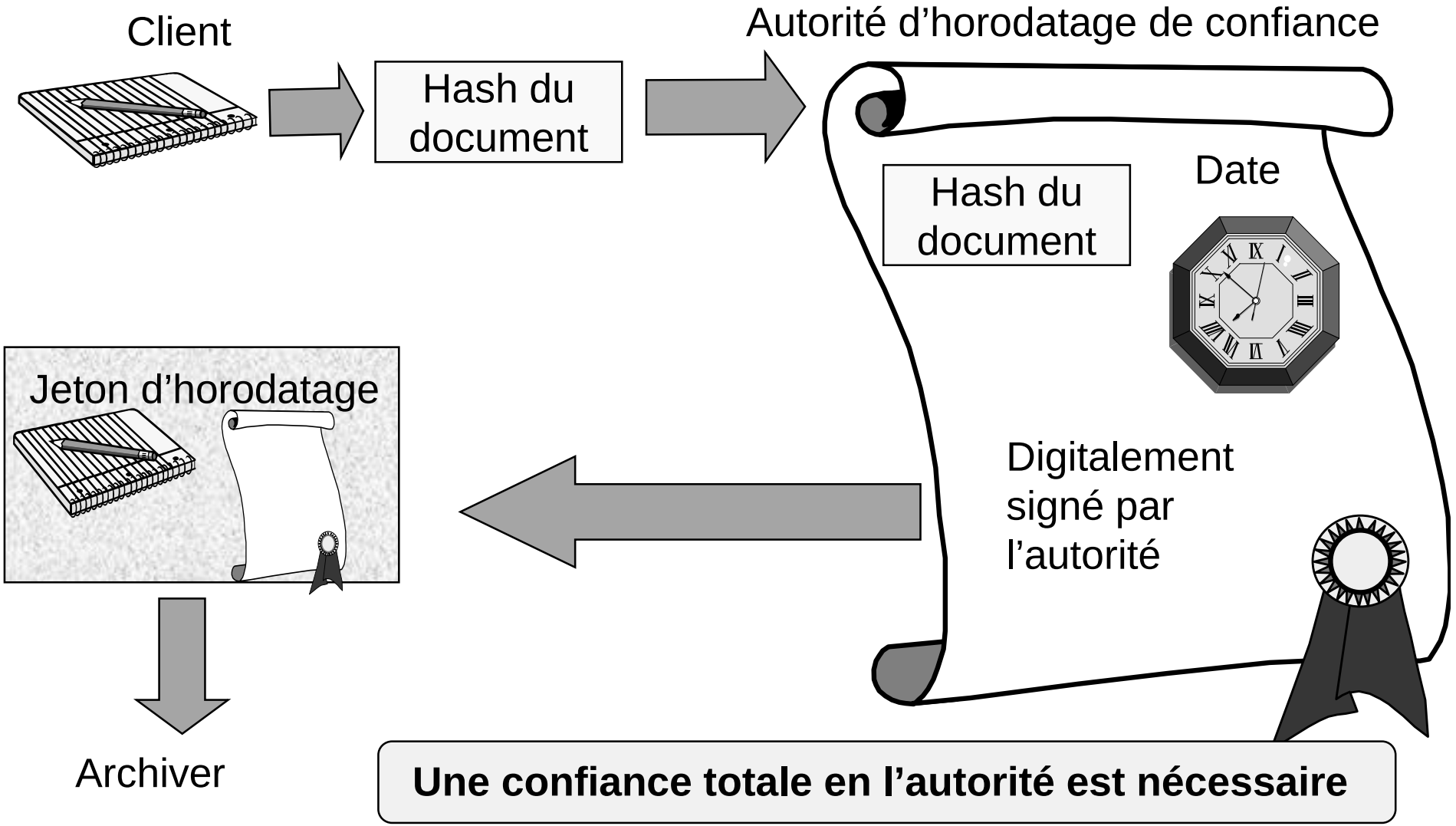
Les interlocuteurs en jeu sont:

- La personne qui **souhaite faire horodater** un document
  - Utilisateur ou « *Time stamp requestor* »
- Le service chargé de **réaliser l'horodatage**
  - Une tierce partie ou un service/module spécifique
  - « Time stamp service » (TSS) ou « Time stamp authority » (TSA)
- Une entité chargée de **vérifier l'horodatage**
  - Peut être l'utilisateur lui-même

Le processus d'horodatage est divisé en 2 parties:

- Un **protocole de réalisation d'horodatage**: génération du jeton d'horodatage, à partir de la requête de l'utilisateur,
- Un **protocole de vérification d'horodatage**: vérifie la validité du jeton d'horodatage.

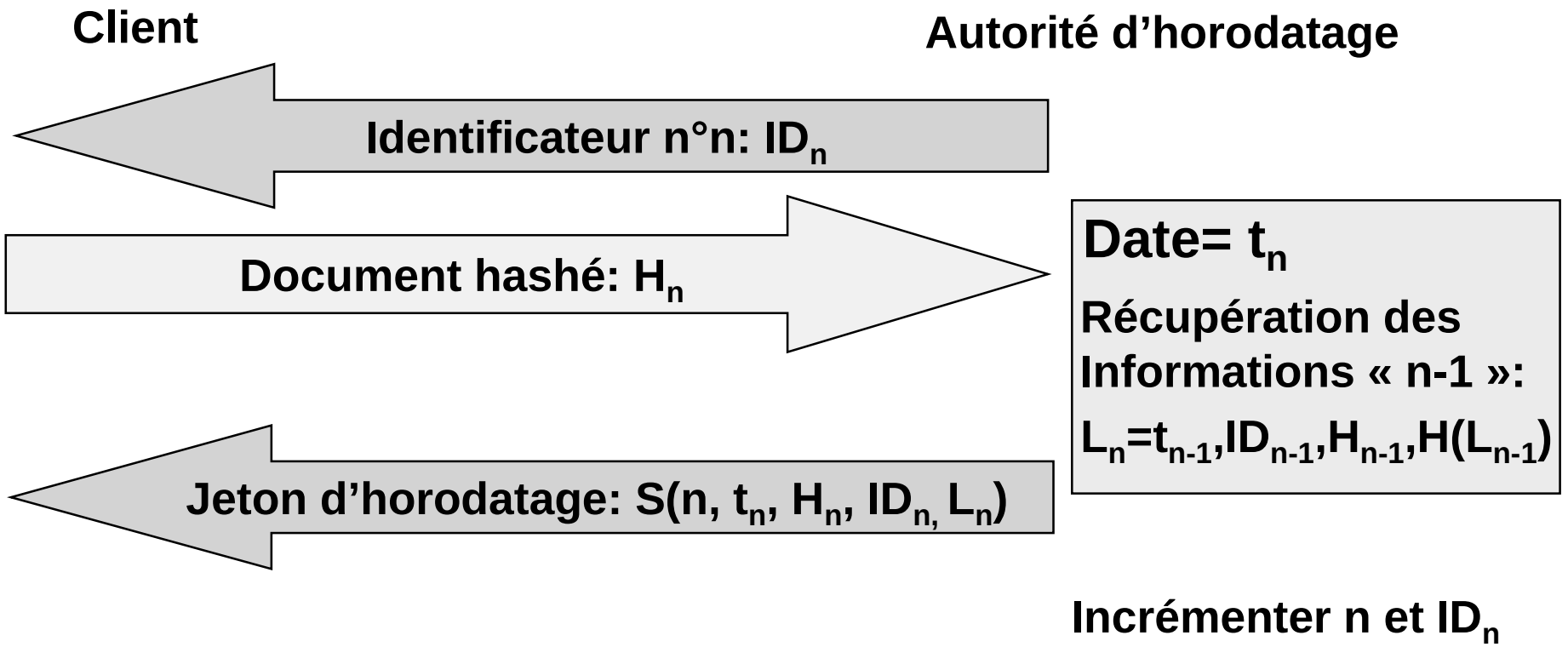
Réaliser un jeton d'horodatage (2)



Une confiance totale en l'autorité est nécessaire

Le modèle « lié » (1)

- Proposé par Haber et Stornetta (1991)
- Principe: la signature contient des informations issues du jeton d'horodatage précédent



Le modèle «lié » (2)

Extension:

- L'information de liaison peut être étendue à une séquence de jeton d'horodatage précédents.

Avantages:

- **Réduit la confiance à accorder à l'autorité,**
- Impossible (ou difficile) de modifier la date du jeton d'horodatage, car on ne maîtrise pas le flux des requêtes d'horodatage des clients,

Inconvénients:

- Rend la vérification de jeton plus délicate: il faut vérifier les informations de liens,
- Il faudra bien arrêter la vérification à un moment donné, donc ne résout pas complètement le problème de confiance…
- Valable s'il y a beaucoup de requêtes.

Le modèle « distribué » (1)

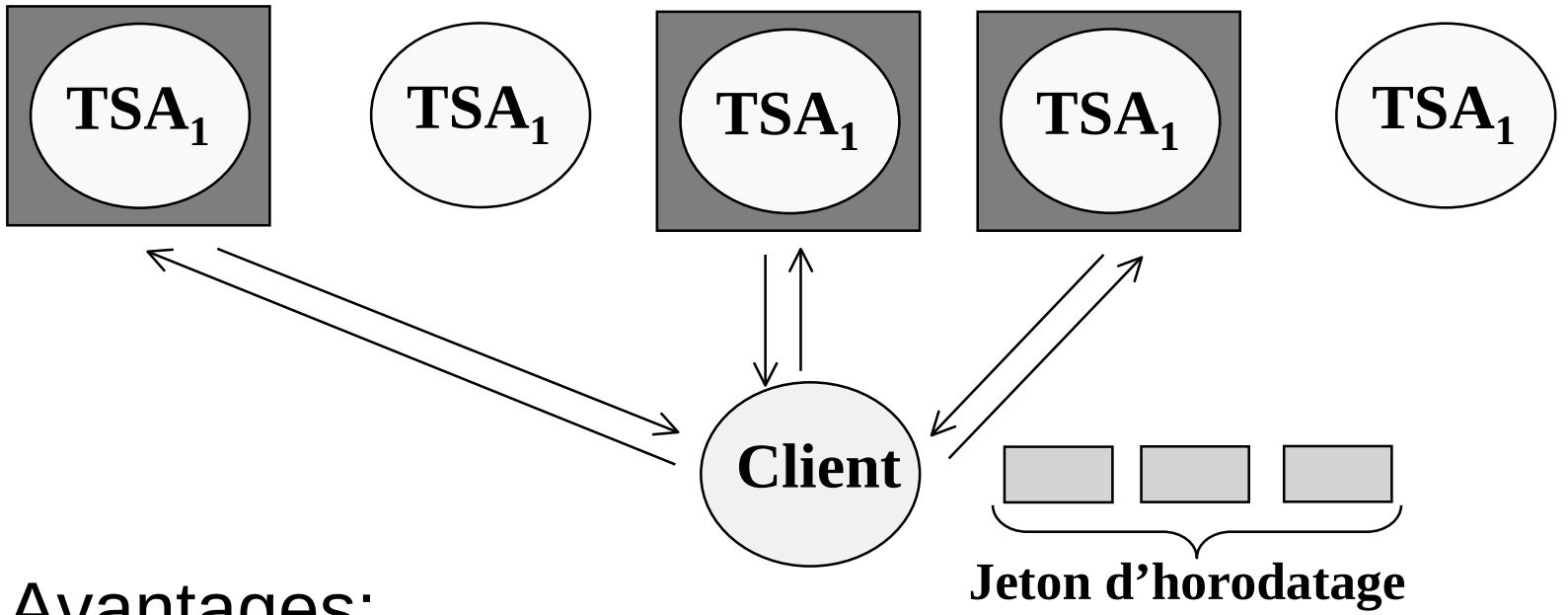
- **Proposé par Haber et Stornetta (1991)**
- **Principe: la requête d'horodatage est envoyée à plusieurs autorités différentes**

Tirer aléatoirement le nom de k autorités d'horodatage (TSA_k)

Envoyer le hash du document (H) à chacune des autorités

Chaque autorité renvoie $S(H,TSA_k,t)$. Le jeton d'horodatage est l'ensemble des réponses des autorités.

Le modèle distribué (2)

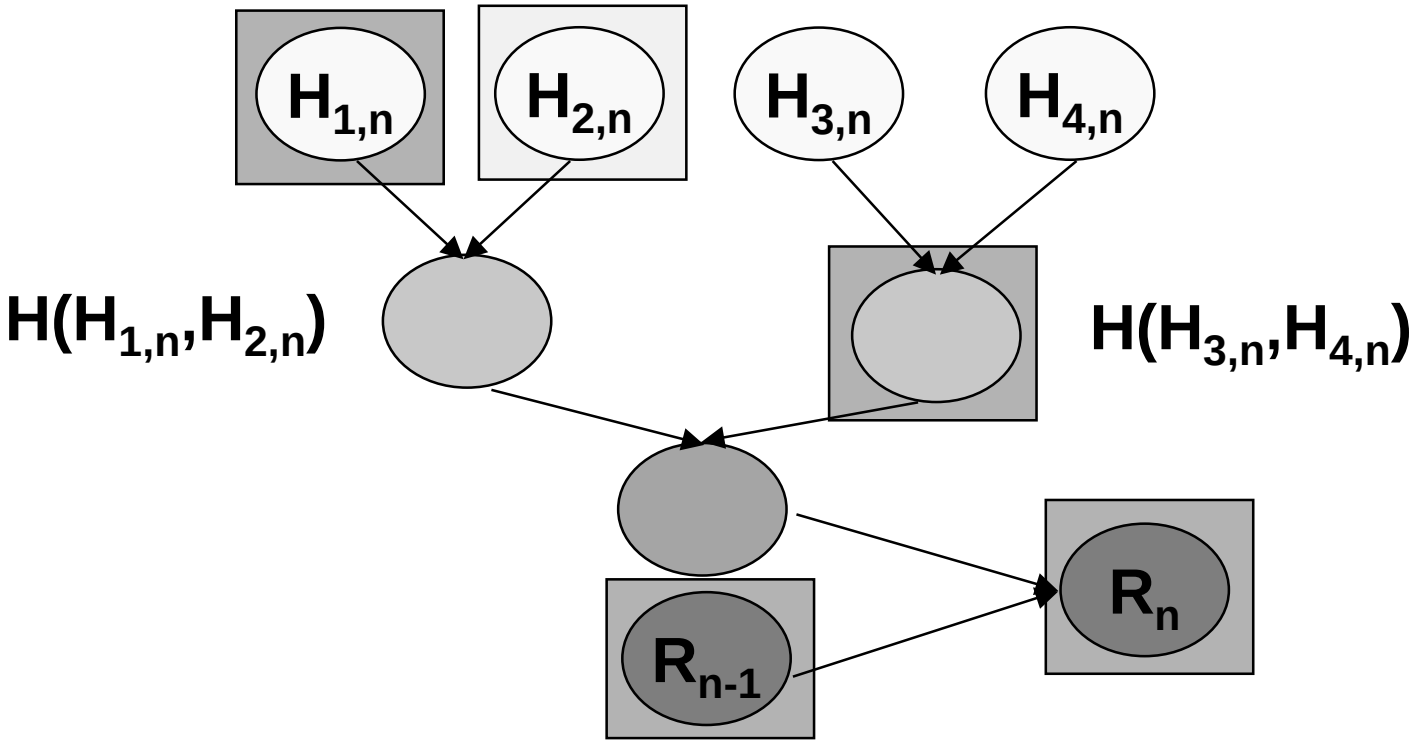


- Avantages:
  - Il est peu probable que les k autorités d'horodatage soient compromises,
- Inconvénients:
  - Vérification de l'horodatage simple mais « longue »: il faut vérifier auprès de chaque autorité.

Le modèle par arbre binaire (1)

- **Plusieurs propositions**: Benaloh & de Mare (1991), Haber & Stornetta (1992),
- **Principe**: toutes les requêtes d'horodatage reçues dans un « tour » sont combinées ensemble dans un arbre binaire.
- **Avantage**: la vérification ne nécessite pas la sauvegarde de toutes les requêtes du tour, mais seulement de $\log_2 n$.
- **Inconvénient**: intéressant uniquement si beaucoup de requêtes arrivent dans le même tour.

Le modèle par arbre binaire (2)



Le jeton d'horodatage (signé) est constitué de:
(1) **date,** (2) **chemin vers Rn,**
(3) **Rn et Rn-1**

Implémentation de protocoles d'horodatage

La « référence » actuelle est la RFC 3161: « *Internet X.509 Public Key Infrastructure Time Stamp Protocol (TSP)* » - Août 2001:

- Décrit les rôles du « client » et du TSA,
- Définit le format des messages `TimeStampReq` et `TimeStampResp`,
- Indépendant de tout algorithme,
- Propose des mécanismes de transport des messages: via e-mail, fichier, socket ou HTTP.

Son utilisation requiert :

- L'encodage des structures ASN.1 en DER,
- L'utilisation de CMS (PKCS#7 v1.5),
- La réalisation de signatures digitales avec un bi-clé dédié à l'horodatage: un seul `extendedKey` usage de `KeyPurposeID` {iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) kp (3) timestamping (8)}.

Les informations d'horodatage dans TSP

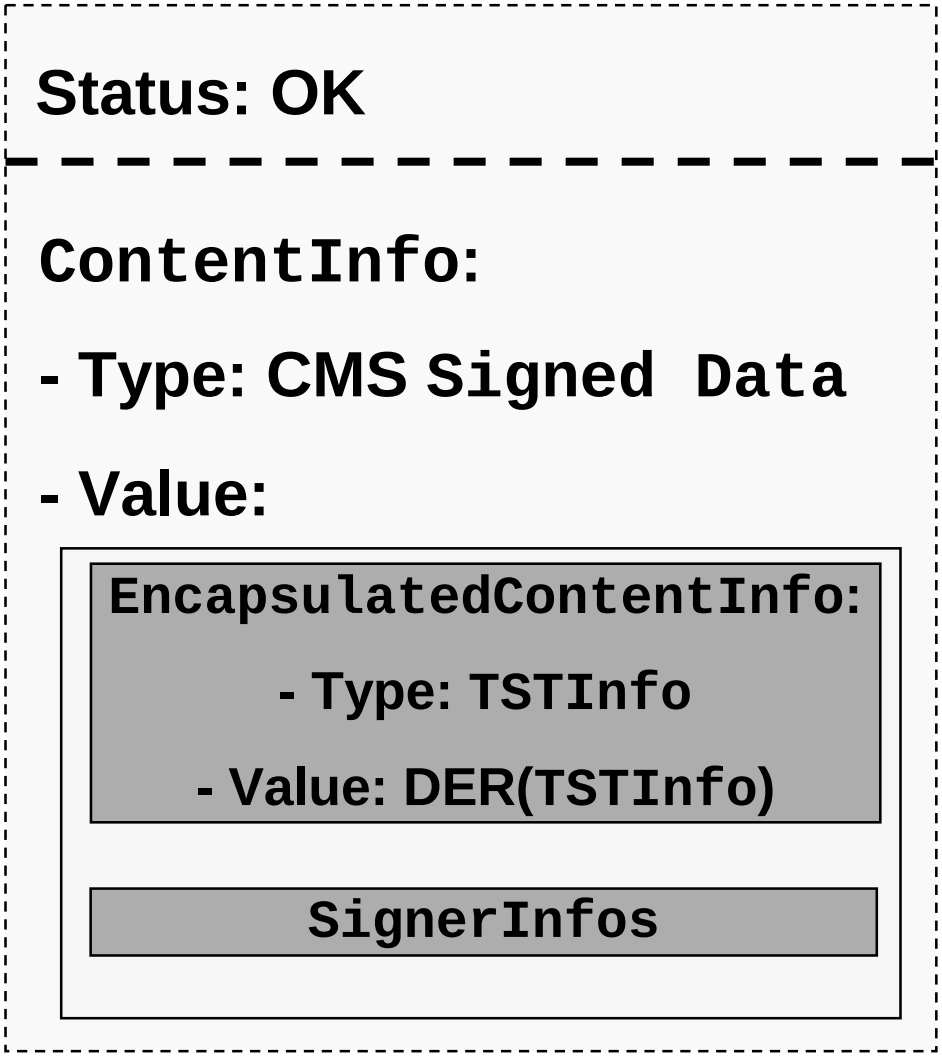
```
TSTInfo ::= SEQUENCE {
      version INTEGER { v1(1) },
      policy TSAPolicyId,
      messageImprint MessageImprint,
      serialNumber INTEGER,
      genTime GeneralizedTime,
      accuracy Accuracy OPTIONAL
      …
}
```

- **Mais où est donc la signature ?**

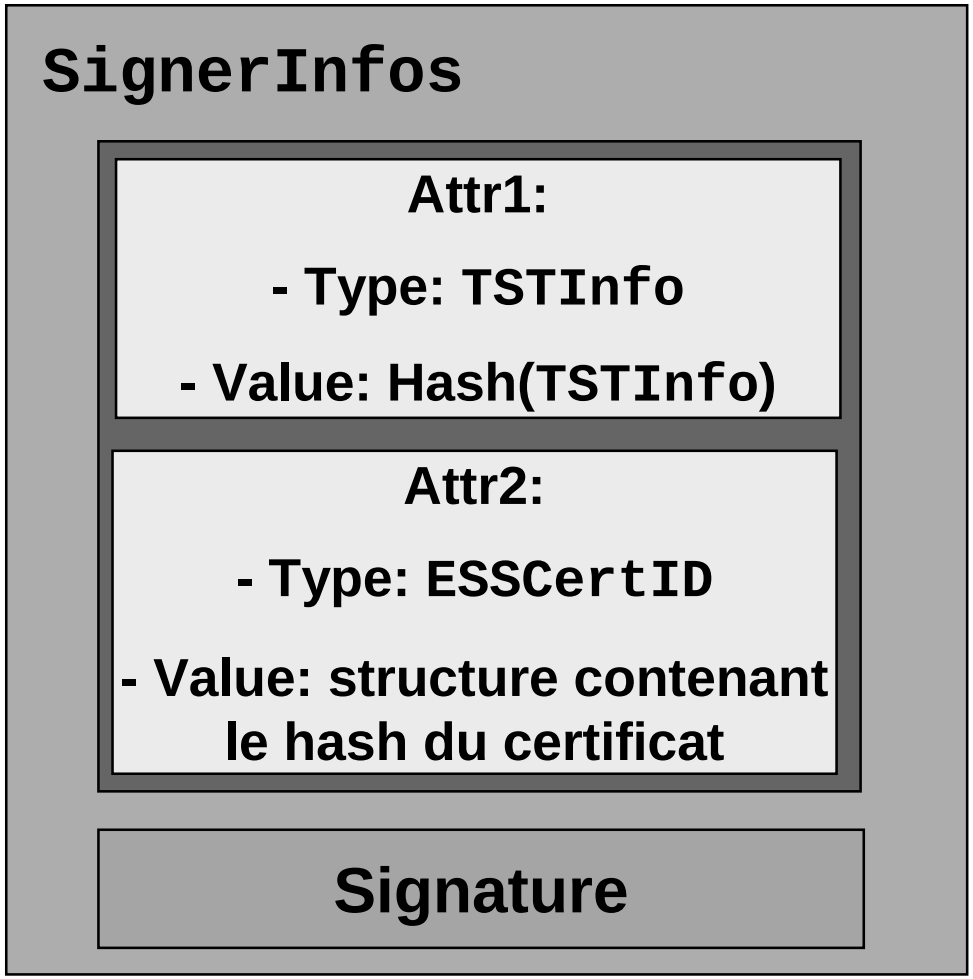
… (voir prochain transparent)

Encapsulation dans un message CMS

- Les informations d'horodatage sont indiquées comme contenu du message CMS,
- Toujours pas la signature en vue ?
  - => voir SignerInfos.



Signature du jeton d'horodatage



- La structure SignerInfos contient une liste d'attributs signés et d'attributs non signés
- Encodage des attributes signés (DER), puis signature.

Les problèmes posés par TSP

Génération d'un certificat pour l'autorité avec un extendedKeyUsage pour l'horodatage:

- Windows 2000 Certificate ServicesPKI : interface graphique pas assez précise, champ `extensionRequest` des requêtes PKCS#10 ignorés…
- OpenCA: à préciser dans les fichiers de configuration
- Exemple:
  - `basicConstraints=CA:FALSE`
  - `extendedKeyUsage=critical, timeStamping`

Nécessite la maîtrise de plusieurs standards:

- TSP
- CMS
- X.509
- ESS (RFC 2634)
- ASN.1 & DER…

TSP et XML

- Mauvaise lisibilité du format DER
  - DER a été conçu pour être compact, pas « lisible »,
  - Présence de plusieurs étapes d'encodage DER que les afficheurs ne comprennent pas.
- Solution: utiliser XML
  - Meilleure lisibilité, existence de nombreux outils… mais impose une réécriture de TSP,
  - Intégration avec XML-DSIG et ETSI XAdES,
  - TSP en cours de « traduction » vers XML chez StorageTek.

Application: le projet V-WORM

Objectifs:

- Archivage de longue durée (10-30 ans),
- Gestion de gros volumes,
- Assurer **l'intégrité** des données indépendamment du support: disque dur, bande magnétique…
- Les documents manuscrits sont **signés et datés**: faire de même pour les documents électroniques avec la notion **d'horodatage**.

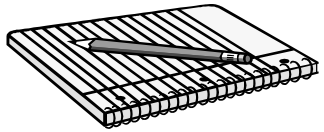
D'un WORM physique à un WORM virtuel

- WORM= ***Write Once Read Many,***
- Traditionnellement réservé aux supports physiquement non réinscriptibles,
- Il existe aussi des E-WORM, S-WORM.

V-WORM = S-WORM + crypto

Les étapes du « WORMage »

Découper en blocs



Hashage chaîné

Jeton d'horodatage

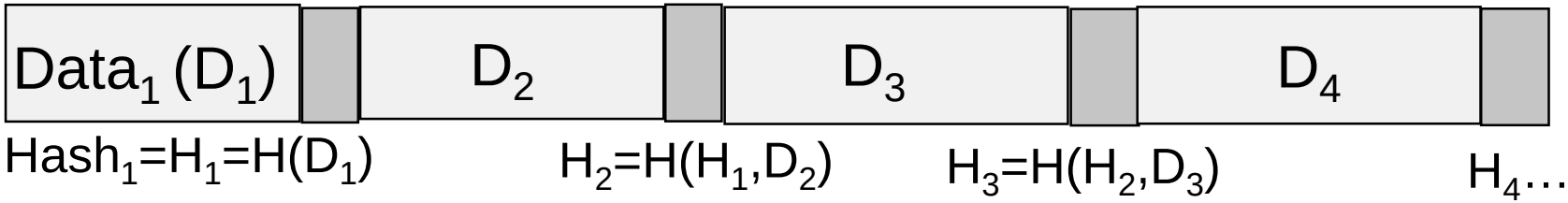
Jeton

Document horodaté à 10:12 am

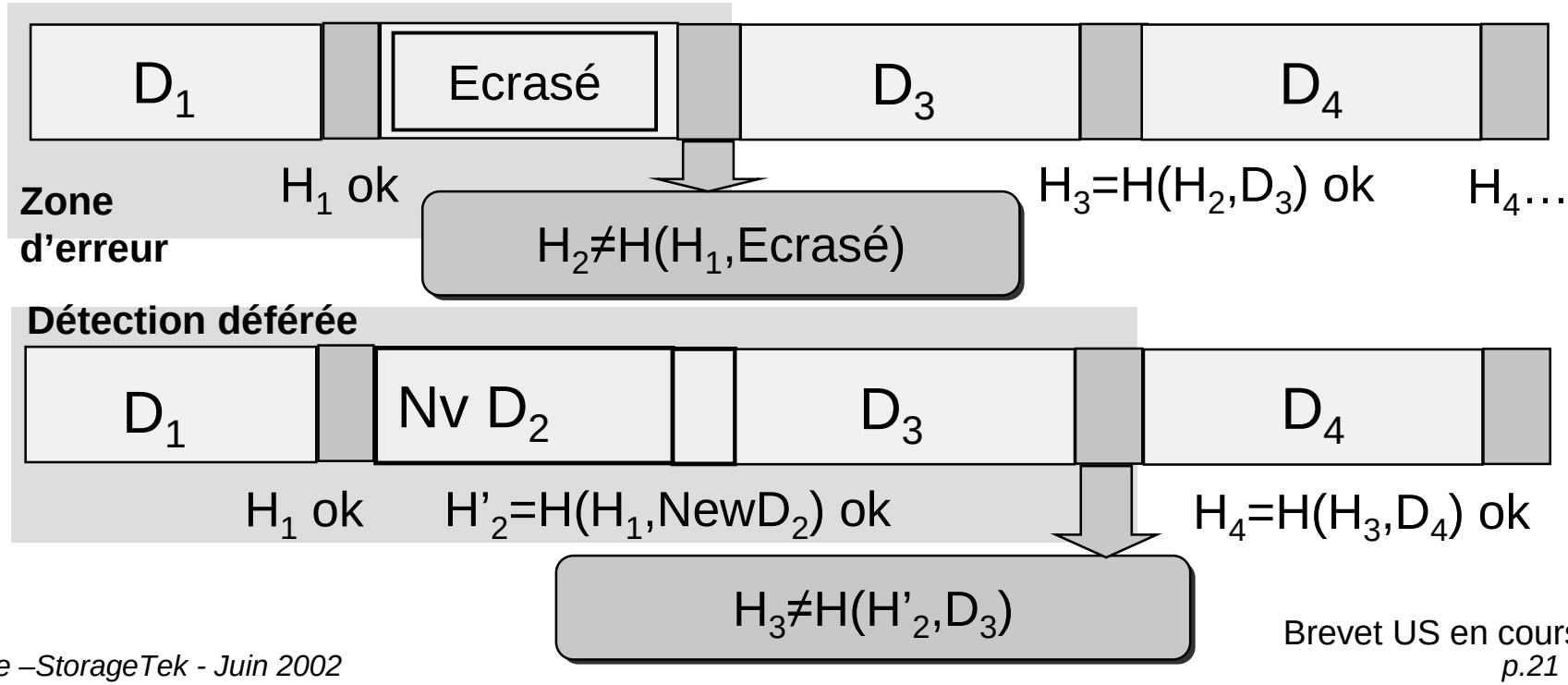
Signature

Hashage chaîné (1)

- Hashage chaîné : i>1, $H_i=H(H_{i-1},D_i)$



- Détection d'erreur :



Brevet US en cours

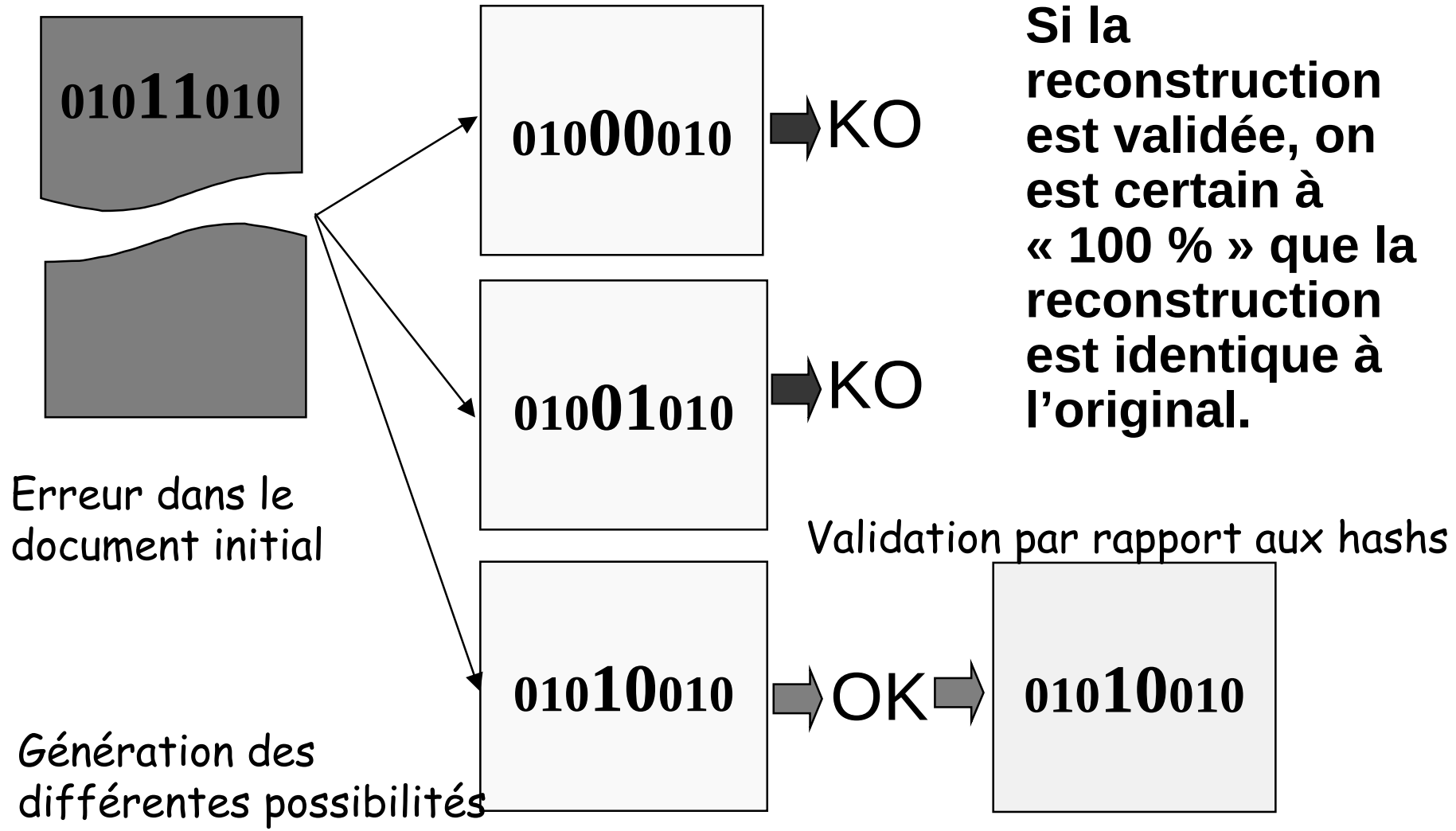
Hashage chaîné (2)

- Se prémunir contre:
  - Les erreurs accidentelles d'écriture,

    Peu de chances de modifier données & hash correctement.
  - Les attaques intentionnelles,

    Au pire, la signature électronique du jeton d'horodatage bloque tout.
- Eviter un horodatage trop intensif:
  - Une signature digitale est une opération « longue » par rapport au hashage,
  - A rapprocher de la notion d'horodatage par tour.

Récupération « validée »



Si la reconstruction est validée, on est certain à « 100 % » que la reconstruction est identique à l'original.

L’archivage… dans la durée

Problèmes à adresser:

En 10-30 ans, les supports changent !

- Ex: Difficile de trouver un ordinateur qui lit les disquettes 5 pouces ¼ actuellement…

Lors d’une migration, il faudra prouver que la copie est identique à l’original.

Les algorithmes, longueurs de clés évoluent.

Solutions possibles:

V-WORM est indépendant du support:
C’est le flux de données qui est sécurisé, pas le support.

Nécessite uniquement la Vérification des hashs.

Upgrader régulièrement.

Comparaison entre supports WORMs

	Irréversi-bilité physique	Réversi-bilité détectée	Intégrité de copie	Indépen-dance matérielle	Horo-datage sécurisé
WORM physique	Oui	Non	Oui, mais longue	Non	Non
WORM virtuel	Non	Oui	Oui	Oui	Oui

Références

- S. Haber and W.S. Stornetta, « ***How to Time Stamp a Digital Document*** », Journal of Cryptology, vol.3, n°2, pp99-111, 1991.
- A. Buldas, P. Laud, H. Lipmaa, J. Villemson, « ***Time-Stamping with Binary Linking Schemes*** », Advances in Cryptology, vol. 1462, pp 486-501, CRYPTO'98.
- D. Eastlake, J. Reagle, D. Solo, « ***(Extensible Markup Language) XML-Signature Syntax and Processing*** », Network working group, RFC 3275, Mars 2002.
- ETSI Technical Committee Security (SEC), « ***XML Advanced Electronic Signatures (XAdES)*** », TS 101 903 v1.1.1, Fév. 2002.
- A. Apvrille, V. Girier « ***XML Security Time Stamping Protocol*** », à paraître à ISSE'02 (Oct. 2002).

Merci !