Groupe de travail Réseau
Request for Comments : 4998
Catégorie : sur la voie de la normalisation
Traduction Claude Brière de L'Isle

T. Gondrom, Open Text Corporation
R. Brandner, InterComponentWare AG
U. Pordesch, Fraunhofer Gesellschaft
août 2007

Syntaxe d'enregistrement de preuve (ERS)

Statut du présent mémoire
Le présent document spécifie un protocole de l'Internet sur la voie de la normalisation pour la communauté de l'Internet, et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "Protocoles officiels de l'Internet" (STD 1) pour voir l'état de normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.
(La présente traduction incorpore l'erratum 7411)

Résumé
Dans de nombreux scénarios, les utilisateurs doivent être capables de prouver l'existence et l'intégrité de données, y compris de données signées numériquement, d'une façon commune et reproductible sur une longue période, éventuellement d'une durée indéterminée. Le présent document spécifie la syntaxe et le traitement de l'enregistrement de preuve (*Evidence Record)* structure conçue pour prendre en charge la non répudiation à long terme de l'existence de données.

Table des Matières

1. Introduction

1.1 Motivation

Dans de nombreux domaines d'application des échanges de données électroniques, une preuve non révocable de l'existence de données numériques doit être possible. Dans certains cas, cette preuve doit survivre longtemps. Un important exemple est celui des données signées numériquement. Des signatures numériques peuvent être utilisées pour démontrer l'intégrité des données et effectuer l'authentification de la source. Dans certains cas, des données signées numériquement doivent être archivées pendant 30 ans ou plus. Cependant, la fiabilité des signatures numériques sur de longues périodes n'est pas absolue. Durant la période d'archivage, les algorithmes de hachage et les algorithmes de clé publique peuvent devenir faibles ou les certificats peuvent devenir invalides. Ces événements mettent en jeu la confiance qu'on peut accorder aux données signées numériquement après de nombreuses années en augmentant la probabilité que des falsifications puissent être créées. Pour éviter de perdre les propriétés de sécurité désirées déduites des signatures numériques, il est nécessaire de prouver que les données signées numériquement existaient avant un tel événement critique. Ceci peut être accompli en utilisant un horodatage. Cependant, certains horodatages s'appuient sur des mécanismes qui vont être soumis aux mêmes problèmes. Pour les résoudre, les horodatages sont renouvelés simplement en obtenant un nouvel horodatage qui couvre les données originales et leurs horodatages avant la compromission des mécanismes utilisés pour générer les horodatages. Le présent document fournit une syntaxe pour prendre en charge le renouvellement périodique des horodatages.

Il est nécessaire de normaliser les formats et les procédures de traitement des données pour ces horodatages afin d'être capable de vérifier et communiquer la preuve de préservation. Une première approche a été faite par l'IETF dans la [RFC3126], où un attribut facultatif Horodatage d'archive était spécifié pour être intégré dans les signatures en accord avec la syntaxe de messages cryptographique (CMS, *Cryptographic Messages Syntax*) [RFC3852].

La syntaxe d'enregistrement de preuve (ERS, *Evidence Record Syntax*) élargit et généralise cette approche pour les données de tout format et prend en compte les exigences de service d'archive à long terme [RFC4810] -- en particulier, le traitement de grands ensembles d'objets de données. ERS spécifie une syntaxe pour un enregistrement de preuve *(EvidenceRecord)* qui contient un ensemble d'horodatages d'archive *(archive timestamping)* et des données supplémentaires. Cet enregistrement de preuve peut être mémorisé séparément de données archivées, comme un fichier, ou intégré dans les données archivées, c'est-à-dire, comme un attribut. ERS spécifie aussi les processus pour la génération et la vérification des enregistrements de preuve. L'Appendice A décrit l'intégration et l'utilisation d'un enregistrement de preuve dans le contexte de messages signés et enveloppés conformément à la syntaxe de messages cryptographique (CMS). ERS ne spécifie pas de protocole pour interagir avec un système d'archive à long terme. La spécification du protocole d'archive à long terme en cours de développement par le groupe de travail LTANS de l'IETF traite cette interface.

1.2 Vue générale et exigences

ERS est conçu poour satisfaire les exigences des structures de données établies dans la [RFC4810].

La base de l'ERS est l'horodatage d'archive qui peut couvrir un seul objet de données (comme le fait un horodatage conforme à la RFC3161) ou peut couvrir un groupe d'objets de données. Les groupes d'objets de données sont traités en utilisant des arborescences de hachage, décrites par Merkle [MER1980], combinées avec un horodatage. Les feuilles de l'arborescence de hachage sont les valeurs de hachages des objets de données dans un groupe. Un horodatage n'est exigé que pour le hachage racine de l'arborescence de hachage. La suppression d'un objet de données dans l'arborescence n'a pas d'influence sur la démontrabilité des autres. Pour tout objet de données particulier, l'arborescence de hachage peut être réduite à quelques ensembles de valeurs de hachage, qui sont suffisants pour prouver l'existence d'un seul objet de données. De façon similaire, l'arborescence de hachage peut être réduite pour prouver l'existence d'un groupe de données, pourvu que tous les membres du groupe de données aient le même nœud parent dans l'arborescence de hachage. Les horodatages d'archive sont composés d'une arborescence de hachage réduite facultative et d'un horodatage.

Un enregistrement de preuve peut contenir plusieurs horodatages d'archive. Pour la génération de l'horodatage d'archive initial, les objets de données à horodater doivent être déterminés. Selon le contexte, cela pourrait être un fichier ou un groupe d'objets de données consistant en plusieurs fichiers, comme un document et sa signature numérique associée.

Avant que les algorithmes de chiffrement utilisés dans l'horodatage d'archive deviennent faibles ou que les certificats d'horodatage deviennent invalides, les horodatages d'archive doivent être renouvelés en générant un nouvel horodatage d'archive. (Note : les informations sur l'affaiblissement des propriétés de sécurité de la clé publique et des algorithmes de hachage, ainsi que le risque de compromission des clés privées des unités d'horodatage, doivent être obervées attentivement par le fournisseur d'archive à long terme ou le propriétaire des objets de données lui-même. Ces informations devraient être rassemblées par des moyens "hors bande" et sortent du domaine d'application du présent document.) ERS distingue deux

façons de renouveler un horodatage d'archive : le renouvellement d'horodatage et le renouvellement d'arborescence de hachage.

Selon les conditions, le type respectif de renouvellement est exigé : le renouvellement d'horodatage est nécessaire si la clé privée d'une unité d'horodatage a été compromise, ou si un algorithme asymmetrique ou un algorithme de hachage utilisé pour la génération des horodatages n'est plus sûr pour la taille de clé donnée. Si l'algorithme de hachage utilisé pour construire les arborescences de hachage dans l'horodatage d'archive perd ses propriétés de sécurité, le renouvellement d'arborescence de hachage est nécessaire.

Dans le cas d'un renouvellement d'horodatage, l'horodatage d'un horodatage d'archive doit être haché et horodaté par un nouvel horodatage d'archive. Ce mode de renouvellement ne peut être utilisé que quand il n'est pas nécessaire d'accéder aux objets de données archivés couverts par l'horodatage. Par exemple, cette forme simple de renouvellement est suffisante si l'algorithme de clé publique de l'horodatage va perdre sa sécurité ou si le certificat d'autorité de l'horodatage est sur le point d'arriver à expiration. Ceci est très efficace, en particulier, si l'horodatage d'archive est fait par un système ou service d'archivage, qui met en œuvre une gestion centrale des horodatages d'archive.

Le renouvellement d'horodatage n'est pas suffisant si l'algorithme de hachage utilisé pour construire l'arborescence de hachage d'un horodatage d'archive devient non sûr. Dans le cas du renouvellement de l'arborescence de hachage, toutes les données de preuve doivent être accessibles et horodatées. Cela inclut non seulement les horodatages mais aussi les horodatages d'archive complets et les objets de données archivés couverts par les horodatages, qui doivent être hachés et horodatés à nouveau par un nouvel horodatage d'archive.

1.3 Terminologie

Objet de données archivées : unité de données qui est archivée et doit être préservée pendant longtemps par le service d'archive à long terme.

Groupe d'objets de données archivées : ensemble de deux ou plus objets de données, qui sont ensemble pour une raison quelconque. Par exemple, un fichier de document et un fichier de signature pourraient être un groupe d'objets de données archivées, qui représente des données signées.

Horodatage d'archive : un horodatage et normalement des listes de valeurs de hachage, qui permettent la vérification de l'existence de plusieurs objets de données à un certain moment. (Dans sa variante la plus simple, quand il couvre seulement un objet, il peut ne consister qu'en l'horodatage.)

Chaîne d'horodatages d'archive : partie d'une séquence d'horodatages d'archive, c'est une séquence ordonnée dans le temps d'horodatages d'archive, où chaque horodatage d'archive préserve la non répudiation de l'horodatage d'archive précédent, même après que le précédent horodatage d'archive est devenu invalide. La non répudiation globale est maintenue jusqu'à ce que le nouvel horodatage d'archive devienne lui-même invalide. Le processus de génération d'une telle chaîne d'horodatages d'archive est appelé le renouvellement d'horodatage.

Séquence d'horodatage d'archive : partie de l'enregistrement de preuve, c'est une séquence de chaînes d'horodatages d'archive, où chaque chaîne d'horodatages d'archive préserve la non répudiation des chaînes précédentes d'horodatages d'archive, même après que l'algorithme de hachage utilisé au sein de l'arborescence de hachage du précédent horodatage d'archive est devenu faible. La non répudiation est préservée jusqu'à ce que le dernier horodatage d'archive de la dernière chaîne soit invalide. Le processus de génération d'une telle séquence d'horodatage d'archive est appelée le renouvellement d'arborescence de hachage.

Preuve : information qui peut être utilisée pour résoudre un conflit sur divers aspects de l'authenticité des objets de données archivés.

Enregistrement de preuve : collection de preuves compilée pour un ou plusieurs objets de données archivés au fil du temps. Un enregistrement de preuve inclut tous les horodatages d'archive (au sein des structures de chaînes d'horodatages d'archive et de séquences d'horodatages d'archive) et des données de vérification supplémentaires, comme des certificats, informations de révocation, ancres de confiance, détails de politique, informations de rôle, etc.

Service d'archive à long terme (LTA, *Long-Term Archiv*) : service chargé de la préservation des données pendant de longues périodes, incluant la génération et la collecte des preuves, la mémorisation des objets de données et des preuves archivés, etc.

Arborescence de hachage réduite : processus de réduction d'une arborescence de hachage de Merkle [MER1980] en une liste de listes de valeurs de hachage. C'est la base de la mémorisation de preuve pour un seul objet de données.

Horodatage : confirmation cryptographiquement sûre générée par une autorité d'horodatage (TSA, *Time Stamping Authority*). La [RFC3161] spécifie une structure pour les horodatages et un protocole pour communiquer avec une TSA. En outre, d'autres structures et protocoles de données peuvent aussi être appropriés, par exemple, comme ceux définis dans [ISO-18014-1], [ISO-18014-2], [ISO-18014-3], et [ANSI.X9-95].

Un horodatage d'archive se rapporte à un objet de données, si la valeur de hachage de cet objet de données fait partie de la première liste de valeurs de hachage de l'horodatage d'archive. Un horodatage d'archive se rapporte à un groupe d'objets de données si il se rapporte à chaque objet de données du groupe et à aucun autre objet de données. Une chaîne d'horodatages d'archive se rapporte à un objet de données / groupe d'objets de données si son premier horodatage d'archive se rapporte à cet objet de données/groupe d'objets de données. Une séquence d'horodatages d'archive se rapporte à un objet de données / groupe d'objets de données si sa première chaîne d'horodatages d'archive se rapporte à cet objet de données/groupe d'objets de données.

1.4 Conventions utilisées dans ce document

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" en majuscules dans ce document sont à interpréter comme décrit dans le BCP 14, [RFC2119].

2. Identification et références

2.1 Définition de module ASN.1

Comme de nombreux compilateurs ASN.1 prennent encore en charge la syntaxe 1988, la présente norme offre la prise en charge de deux versions d'ASN.1 , l'ASN.1 1997 et l'ASN.1 1988. (Pour la spécification de l'ASN.1, se référer à [X.208], [X.209], [X.680] et [X.690].) La présente spécification définit les deux modules ASN.1, un pour l'ASN.1 conforme à l'ASN.1 1988 et l'autre dans la syntaxe de l'ASN.1 1997. Selon la version de syntaxe de la mise en œuvre de compilateur, on peut utiliser les importations pour la syntaxe conforme à l'ASN.1 1988 ou celles pour la syntaxe de l'ASN.1 1997. L'Appendice à ce document donne les deux modules ASN.1 complets. Si il y a un conflit entre les deux modules, celui de 1988 l'emporte.

2.1.1 Définition de module ASN.1 pour la syntaxe ASN.1 1988

Début du module ASN.1 1988

```
ERS {iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) ltans(11) id-mod(0) id-mod-ers88(2)
      id-mod-ers88-v1(1) }
ÉTIQUETTES IMPLICITES DE DÉFINITIONS ::=
DÉBUT
```

2.1.2 Définition de module ASN.1 pour la syntaxe ASN.1 1997

Début du module ASN.1

```
ERS {iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) ltans(11) id-mod(0) id-mod-ers(1) id-
      mod-ers-v1(1) }
ÉTIQUETTES IMPLICITES DE DÉFINITIONS ::=
DÉBUT
```

2.2 Importations et exportations ASN.1

La spécification exporte toutes les définitions et importe diverses définitions. Selon la version de syntaxe ASN.1 de la mise en œuvre, on peut utiliser les importations pour la syntaxe conforme à l'ASN.1 1988 ci-dessous ou les importations pour la syntaxe ASN.1 1997 du paragraphe 2.2.2.

2.2.1 Importations et exportations conformes à l'ASN.1 1988

```
-- EXPORTE TOUT --

  IMPORTE

-- Importe de la RFC 3852 Syntaxe de message cryptographique
ContentInfo, Attribute
   DE CryptographicMessageSyntax2004 -- DE [RFC3852]
      { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0) cms-2004(24) }

-- Importe de la [RFC3280], Appendice A.1
AlgorithmIdentifier
   DE PKIX1Explicit88
     { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) mod(0) pkix1-explicit(18) } ;
```

2.2.2 Importations et exportations conformes à l'ASN.1 1997

```
-- EXPORTE TOUT --

IMPORTE

-- Importe de PKCS-7
  ContentInfo
   DE PKCS7
        {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-7(7) modules(0)}

-- Importe de AuthenticationFramework
AlgorithmIdentifier
  DE AuthenticationFramework
        {joint-iso-itu-t ds(5) module(1) authenticationFramework(7) 4}

-- Importe de InformationFramework
Attribute
  DE InformationFramework
        {joint-iso-itu-t ds(5) module(1) informationFramework(1) 4}
;
```

2.3 Identification LTANS

Le présent document définit la racine d'arborescence d'identifiant d'objet LTANS.

Racine d'arborescence d'identifiant d'objet LTANS

```
IDENTIFIANT D'OBJET ltans ::=
        { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) ltans(11) }
```

3. Enregistrement de preuve

Un enregistrement de preuve est une unité de données, qui peut être utilisée pour prouver l'existence d'un objet de données archivées ou d'un groupe d'objets de données archivées à un certain moment. L'enregistrement de preuve contient des horodatages d'archive, générés durant une longue période d'archivage et éventuellement des données utiles pour leur validation. Il est possible de mémoriser cet enregistrement de preuve séparément des objets de données archivées ou de l'intégrer dans les données elles-mêmes. Pour les types de données, les données signées et les données enveloppées de l'intégration CMS sont spécifiées à l'Appendice A.

3.1 Syntaxe

L'enregistrement de preuve a la syntaxe ASN.1 suivante :

Enregistrement de preuve ASN.1

```
EvidenceRecord ::= SEQUENCE {
  version               ENTIER { v1(1) } ,
  digestAlgorithms        SEQUENCE DE AlgorithmIdentifier,
  cryptoInfos             [0] CryptoInfos FACULTATIF,
  encryptionInfo          [1] EncryptionInfo FACULTATIF,
  archiveTimeStampSequence  ArchiveTimeStampSequence
  }

CryptoInfos ::= TAILLE DE SEQUENCE (1..MAX) DE Attribute
```

Les champs ont la signification suivante :

"version" indique la version de la syntaxe, pour la compatibilité avec de futures révisions de cette spécification et pour la distinguer de versions antèrieures non conformes ou propriétaires de ERS. La valeur 1 indique la présente spécification. Les valeurs inférieures indiquent qu'une version antérieure de ERS a été utilisée. Une mise en œuvre conforme à la présente spécification DEVRAIT rejeter une valeur de version inférieure à 1.

"digestAlgorithms" est une séquence de tous les algorithmes de hachage utilisés pour hacher l'objet de données sur la période d'archivage. C'est l'union de toutes les valeurs de digestAlgorithm provenant de ArchiveTimestamps contenues dans le EvidenceRecord. L'ordre des valeurs n'a pas d'importance.

"cryptoInfos" permet la mémorisation des données utiles dans la validation de la archiveTimeStampSequence. Cela pourrait inclure de possibles ancres de confiance, certificats, informations de révocation, ou la définition actuelle de la convenance d'algorithmes de chiffrement, passés et présents (par exemple,, RSA à 768 bits valide jusqu'en 1998, RSA à 1024 bits valide jusqu'en 2008, SHA1 valide jusqu'en 2010). Ces éléments peuvent être ajoutés en fonction de la politique utilisée. Comme ces données ne sont pas protégées dans un horodatage, les données devraient être vérifiables par d'autres mécanismes. Une telle vérification sort du domaine d'application du présent document.

"encryptionInfo" contient les informations nécessaires pour prendre en charge le traitement du contenu chiffré. Pour la discussion de la syntaxe, voir au paragraphe 6.1.

"ArchiveTimeStampSequence" est une séquence de ArchiveTimeStampChain, décrit à la Section 5.

Si les objets de données d'archive n'étaient pas chiffrés avant de générer les horodatages d'archive mais si une preuve de non répudiation est nécessaire pour des objets de données non chiffrés, le champ facultatif encryptionInfos contient les données nécessaires pour rechiffrer de façon non ambiguë les objets de données. Si il est omis, cela signifie que les objets de données ne sont pas chiffrés ou qu'une preuve de non répudiation n'est pas exigée pour les données non chiffrées. Voir les détails à la Section 6.

3.2 Génération

La génération d'un enregistrement de preuve peut être décrite comme suit :
1. Choisir un objet ou groupe d'objets de données à archiver.
2. Créer l'horodatage d'archive initial (voir la Section 4, "Horodatage d'archive").
3. Rafraîchir l'horodatage d'archive quand nécessaire, par renouvellement d'horodatage ou par renouvellement d'arborescence de hachage (voir la Section 5).

Le processus de génération dépend de si les horodatages d'archive sont générés, mémorisés, et gérés par une instance centralisée. Dans le cas d'une gestion centrale, il est possible de collecter de nombreux objets de données, construire des arborescences de hachage, les mémoriser, et les réduire ensuite. Dans le cas de génération locale, il pourrait être plus facile de générer un simple horodatage d'archive sans constuire d'arborescence de hachage. Cela peut être accompli en omettant le champ reducedHashtree dans ArchiveTimestamp. Dans ce cas, le ArchiveTimestamp couvre un seul objet de données. En utilisant cette approche, il est possible de "convertir" des horodatages existants en ArchiveTimestamps pour renouvellement.

3.3 Vérification

La vérification d'un enregistrement de preuve global peut être décrite comme suit :
1. Choisir un objet de données ou groupe d'objets de données archivées.
2. Re-chiffrer l'objet de données/groupe d'objet de données, si le champ Chiffrement est utilisé (voir les détails à la Section 6).
3. Vérifier la séquence d'horodatages d'archive (voir les détails aux Sections 4 et 5).

4. Horodatage d'archive

Un horodatage d'archive est un horodatage et un ensemble de listes de valeurs de hachage. Les listes de valeurs de hachage sont générées par la réduction d'une arborescence de hachage ordonnée de Merkle [MER1980]. Les feuilles de cette arborescence de hachage sont les valeurs de hachage des objets de données à horodater. Chaque nœuds interne de l'arborescence contient une valeur de hachage, qui est générée en hachant l'enchaînement des nœuds fils. La valeur de hachage racine, qui représente sans ambiguïté tous les objets de données, est horodatée.

4.1 Syntaxe

Un horodatage d'archive a la syntaxe ASN.1 suivante :

Horodatage d'archive ASN.1

```
ArchiveTimeStamp ::= SEQUENCE {
  digestAlgorithm [0] AlgorithmIdentifier FACULTATIF,
  attributes          [1] Attributes FACULTATIF,
  reducedHashtree [2] SEQUENCE DE PartialHashtree FACULTATIF,
  timeStamp       ContentInfo}
PartialHashtree ::= SEQUENCE DE CHAÎNE D'OCTETS
Attributes ::= ENSEMBLE TAILLE (1..MAX) DE Attribute
```

Les champs de type ArchiveTimeStamp ont la signification suivante :

digestAlgorithm identifie l'algorithme de résumé et tous les paramètres associés utilisés dans l'arborescence de hachage réduite. Si le champ facultatif digestAlgorithm n'est pas présent, l'algorithme de résumé de l'horodatage DOIT être utilisé. Ce qui signifie que si des horodatages conformes à la [RFC3161] sont utilisés dans ce cas, le contenu de ce champ est identique au hashAlgorithm du champ messageImprint de TSTInfo.

attributes contient les informations qu'un LTA pourrait vouloir fournir pour documenter les étapes individuelles de renouvellement et la création des ArchiveTimeStamps individuels, par exemple, les politiques appliquées. Comme la structure du ArchiveTimeStamp peut être protégée par un hachage et des horodatages, l'ordre est pertinent, c'est pourquoi SET est utilisé à la place de SEQUENCE.

reducedHashtree contient des listes de valeurs de hachage, organisées en PartialHashtrees *(arborescences de hachage partielles)* pour une meilleure compréhension. Elles peuvent être déduites en réduisant une arborescence de hachage aux nœuds nécessaires pour vérifier un seul objet de données. Les valeurs de hachage sont représentées comme des chaînes d'octets. Si le champ facultatif reducedHashtree n'est pas présent, le ArchiveTimestamp contient simplement un horodatage ordinaire.

timeStamp devrait contenir l'horodatage défini au paragraphe 1.3. (Par exemple, comme défini avec TimeStampToken dans la [RFC3161]). D'autres types d'horodatage PEUVENT être utilisés, si ils contiennent des données de temps, des données horodatées, et une confirmation cryptographiquement sûre provenant du TSA de ces données.

4.2 Génération

Les listes de valeurs de hachage d'un horodatage d'archive peuvent être générées en construisant et en réduisant une arborescence de hachage de Merkle [MER1980].

Une telle arborescence de hachage peut être construite comme suit :

1. Collecter les objets de données à horodater.
2. Choisir un algorithme de hachage sûr H et générer les valeurs de hachage pour les objets de données. Ces valeurs vont être les feuilles de l'arborescence de hachage.
3. Pour chaque groupe de données contenant plus d'un document, les hachages respectifs de document sont triés en binaire en ordre ascendant, enchaînés, et hachés. Les valeurs de hachages sont le résultat complet de l'algorithme de hachage, c'est-à-dire, les zéros en tête ne sont pas retirés, avec le bit de poids fort en premier.
4. Si il y a plus d'une valeur de hachage, on les place en groupes et on trie chaque groupe en ordre binaire ascendant. On enchaîne ces valeurs et on génère de nouvelles valeurs de hachages, qui sont les nœuds internes de cette arborescence. (Si des valeurs de hachage supplémentaires sont nécessaires, par exemple, pour que tous les nœuds aient le même nombre d'enfants, chaque donnée peut être hachée en utilisant H et ensuite utilisée.) Répéter cette étape jusqu'à ce qu'il y ait seulement une valeur de hachage, qui est le nœud racine de l'arborescence de hachage.
5. Obtenir un horodatage pour cette valeur de hachage racine. L'algorithme de hachage dans la demande d'horodatage DOIT être le même que l'algorithme de hachage de l'arborescence de hachage, ou le champ digestAlgorithm de ArchiveTimeStamp DOIT être présent et spécifier l'algorithme de hachage de l'arborescence de hachage.

Un exemple d'une construction d'arborescence de hachage pour trois groupes de données, où les groupes de données 1 et 3 contiennent seulement un document, et le groupe de données 2 contient 3 documents :

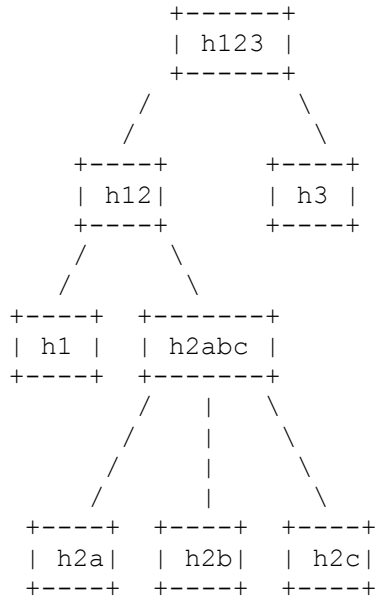


Figure 1 : arborescence de hachage

h1 = H(d1) où d1 est le seul objet de données dans le groupe de données 1.
h3 = H(d3) où d3 est le seul objet de données dans le groupe de données 3.
h12 = H(tri binaire et enchaînement de (h1, h2abc)).
h123 = H(tri binaire et enchaînement de (h12, h3)).
h2a = H(premier objet de données du groupe d'objets de données 2).
h2b = H(second objet de données du groupe d'objets de données 2).
h2c = H(troisième objet de données du groupe d'objets de données 2).
h2abc = H(tri binaire et enchaînement de (h2a, h2b, h2c)).

L'arborescence de hachage peut être réduite aux listes de valeurs de hachages, nécessaires pour avoir une preuve de l'existence d'un seul objet de données :

1. Générer la valeur de hachage h de l'objet de données, en utilisant l'algorithme de hachage H de l'arborescence de hachage.

2. Retenir toutes les valeurs de hachage qui ont le même nœud parent que h. Générer la première liste de valeurs de hachage en rangeant ces hachages en ordre binaire ascendant. Cela va être mémorisé dans la structure de la PartialHashtree. Répéter cette étape pour le nœud parent de tous les hachages jusqu'à atteindre le hachage racine. Les nœuds parents eux-mêmes ne sont pas sauvegardés dans les listes de hachages -- ils sont calculables.

3. La liste de toutes les partialHashtrees est finalement la reducedHashtree. (Toutes les valeurs de hachage spécifiées en dessous du même nœud parent, sauf le nœud parent des nœuds en dessous, sont groupées dans une PartialHashtree. La liste de séquence de toutes les Partialhashtrees est la reducedHashtree.)

4. Finalement, ajouter l'horodatage et les informations sur l'algorithme de hachage pour obtenir un horodatage d'archive.

En supposant que l'ordre binaire trié des hachages de la Figure 1 est h2abc < h1, l'arborescence de hachage réduite pour le groupe de données 1 (d1) est :

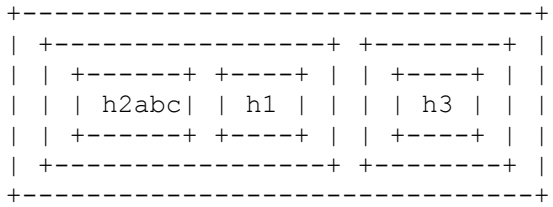


Figure 2 : Arborescence de hachage réduite pour le groupe 1 de données

Le pseudo ASN1 pour cette arborescence de hachage réduite rht1 ressemblerait à :
rht1 = SEQ(pht1, pht2)
avec les arborescences partielles de hachage pht1 et pht2
pht1 = SEQ (h2abc, h1)
pht2 = SEQ (h3)

En supposant la même arborescence de hachage que dans la Figure 1, l'arborescence de hachage réduite pour tous les objets de données dans le groupe de données 2 est identique.

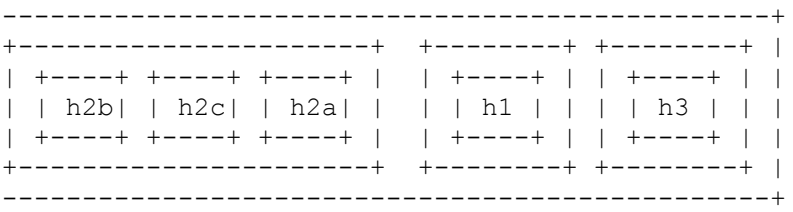


Figure 3 : Arborescence de hachage réduite pour le groupe d'objets de données 2

Le pseudo ASN1 pour cette arborescence de hachage réduite ressemblerait à
rht2 = SEQ(pht3, pht4, pht5)
avec les arborescences partielles de hachage pht3, pht4, et pht5
pht3 = SEQ (h2b, h2c, h2a)
pht4 = SEQ (h1)
pht5 = SEQ (h3)

Noter qu'il n'y a pas de restriction sur la quantité ou la longueur des listes de valeurs de hachage. Noter aussi qu'il est avantageux mais pas exigé de construire des arborescences de hachage et de les réduire. Un horodatage d'archive peut consister en une seule liste de valeurs de hachage et un horodatage ou seulement un horodatage sans liste de valeurs de hachage.

Les données (par exemple, certificats, listes de révocation de certificats (CRL), ou réponses du protocole d'état de certificat en ligne (OCSP)) nécessaires pour vérifier l'horodatage DOIVENT être préservées, et DEVRAIENT être mémorisées dans l'horodatage lui-même sauf si cela cause des duplications inutiles. Un horodatage selon la [RFC3161] est un objet de CMS dans lequel des certificats peuvent être mémorisés dans le champ Certificats et les CRL peuvent être mémorisées dans le champ crls des données signées. Les réponses OCSP peuvent être mémorisées comme des attributs non signés [RFC3126]. Noter que [ANSI.X9-95], [ISO-18014-2], et [ISO-18014-3] spécifient des horodatages vérifiables qui ne dépendent pas des certificats, des CRL, ou des réponses OCSP.

4.3 Vérification

Un horodatage d'archive devra prouver qu'un objet de données existait à un certain moment, donné par l'horodatage. Cela peut être vérifié comme suit :

1. Calculer la valeur de hachage h de l'objet de données avec l'algorithme de hachage H donné dans le champ digestAlgorithm de l'horodatage d'archive.

2. Rechercher la valeur de hachage h dans la première liste (partialHashtree) de reducedHashtree. Si elle n'est pas présente, terminer le processus de vérification avec un résultat négatif.

3. Enchaîner les valeurs de hachages de la liste réelle (partialHashtree) des valeurs de hachage en ordre binaire ascendant et calculer la valeur de hachage h' avec l'algorithme H. Cette valeur de hachage h' DOIT devenir un membre de la prochaine plus haute liste des valeurs de hachage (à partir de la prochaine partialHashtree). Continuer l'étape 3 jusqu'à ce qu'une valeur de hachage racine soit calculée.

4. Vérifier l'horodatage. Dans le cas d'un horodatage selon la [RFC3161], la valeur de hachage racine doit correspondre au message haché, et digestAlgorithm doit correspondre au champ hashAlgorithm, tous deux dans le champ messageImprint du timeStampToken *(jeton d'horodatage)*. Dans le cas d'autres formats d'horodatage, la valeur de hachage et l'algorithme de résumé doivent aussi correspondre à leurs champs équivalents si ils existent.

Si une preuve est nécessaire pour plus d'un objet de données, les étapes 1 et 2 doivent être faites pour tous les objets de données à prouver. Si une preuve supplémentaire est nécessaire que l'horodatage d'archive se rapporte à un groupe d'objets de données (par exemple, un document et toutes ses signatures) il peut être vérifié en plus que seules les valeurs de hachage des objets de données concernés sont dans la première liste de valeurs hachées.

5. Chaîne d'horodatages d'archive et séquence d'horodatages d'archive

Un horodatage d'archive prouve l'existence d'objets de données seuls ou d'un groupe d'objets de données à un certain moment. Cependant, ce premier horodatage d'archive dans la première ArchiveTimeStampChain *(chaîne d'horodatages d'archive)* peut devenir invalide si les algorithmes de hachage ou les algorithmes de clé publique utilisés dans son arborescence de hachage ou d'horodatage deviennent faibles ou si la période de validité du certificat d'autorité d'horodatage expire ou est révoqué.

Avant un tel événement, l'existence de l'horodatage d'archive ou des données d'archive horodatées doit être réassurée. Cela peut être fait en créant un nouvel horodatage d'archive. Selon que l'horodatage devient invalide ou que l'algorithme de hachage de l'arborescence de hachage devient faible, deux sortes de renouvellement d'horodatage d'archive sont possibles :

- Renouvellement d'horodatage : un nouvel horodatage d'archive est généré, qui couvre l'horodatage de l'ancien. Un ou plusieurs horodatages d'archive générés par le renouvellement d'horodatage donnent une chaîne d'horodatage d'archive pour un objet de données ou groupe d'objets de données.

- Renouvellement d'arborescence de hachage : un nouvel horodatage d'archive est généré, qui couvre tous les anciens horodatages d'archive ainsi que les objets de données. Une nouvelle chaîne d'horodatage d'archive commence. Une ou plusieurs chaînes d'horodatages d'archive pour un objet de données ou groupe d'objets de données donne une séquence d'horodatages d'archive.

Après le renouvellement, toujours seul le dernier (c'est-à-dire, le plus récent) ArchiveTimeStamp et les algorithmes et horodatages qu'il utilise doivent être observés concernant l'expiration et la perte de sécurité.

5.1 Syntaxe

ArchiveTimeStampChain et ArchiveTimeStampSequence ont la syntaxe ASN.1 suivante :

ArchiveTimeStampChain et ArchiveTimeStampSequence ASN.1 :

```
ArchiveTimeStampChain  ::= SEQUENCE DE ArchiveTimeStamp

ArchiveTimeStampSequence ::= SEQUENCE DE ArchiveTimeStampChain
```

ArchiveTimeStampChain et ArchiveTimeStampSequence DOIVENT être ordonnés en heure d'horodatage ascendante. Dans une ArchiveTimeStampChain, toutes les reducedHashtrees des ArchiveTimeStamps contenues DOIVENT utiliser le même algorithme de hachage.

5.2 Génération

L'horodatage d'archive initial se rapporte à un objet de données ou groupe d'objets de données. Avant que les algorithmes de chiffrement qui sont utilisés dans le plus récent horodatage d'archive (qui est, au début, l'algorithme initial) deviennent faibles ou que leurs certificats d'horodatage deviennent invalides, les horodatages d'archive doivent être renouvelés en générant un nouvel horodatage d'archive.

Dans le cas d'un renouvellement d'horodatage, le contenu du champ Horodatage de l'ancien horodatage d'archive doit être haché et horodaté par un nouvel horodatage d'archive. Le nouvel horodatage d'archive PEUT ne pas contenir de champ reducedHashtree, si l'horodatage couvre seulement simplement l'horodatage précédent. Cependant, on peut généralement collecter un certain nombre de vieux horodatages d'archive et construire la nouvelle arborescence de hachage avec les valeurs de hachage du contenu de leurs champs Horodatage.

Le nouvel horodatage d'archive DOIT être ajouté à la chaîne d'horodatages d'archive. Cette arborescence de hachage du nouvel horodatage d'archive DOIT utiliser le même algorithme de hachage que l'ancien, qui est spécifié dans le champ digestAlgorithm de l'horodatage d'archive ou, si cette valeur n'est pas établie (car elle est facultative) au sein de l'horodatage lui-même.

Dans le cas d'un renouvellement d'arborescence de hachage, l'horodatage d'archive et les objets de données archivés couverts par l'horodatage d'archive doivent être à nouveau hachés et horodatés, comme décrit ci-dessous :

1. Choisir un algorithme de hachage sûr H.

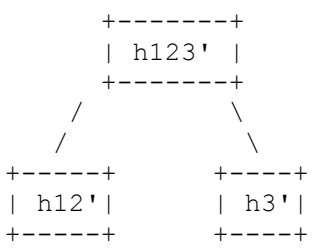
2. Choisir les objets de données d(i) auxquels on se réfère par un horodatage d'archive initial (objets qui sont encore présents et non supprimés). Générer les valeurs de hachage h(i) = H((d(i)). Si des groupes de données avec plus d'un document sont présents, l'un d'eux aura alors plus d'un hachage pour un groupe, c'est-à-dire, h(i_a), h(i_b).., h(i_n)

3. atsc(i) est la séquence d'horodatages d'archive *(ArchiveTimeStampSequence)* codée, l'enchaînement de toutes les précédentes chaînes horodatages d'archive (en ordre chronologique) par rapport à l'objet de données d(i). Générer la valeur de hachage ha(i) = H(atsc(i)).
   Note : les chaînes horodatage d'archive utilisées sont codées en DER, c'est-à-dire, elles contiennent leurs étiquettes de séquence et de longueur.

4. Enchaîner chaque h(i) avec ha(i) en ordre binaire ascendant et générer les valeurs de hachage h(i)' = H (h(i) + ha(i)). Pour les groupes multi documents, c'est :
   h(i_a)' = H (h(i_a) + ha(i))
   h(i_b)' = H (h(i_b) + ha(i)), etc.

5. Construire un nouvel horodatage d'archive pour chaque h(i)'. (La génération et la réduction d'arborescence de hachage sont définies au paragraphe 4.2 ; noter que chaque h(i)' va être traité au paragraphe 4.2 comme le hachage de document. La première liste de valeurs de hachage dans l'arborescence de hachage réduite devrait seulement contenir h(i)'. Pour un groupe multi documents, la première liste de valeurs de hachage va contenir les nouveaux hachages pour tous les documents dans ce groupe, c'est-à-dire, h(i_a)', h(i_b)'.., h(i_n)')

6. Créer une nouvelle chaîne d'horodatages d'archive *(ArchiveTimeStampChain)* contenant le nouvel horodatage d'archive et ajouter cette ArchiveTimeStampChain à la séquence d'horodatages d'archive *(ArchiveTimeStampSequence)*.



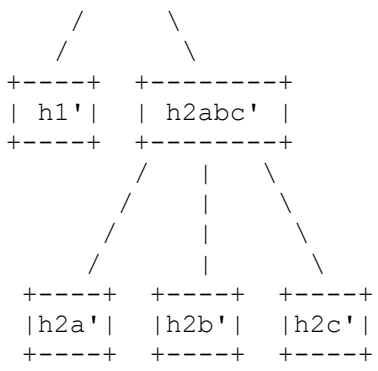


Figure 4 : Arborescence de hachage provenant du renouvellement d'arborescence de hachage

Soit H le nouvel algorithme de hachage sûr, ha(1), ha(2), ha(3) sont comme défini à l'étape 4 ci-dessus,
h1' = H(trié en binaire et enchaîné (H(d1), ha(1))) d1 est le document original provenant du groupe de données 1,
h3' = H(trié en binaire et enchaîné (H(d3), ha(3))) d3 est le document original provenant du groupe de données 3
h2a = H(premier objet de données du groupe d'objets de données 2)
...
h2c = H(troisième objet de données du groupe d'objets de données 2)
h2a' = H(trié en binaire et enchaîné (h2a, ha(2)))
...
h2c' = H(trié en binaire et enchaîné (h2c, ha(2)))
h2abc = H(trié en binaire et enchaîné (h2a', h2b', h2c'))

Les horodatages d'archive qui ne sont pas nécessaires à la vérification ne devraient pas être ajoutés à une chaîne d'horodatages d'archive ou à une séquence d'horodatages d'archive.

5.3 Vérification

Pour obtenir une preuve de non répudiation qu'un objet de données existait à un certain moment, les chaînes d'horodatages d'archive et leurs relations mutuelles et avec les objets de données doivent être prouvées :

1. Vérifier que le premier horodatage d'archive de la première chaîne d'horodatages d'archive (l'horodatage d'archive initial) contient la valeur de hachage de l'objet de données.

2. Vérifier chaque chaîne d'horodatages d'archive. La première liste de valeurs de hachage de chaque horodatage d'archive DOIT contenir la valeur de hachage de l'horodatage de l'horodatage d'archive précédent. Chaque horodatage d'archive DOIT être valide par rapport au moment de l'horodatage d'archive suivant. Tous les horodatages d'archive au sein d'une chaîne DOIVENT utiliser le même algorithme de hachage et cet algorithme DOIT être sûr au moment du premier horodatage d'archive de la chaîne d'horodatages d'archive suivante.

3. Vérifier que la première liste de valeurs de hachage (partialHashtree) du premier horodatage d'archive de toutes les autres chaînes d'horodatages d'archive contiennent une valeur de hachage de l'enchainement du hachage de l'objet de données et de la valeur de hachage de toutes les anciennes chaînes d'horodatages d'archive. Vérifier que cet horodatage d'archive a été généré avant que le dernier horodatage d'archive de la chaîne d'horodatages d'archive devienne invalide.

Afin d'achever la preuve de non répudiation pour les objets de données, le dernier horodatage d'archive doit être valide au moment où la vérification est effectuée.

Si la preuve est nécessaire pour plus d'un objet de données, les étapes 1 et 3 doivent être faites pour tous ces objets de données. Pour prouver que la séquence d'horodatages d'archive se rapporte à un groupe d'objets de données, vérifier que chaque premier horodatage d'archive de la première chaîne d'horodatages d'archive de la séquence d'horodatages d'archive de chaque objet de données ne contient pas d'autres valeurs de hachage dans sa première liste de valeurs de hachage (que les valeurs de hachage des autres objets de données).

6. Chiffrement

Si des fournisseurs de services sont utilisés pour archiver des données et générer des horodatages d'archive, il pourrait être désirable ou exigé que les clients envoient seulement des données chiffrées à archiver. Cependant, cela signifie que les enregistrements de preuves se réfèrent à des objets de données chiffrés. ERS protège directement l'intégrité du flux binaire et cela gèle la structure de bits au moment de l'archivage. Cela empêche de changer le schéma de chiffrement durant la période d'archivage, par exemple, si le schéma de chiffrement n'est plus sûr, un changement n'est pas possible sans perte de la preuve d'intégrité du EvidenceRecord. Dans ce cas, le service d'une transformation des données (et par là aussi un rechiffrement possible) fait par un service de notaire pourrait être une solution. Pour éviter des problèmes quand on utilise les enregistrements de preuve à l'avenir, des précautions particulières supplémentaires doivent être prises :

- On ne peut pas toujours s'appuyer sur la preuve générée pour prouver l'existence de données chiffrées pour prouver l'existence des données non chiffrées. Il peut être possible de choisir un algorithme ou une clé de déchiffrement qui ne soit pas l'algorithme ou la clé utilisé pour le chiffrement. Dans ce cas, l'enregistrement de preuve ne va pas être une preuve de non répudiation pour les données non chiffrées. Donc, seules devraient être utilisées des méthodes de chiffrement qui rendent possible de prouver que les objets de données chiffrés avec un horodatage d'archive représentent sans ambiguïté les objets de données non chiffrés. Toutes les données nécessaires pour prouver une représentation non ambiguë devraient être incluses dans les objets de données archivés. (Note : De plus, la sécurité à long terme des schémas de chiffrement devrait être analysée pour déterminer si ils pourraient être utilisés pour créer des attaques de collision.)

- Quand un consommateur d'assertions utilise un enregistrement de preuve pour prouver l'existence des objets de données chiffrés, il peut être souhaitable que les clients mémorisent seulement les objets de données non chiffrés et suppriment la copie chiffrée. Afin d'utiliser l'enregistrement de preuve, il doit alors être possible de rechiffrer sans ambiguïté les données non chiffrées pour obtenir exactement les données qui ont été archivées à l'origine. Donc, les données supplémentaires nécessaires pour re-chiffrer les objets de données devraient être insérées dans l'enregistrement de preuve par le client, c'est-à-dire, le LTA ne voit jamais ces valeurs.

Une structure extensible est définie pour mémoriser les paramètres nécessaires des méthodes de chiffrement. L'utilisation de encryptionInfoType *(type d'informations de chiffrement)* et encryptionInfoValue *(valeur d'informations de chiffrement)* spécifiés peut dépendre beaucoup des mécanismes et devra être défini dans d'autres spécifications.

6.1 Syntaxe

Le champ EncryptionInfo dans EvidenceRecord a la syntaxe suivante selon la version de l'ASN.1.

6.1.1 EncryptionInfo en ASN.1 1988

1988 ASN.1 EncryptionInfo

```
EncryptionInfo  ::= SEQUENCE {
   encryptionInfoType  IDENTIFIANT D'OBJET,
   encryptionInfoValue  TOUT DEFINI PAR encryptionInfoType
}
```

6.1.2 EncryptionInfo en ASN.1 1997

1997-ASN.1 EncryptionInfo

```
EncryptionInfo ::=  SEQUENCE {
   encryptionInfoType  ENCINFO-TYPE.&id ({SupportedEncryptionAlgorithms}),
   encryptionInfoValue  ENCINFO-TYPE.&Type ({SupportedEncryptionAlgorithms}{@encryptionInfoType})
}

ENCINFO-TYPE ::= IDENTIFIANT DE TYPE

SupportedEncryptionAlgorithms ENCINFO-TYPE ::= {...}
```

encryptionInfo contient les informations nécessaires pour le rechiffrement non ambigu du contenu non chiffré afin qu'il corresponde exactement aux objets de données chiffrés protégés par l'enregistrement de preuve.

7. Considérations sur la sécurité

Algorithmes sûrs : les algorithmes et paramètres de chiffrement qui sont utilisés dans les horodatages d'archive doivent être sécurisés au moment de la génération. Cela concerne l'algorithme de hachage utilisé dans les listes de hachage d'horodatage d'archive ainsi que dans les algorithmes de hachage et les algorithmes de clé publique des horodatages. Les publications concernant la sécurité des algorithmes de chiffrement ([NIST.800-57] et [TS.102 176-1]) doivent être considérées par les composants de vérification. Une solution générique pour l'interprétation automatique des politiques de sécurité convenables en forme électronique est souhaitable mais n'est pas l'objet de la présente spécification.

Redondance : rétrospectivement, certaines parties d'un horodatage d'archive peuvent se trouver avoir perdu leur acceptabilité pour la sécurité avant que cela soit connu publiquement. Par exemple, retrospectivement, il peut se trouver que les algorithmes aient perdu leurs capacités de sécurité, et que même les TSA ne soient plus de confiance. Il peut en résulter que les horodatages d'archive qui les utilisent perdent leur force probante. De nombreux TSA utilisent les mêmes algorithmes de signature. Bien que la compromission d'une clé privée n'affecte la sécurité que d'un TSA spécifique, la perte rétrospective de sécurité d'un algorithme de signature impactera en une fois un nombre potentiellement grand de TSA. Pour contrer de tels risques, il est recommandé de générer et gérer au moins deux enregistrements de preuve redondants avec des séquences d'horodatage d'archive, en utilisant différents algorithmes de hachage et différents TSA en utilisant différents algorithmes de signature. Pour mieux réaliser et gérer cette redondance, il est recommandé de gérer les horodatages d'archive dans un LTA central.

Horodatages sûrs : l'horodatage d'archive dépend de la sécurité de l'horodatage normal. Les exigences de sécurité pour les autorités d'horodatage déclarées dans les politiques de sécurité doivent être respectées. Les horodatages d'archive renouvelés devraient avoir une qualité égale ou supérieure à celle de l'horodatage d'archive initial. Les horodatages d'archive utilisés pour le renouvellement de signature de données signées, devraient avoir une qualité égale ou supérieure à la qualité maximum des signatures.

Chiffrement sûr : pour une preuve de non répudiation, il n'importe pas que le chiffrement ait été cassé ou non. Néanmoins, les utilisateurs devraient garder secrètes leurs clés privées et les nombres aléatoires utilisés pour le chiffrement et ne les divulguer que si nécessaire, par exemple, dans un procès, à un juge ou un expert. Ils devraient utiliser dcs algorithmes et paramètres de chiffrement réputés incassables tant que la confidentialité des données archivées est importante.

8. Références

8.1 Références normatives

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997. *(MàJ par* RFC8174)

[RFC3161] C. Adams, P. Cain, D. Pinkas et R. Zuccherato, "Protocole d'horodatage (TSP) d'infrastructure de clé publique X.509 pour l'Internet", août 2001.

[RFC3280] R. Housley, W. Polk, W. Ford et D. Solo, "Profil de certificat d'infrastructure de clé publique X.509 et de liste de révocation de certificat (CRL) pour l'Internet", avril 2002. *(Obsolète, voir* RFC5280)

[RFC3852] R. Housley, "Syntaxe de message cryptographique (CMS)", juillet 2004. *(Obsolète, voir la* RFC5652)

[X.208] Recommandation UIT-T X.208, "Spécification de la notation numéro un de syntaxe abstraite (ASN.1)", Genève, novembre 1988.

[X.209] Recommandation UIT-T X.209, "Règles de codage de l'ASN.1 : spécification des règles de codage de base (BER)", Genève, 1988.

8.2 Références pour information

[ANSI.X9-95] American National Standard for Financial Services, "Trusted Timestamp Management and Security", ANSI X9.95, juin 2005.

[X.680] Recommandation UIT-T X.680 | ISO/CEI 8824-1:2002 "Technologies de l'information - Notation de syntaxe abstraite n°1 (ASN.1) : Spécification de la notation de base". (07/2002)

[X.690] Recommandation UIT-T X.690 | ISO/CEI 8825-1:2002, "Technologies de l'information - Règles de codage de l'ASN.1 : Spécification des règles de codage de base (BER), règles de codage canoniques (CER) et règles de codage distinctives (DER)", (07/2002).

[TS.102 176-1] ETSI TS 102 176-1 V1.2.1, "Electronic Signatures and Infrastructures (ESI) Algorithms and Parameters for Secure Electronic Signatures ; Part 1: Hash functions and asymmetric algorithms", European Telecommunication Standards Institute, juillet 2005.

[ISO-18014-1] ISO/CEI JTC 1/SC 27, "Services d'horodatage - Partie 1 : cadre", ISO-18014-1, février 2002.

[ISO-18014-2] ISO/CEI JTC 1/SC 27, "Services d'horodatage - Partie 2 : mécanismes de production de jetons indépendants", ISO-18014-2, décembre 2002.

[ISO-18014-3] ISO/CEI JTC 1/SC 27, "Services d'horodatage - Partie 3 : mécanismes de production de jetons reliés", ISO-18014-3, février 2004.

[MER1980] Merkle, R., "Protocols for Public Key Cryptosystems, Proceedings of the 1980 IEEE Symposium on Security and Privacy (Oakland, CA, USA)", pages 122-134, avril 1980.

[NIST.800-57] National Institute of Standards and Technology, "Recommendation for Key Management - Part 1: General (Revised)", NIST 800-57 Part1, mai 2006.

[RFC3126] D. Pinkas, J. Ross et N. Pope, "Formats de signature électronique pour signatures électroniques à long terme", septembre 2001. *(Remplacée par* RFC5126) *(Info.)*

[RFC4810] C. Wallace et autres, "Exigences pour un service d'archive à long terme", mars 2007. *(Information)*

Appendice A. Enregistrement de preuve avec la CMS

Un enregistrement de preuve peut être ajouté à des données signées ou des données enveloppées afin de les transférer de façon concluante. Pour la CMS, un endroit adéquat pour mémoriser un tel enregistrement de preuve est dans un attribut non signé (d'un message signé) ou dans un attribut non protégé (d'un message enveloppé).

Un avantage de la mémorisation de l'enregistrement de preuve au sein de la structure de CMS est que toutes les données peuvent être transférées dans un fichier concluant et qu'elles sont directement connectées. Les documents, les signatures, et leurs enregistrements de preuve peuvent être regroupés et gérés ensemble. La description de cet appendice contient la spécification normative de la façon d'intégrer ERS dans les structures de CMS.

L'enregistrement de preuve contient aussi des informations sur la méthode de choix qui a été utilisée pour la génération des objets de données à horodater. Dans le cas de la CMS, deux méthodes de choix peuvent être distinguées :

1. L'objet CMS comme un tout incluant contentInfo est choisi comme objet de données et d'archive horodatée. Cela signifie qu'une valeur de hachage de l'objet CMS DOIT être située dans la première liste de valeurs de hachage des horodatages d'archive.

2. L'objet CMS et le contenu signé ou chiffré sont inclus dans l'horodatage d'archive comme des objets séparés. Dans ce cas, la valeur de hachage de l'objet CMS ainsi que la valeur de hachage du contenu doivent être mémorisés dans la première liste des valeurs de hachage comme un groupe d'objets de données.

Cependant, d'autres méthodes de choix pourraient aussi être appliquées, par exemple, comme dans la [RFC3126].

Dans le cas des deux méthodes de choix définies ci-dessus, l'enregistrement de preuve doit être ajouté à la première signature de l'objet CMS des données signées. Selon la méthode de choix, les identifiants d'objet suivants sont définis pour l'enregistrement de preuve :

Attribut ASN.1 interne EvidenceRecord

IDENTIFIANT D'OBJET id-aa-er-internal ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) id-aa(2) 49 }

Attribut ASN.1 externe EvidenceRecord

IDENTIFIANT D'OBJET id-aa-er-external ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) id-aa(2) 50 }

Les attributs DEVRAIENT ne se produire qu'une seule fois. Si ils apparaissent plusieurs fois, ils doivent être mémorisés dans la première signature dans l'ordre chronologique.

Si l'objet CMS n'a pas d'attribut EvidenceRecord -- qui indique que l'EvidenceRecord a été fourni en externe -- l'objet de données d'archive horodaté doit être généré sur l'objet CMS complet dans le codage existant.

En cas de vérification, si un seul EvidenceRecord est contenu dans l'objet CMS, la valeur de hachage doit être générée sur l'objet CMS sans le seul EvidenceRecord. Cela signifie que l'attribut doit être retiré avant la vérification. La longueur des champs contenant des étiquettes doit être adapté. À part cela, le codage existant ne doit pas être modifié.

Si plusieurs horodatages d'archive se produisent, l'objet de données doit être généré comme suit :

- Durant la vérification du premier (en ordre chronologique) EvidenceRecord, tous les EvidenceRecord doivent être retirés afin de générer l'objet de données.

- Durant la vérification du nième EvidenceRecord, les n-1 premiers attributs devraient rester dans l'objet CMS.

- La vérification du nième EvidenceRecord doit résulter en un instant où le document doit avoir existé avec les n premiers attributs. La vérification de l'attribut n+1 doit prouver que cette exigence a été satisfaite.

Appendice B. Module ASN.1 avec syntaxe 1988

Module ASN.1

```
ERS {iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) ltans(11) id-mod(0) id-mod-ers88(2)
id-mod-ers88-v1(1) }
DEFINITIONS D'ÉTIQUETTES IMPLICITES ::=
DÉBUT

-- EXPORTE TOUT --

IMPORTE

-- Importe de la syntaxe de message cryptographique de la from RFC 3852
ContentInfo, Attribute

DE CryptographicMessageSyntax2004 -- DE [RFC3852]
      { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0) cms-2004(24) }

-- Importe de la [RFC3280], Appendice A.1
AlgorithmIdentifier
   DE PKIX1Explicit88
      { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) mod(0) pkix1-explicit(18) }
;
```

```
IDENTIFIANT D'OBJET ltans ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5)
                                    ltans(11) }

EvidenceRecord ::= SEQUENCE {
  version               ENTIER { v1(1) } ,
  digestAlgorithms   SEQUENCE DE AlgorithmIdentifier,
  cryptoInfos             [0] CryptoInfos FACULTATIF,
  encryptionInfo          [1] EncryptionInfo FACULTATIF,
  archiveTimeStampSequence  ArchiveTimeStampSequence
  }

CryptoInfos ::= TAILLE DE SEQUENCE (1..MAX) DE Attribute

ArchiveTimeStamp ::= SEQUENCE {
 digestAlgorithm [0] AlgorithmIdentifier FACULTATIF,
 attributes           [1] Attributes FACULTATIF,
 reducedHashtree [2] SEQUENCE DE PartialHashtree FACULTATIF,
 timeStamp      ContentInfo}

PartialHashtree ::= SEQUENCE DE CHAINE D'OCTETS

Attributes ::= TAILLE D'ENSEMBLE (1..MAX) DE Attribute

ArchiveTimeStampChain    ::= SEQUENCE DE ArchiveTimeStamp

ArchiveTimeStampSequence ::= SEQUENCE DE ArchiveTimeStampChain

EncryptionInfo  ::= SEQUENCE {
     encryptionInfoType    IDENTIFIANT D'OBJET,
     encryptionInfoValue    TOUT DEFINI PAR encryptionInfoType}

IDENTIFIANT D'OBJET id-aa-er-internal ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
                                              smime(16) id-aa(2) 49 }

IDENTIFIANT D'OBJET id-aa-er-external ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
                                              smime(16) id-aa(2) 50 }

FIN
```

Appendice C. Module ASN.1 avec syntaxe 1997

```
Module ASN.1

ERS {iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) ltans(11) id-mod(0) id-mod-ers(1) id-
mod-ers-v1(1) }
DEFINITIONS D'ÉTIQUETTES IMPLICITES ::=
DÉBUT

-- EXPORTE TOUT --

IMPORTE

-- Importe de PKCS-7
  ContentInfo
     DE PKCS7
        {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-7(7) modules(0)}

-- Importe de AuthenticationFramework
```

```
AlgorithmIdentifier
   DE AuthenticationFramework
      {joint-iso-itu-t ds(5) module(1) authenticationFramework(7) 4}

-- Importe de InformationFramework
Attribute
   DE InformationFramework
      {joint-iso-itu-t ds(5) module(1) informationFramework(1) 4}
  ;

IDENTIFIANT D'OBJET ltans ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5)
                                ltans(11) }

EvidenceRecord ::= SEQUENCE {
  version               ENTIER { v1(1) } ,
  digestAlgorithms        SEQUENCE DE AlgorithmIdentifier,
  cryptoInfos             [0] CryptoInfos FACULTATIF,
  encryptionInfo          [1] EncryptionInfo FACULTATIF,
  archiveTimeStampSequence  ArchiveTimeStampSequence
  }

CryptoInfos ::= TAILLE DE SEQUENCE (1..MAX) DE Attribute
        (AVEC COMPOSANTS {
          ...,
          valuesWithContext   ABSENT
         })

ArchiveTimeStamp ::= SEQUENCE {
 digestAlgorithm [0] AlgorithmIdentifier FACULTATIF,
 attributes           [1] Attributes FACULTATIF,
 reducedHashtree [2] SEQUENCE DE PartialHashtree FACULTATIF,
 timeStamp      ContentInfo}

PartialHashtree ::= SEQUENCE DE CHAINE D'OCTETS

Attributes ::= TAILLE D'ENSEMBLE (1..MAX) DE Attribute
        (AVEC COMPOSANTS {
          ...,
          valuesWithContext   ABSENT
         })

ArchiveTimeStampChain    ::= SEQUENCE DE ArchiveTimeStamp

ArchiveTimeStampSequence ::= SEQUENCE DE ArchiveTimeStampChain

EncryptionInfo  ::=  SEQUENCE {
     encryptionInfoType   ENCINFO-TYPE.&id ({SupportedEncryptionAlgorithms}),
     encryptionInfoValue  ENCINFO-TYPE.&Type ({SupportedEncryptionAlgorithms}{@encryptionInfoType})
}

ENCINFO-TYPE ::= TYPE-IDENTIFIANT

  SupportedEncryptionAlgorithms ENCINFO-TYPE ::= {...}

IDENTIFIANT D'OBJET id-aa-er-internal ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
                                            smime(16) id-aa(2) 49 }

IDENTIFIANT D'OBJET id-aa-er-external ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
                                            smime(16) id-aa(2) 50 }

FIN
```

Adresse des auteurs

Tobias Gondrom
Open Text Corporation
Werner-von-Siemens-Ring 20
Grasbrunn, Munich D-85630
Allemagne
téléphone : +49 (0) 89 4629-1816
mél : tobias.gondrom@opentext.com

Ralf Brandner
InterComponentWare AG
Industriestrasse 41
Walldorf D-69119
Allemagne
ralf.brandner@intercomponentware.com

Ulrich Pordesch
Fraunhofer Gesellschaft
Rheinstrasse 75
Darmstadt D-64295
Allemagne
mél : ulrich.pordesch@zv.fraunhofer.de

Déclaration complète de droits de reproduction

Remerciement
Le financement de la fonction d'édition des RFC est actuellement fourni par la Internet Society.