Soumission indépendante
Request for Comments : 5544
Catégorie : Information
ISSN: 2070-1721

A. Santoni, Actalis S.p.A.
février 2010

Traduction Claude Brière de L'Isle

Syntaxe pour lier des documents avec des horodatages

Résumé
Le présent document décrit une enveloppe qui peut être utilisée pour lier un fichier (pas nécessairement protégé au moyen de techniques cryptographiques) à un ou plusieurs jetons d'horodatage obtenus pour ce fichier, où "jeton d'horodatage" a la signification définie dans la RFC 3161 ou ses successeurs. Des types supplémentaires de preuve temporelle sont aussi permis.

L'enveloppe proposée se fonde sur la syntaxe de message cryptographique comme définie dans la RFC 5652.

Statut de ce mémoire
Le présent document n'est pas une spécification sur la voie de la normalisation de l'Internet ; il est publié à des fins d'information.

Ceci est une contribution à la série des RFC, indépendamment de tout autre flux de RFC. L'éditeur des RFC a choisi de publier le présent document à sa discrétion et ne fait aucune déclaration sur sa valeur de déploiement ou mise en œuvre. Les documents approuvés pour publication par l'éditeur des RFC ne sont candidats à aucun niveau de norme de l'Internet ; voir la Section 2 de la RFC 5741.

Les informations sur le statut actuel du présent document, tout errata, et comment fournir des réactions sur lui peuvent être obtenues à http://www.rfc-editor.org/info/rfc5544

Note de l'IESG
La présente RFC n'est candidate à aucun niveau de norme de l'Internet. La spécification sur la voie de la normalisation RFC 4998, syntaxe d'enregistrement de preuve (ERS, *Evidence Record Syntax*) spécifie un autre mécanisme. Les lecteurs sont invités à voir aussi la RFC4998 quand ils évaluent la convenance de ce mécanisme.

Table des matières

1. Introduction

L'horodatage est devenu la technique standard pour prouver l'existence d'un document avant un certain instant. Plusieurs législations dans le monde adoptent le concept et fournissent des services d'horodatage, principalement dans le but d'étendre la validité de documents signés. Cependant, bien que l'horodatage améliore les signatures numériques, sa valeur n'en dépend pas. Il peut clairement être utile d'horodater un document même si il n'est pas signé. Et il peut aussi être utile, ou même obligatoire dans certains cas, d'horodater un document signé dans sa totalité, sans considération du nombre de signatures qu'il contient.

Quand un horodatage se rapporte à une signature numérique, il existe déjà un moyen de garder les deux éléments ensemble : la [RFC3161] décrit comment un ou plusieurs TimeStampTokens *(jetons d'horodatage)* peuvent être inclus dans une structure SignerInfo *(informations de signataire)* comme attributs non signés. Par ailleurs, il n'y a pas de façon standard de garder ensemble un document horodaté, signé ou non, et les horodatages qui s'y rapportent.

Dans ce cas, deux approches sont normalement adoptées :
- o Les horodatages sont conservés comme des fichiers séparés (garder trace de quels horodatages appartiennent à quels documents relève de l'utilisateur).
- o Une solution ad hoc est adoptée pour des applications spécifiques, par exemple, une archive ZIP ou une "enveloppe" propriétaire d'une certaine sorte.

Les deux solutions entravent l'interopérabilité, ce qui est l'objectif du présent mémoire.

Le présent document décrit une syntaxe simple pour lier un document (en fait, toute sorte de fichier) à la preuve temporelle correspondante ; cette dernière est normalement représentée par un ou plusieurs TimeStampTokens de la RFC 3161. Des types supplémentaires de preuve temporelle, par exemple, un EvidenceRecord *(enregistrement de preuve)* de la [RFC4998], sont aussi pris en charge via une syntaxe "ouverte". Cependant, au nom de l'interopérabilité, l'accent est mis dans ce document sur les TimeStampTokens.

La syntaxe proposée est en gros fondée sur la syntaxe de message cryptographique (CMS, *Cryptographic Message Syntax*) définie dans la [RFC5652].

1.1 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].

Les termes "document" et "fichier" sont utilisés de façon interchangeable. Les termes "TimeStampToken" et "jeton d'horodatage" sont utilisés de façon interchangeable, tous deux se référant à la structure de données définie dans la RFC 3161.

2. Syntaxe pour TimeStampedData

La structure de données proposée est appelée TimeStampedData *(données horodatées)* et est fondée sur l'enveloppe ContentInfo définie dans la [RFC5652] :

```
ContentInfo ::= SEQUENCE {
      contentType ContentType,
      content [0] TOUT EXPLICITE DEFINI PAR contentType }

ContentType ::= IDENTIFIANT D'OBJET
```

Alors que la CMS définit six types de contenu (data, signed-data, enveloped-data, digested-data, encrypted-data, et authenticated-data) le présent mémoire définit un type de contenu supplémentaire, timestamped-data *(données horodatées)* identifié par l'identifiant d'objet (OID, *Object Identifier*) suivant :

IDENTIFIANT D'OBJET id-ct-timestampedData ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) id-smime(16) id-ct(1) 31 }

Cet OID particulier signale que le champ de contenu de ContentInfo a la syntaxe suivante :

```
TimeStampedData ::= SEQUENCE {
      version              ENTIER { v1(1) },
      dataUri              IA5String FACULTATIF,
      metaData             MetaData FACULTATIF,
      content              CHAINE D'OCTETS FACULTATIF,
      temporalEvidence  Evidence
 }

MetaData ::= SEQUENCE {
      hashProtected       BOOLEEN,
      fileName            UTF8String FACULTATIF,
      mediaType           IA5String FACULTATIF,
      otherMetaData       Attributes FACULTATIF
 }

Attributes ::= ENSEMBLE TAILLE(1..MAX) DE Attribute      -- conformément à la RFC 5652

Evidence ::= CHOIX {
      tstEvidence     [0] TimeStampTokenEvidence,          -- voir la RFC 3161
      ersEvidence     [RFC2119] EvidenceRecord,            -- voir la RFC 4998
      otherEvidence  [LEGAL] OtherEvidence
 }

OtherEvidence ::= SEQUENCE {
      oeType            IDENTIFIANT D'OBJET,
      oeValue           TOUT DEFINI PAR oeType }

TimeStampTokenEvidence ::= SEQUENCE TAILLE(1..MAX) DE TimeStampAndCRL

TimeStampAndCRL ::= SEQUENCE {
      timeStamp   TimeStampToken,                      -- conformément à la RFC 3161
      crl             CertificateList FACULTATIF       -- conformément à la RFC 5280
 }
```

Le champ version contient le numéro de version de la syntaxe de TimeStampedData. Il DEVRA être 1 pour la présente version du document.

Le champ dataUri contient une référence conforme à la [RFC3986]. Quand le champ content est absent, dataUri DOIT être présent et contenir un URI permettant la restitution du document qui a été horodaté (sauf si le document est déplacé ultérieurement). Quand le champ content est présent, ce champ PEUT aussi être présent.

Le champ metaData contient des métadonnées relatives au document qui a été horodaté, si applicable. En particulier :

Le champ hashProtected indique si les métadonnées ont été incluses dans le calcul du résumé au sein du premier TimeStampToken (voir plus loin). Cela rend possible la détection d'une altération ultérieure des métadonnées.

Le champ fileName contient le nom de fichier original du document qui a été horodaté.

Le champ mediaType contient un type/sous type de support et d'éventuels paramètres pour le document horodaté, conformément à la [RFC2045]. Ces informations peuvent aider à décider comment "ouvrir" ou traiter le document horodaté.

Le champ otherMetaData contient d'autres attributs du document horodaté (par exemple, une description, l'auteur prétendu, etc.) où chaque attribut est spécifié par un identifiant d'objet et un ensemble correspondant de valeurs, comme décrit dans la [RFC5652]. Quand ce champ est présent, il DOIT contenir au moins un attribut.

Dans le champ metaData (si il est présent) au moins un des sous champs fileName, mediaType, et otherMetaData DOIT être présent.

Les valeurs d'attribut dans le champ otherMetaData DOIVENT être codées en DER, même si le reste de la structure est codé en BER.

Le champ content, quand il est présent, porte le contenu entier, dans son format et codage original, du document qui a été horodaté. Cela peut en fait être toutes sortes de données, par exemple, un document de texte, un exécutable, un film, un message, etc. L'omission du champ content rend possible de lier la preuve temporelle à des données externes. Dans ce cas, la preuve temporelle est calculée comme si le champ content était présent.

Le champ temporalEvidence porte la preuve que le document horodaté n'existait pas avant un certain instant. Plusieurs types de preuve sont permis, mais les applications conformes sont seulement obligées de prendre en charge le type de la RFC 3161 -- à savoir le choix tstEvidence.

La séquence TimeStampTokenEvidence DOIT contenir au moins un élément de type TimeStampAndCRL.

Les éléments de la séquence TimeStampTokenEvidence DOIVENT se conformer à la règle suivante :
- si le champ metaData est absent ou si la valeur de son champ hashProtected est FAUX, alors le TimeStampToken dans le premier élément DEVRA être calculé sur les octets de valeur du champ content (si ce champ est absent, utiliser les octets restitués via le champ dataUri) ;
- autrement (le champ metaData est présent et la valeur de son champ hashProtected est VRAI) le TimeStampToken dans le premier élément DEVRA être calculé sur l'enchaînement des champs suivants :
  - le codage en DER du champ metaData ;
  - les octets de la valeur du champ content (si ce champ est absent, utiliser les octets restitués via le champ dataUri) ;
- le TimeStampToken dans le second élément DEVRA être calculé sur le premier élément ;
- le TimeStampToken dans chaque élément suivant DEVRA être calculé sur son élément précédent dans la séquence.

Dans la construction TimeStampAndCRL, le champ facultatif crl porte une liste de révocation de certificats (CRL, *Certificate Revocation List*) convenable montrant que le certificat de l'autorité d'horodatage (TSA, *Time-Stamping Authority*) qui a produit l'horodatage n'était pas révoqué au moment où l'élément suivant dans la séquence TimeStampTokenEvidence a été ajouté. Voir à la section des considérations sur la sécurité la discussion de ce sujet.

3. Exigences de conformité

Les applications conformes DOIVENT prendre en charge au moins le type de preuve fondé sur la RFC 3161 (c'est-à-dire, le CHOIX tstEvidence).

4. Traitement recommandé

Cette section se concentre sur le type de preuve fondé sur la RFC 3161. Le traitement de la structure pour d'autres types de preuve serait fait de manière similaire.

4.1 Génération d'une nouvelle structure TimeStampedData

Dans ce cas, les applications sont supposées se comporter comme suit :

- remplir le champ version avec la valeur d'entier v1(1) ;

- si une enveloppe auto-contenue doit être générée, toujours remplir le champ content avec le contenu du fichier dans son format et codage original ; selon l'application, le champ dataUri peut aussi être ajouté ;

- autrement (une enveloppe séparée doit être générée) toujours remplir le champ dataUri avec l'URI du document horodaté (par exemple, http://foo.example.com/Contract12345.pdf) ; utiliser une référence d'URI absolu ou d'URI relatif dépend de l'application ;

- si le champ metaData est ajouté, décider de la valeur de son champ hashProtected ; régler sa valeur à VRAI si l'application a besoin que les champs restants de la construction metaData soient protégés par un hachage comme décrit à la Section 2 ; autrement, le régler à FAUX ;

- si le champ metaData est ajouté, remplir facultativement le champ fileName (par exemple, "Contract12345.pdf") le champ mediaType avec un type/sous-type de support convenable et d'éventuels paramètres conformément à la [RFC2045], et le champ otherMetaData, selon l'application ;

- Choisir une fonction convenable de hachage unidirectionnel et calculer une valeur de hachage en utilisant cette fonction sur le contenu, ou l'enchaînement des métadonnées et du contenu, comme décrit à la Section 2 ; cette valeur de hachage va alors être utilisée pour demander le premier TimeStampToken ;

- obtenir la première preuve temporelle d'une TSA et l'ajouter au champ temporalEvidence ;

- insérer les TimeStampedData dans une structure ContentInfo, avec l'OID id-ct-timestampedData dans le champ contentType ;

- coder en BER la structure ContentInfo (sauf les champs qui sont obligés d'être codés en DER) et la sauvegarder avec un nom de fichier raisonnable (par exemple, déduit du nom du fichier horodaté).

4.2 Vérification d'une structure TimeStampedData existante

Dans ce cas, les applications sont supposées se comporter comme suit :

- vérifier que le champ contentType de la structure ContentInfo a la valeur attendue (id-ct-timestampedData) dans son champ contentType ; puis, extraire la structure TimeStampedData interne et continuer le traitement ;

- vérifier le champ version (il devrait être v1) ;

- vérifier que le champ temporalEvidence n'est pas vide ;

- vérifier si le contenu est présent ; si il ne l'est pas, utiliser le champ dataUri pour restituer le fichier ;

- ouvrir le premier élément de la séquence TimeStampTokenEvidence, ouvrir le jeton horodateur en son sein et utiliser la fonction de hachage qui a été utilisée pour l'obtenir pour recalculer le hachage des champs indiqués à la Section 2 ; si la valeur de hachage recalculée correspond à celle du jeton horodateur, continuer le traitement ; autrement, la structure TimeStampedData a été modifiée ;

- valider la temporalEvidence en vérifiant que :
  - chaque TimeStampToken dans la chaîne contient bien la valeur de résumé correcte (conformément aux règles décrites à la Section 2) et qu'il a été signé par une TSA de confiance,
  - la TSA correspondante qui signe le certificat n'a pas été révoquée au moment de la production du TimeStampToken suivant, fondé sur la CRL associée ;

- selon l'application, utiliser la preuve temporelle pour laquelle l'application a été conçue, quelle qu'elle soit ;

- selon l'application, montrer le dataUri, le fileName, le mediaType, les otherMetaData, et la preuve temporelle à l'utilisateur ;

- selon l'application, sauvegarder le contenu dans un fichier séparé ;

- selon l'application, mémoriser en un lieu différent le contenu qui a été restitué en utilisant le champ dataUri, puis mettre à jour le champ dataUri en conséquence ;

- selon l'application, montrer le fichier horodaté à l'utilisateur, éventuellement en activant un "lecteur" convenable.

4.3 Extension de validité d'une structure TimeStampedData existante

Dans ce cas, les applications sont supposées se comporter comme suit :

- valider la structure TimeStampedData comme décrit ci-dessus ;

- choisir le jeton d'horodatage provenant du dernier élément TimeStampAndCRL dans la chaîne et obtenir la dernière CRL disponible pour le certificat de TSA correspondant (si cette CRL n'est pas assez fraîche, attendre que la suivante soit disponible) puis la mémoriser dans l'élément TimeStampAndCRL ;

- instancier un nouvel élément TimeStampAndCRL et obtenir un nouveau jeton d'horodatage calculé sur le précédent, conformément aux règles décrites à la Section 2 ; insérer le nouveau jeton horodatage dans le nouvel élément TimeStampAndCRL, puis ajouter le dernier à la fin de la chaîne.

Voir la section des considérations sur la sécurité pour plus de discussion sur l'extension de la validité d'une structure TimeStampedDatae existante.

5. Considérations sur la sécurité

Quand le champ metaData est présent et que le sous champ hashProtected est réglé à VRAI, les métadonnées sont aussi incluses dans le calcul du résumé dans le premier jeton d'horodatage, de sorte que toute altération ultérieure des métadonnées va être facilement détectée. Cependant, l'intégrité des métadonnées protégées par hachage n'implique pas que les métadonnées étaient correctes au moment où l'objet TimeStampedData a été créé. Cela peut seulement être déduit par d'autre moyens (par exemple, du contexte). Par exemple, quand des objets TimeStampedData sont créés par un fournisseur de service d'archivage, il peut être raisonnable de supposer que les métadonnées sont correctes au moment de création. Plutôt, quand un objet TimeStampedData est reçu d'un inconnu, le receveur ne peut pas supposer sainement que les métadonnées sont correctes, par manque d'autres informations.

En général, un jeton d'horodatage ne devrait pas être considéré comme valide après l'expiration du certificat de la TSA productrice (aussi, cette considération dépend de la législation et de la politique selon lesquelles fonctionne la TSA). Cependant, un jeton d'horodatage peut lui-même être horodaté pour étendre la validité de la signature de la TSA. En appliquant cette technique de façon répétée, toute une chaîne de jetons d'horodatage peut être constituée pour étendre ad libitum la validité du premier. Donc, cette approche peut être adoptée pour étendre la validité d'une structure TimeStampedData au-delà de la date d'expiration du premier TimeStampToken qu'elle contient, en ajoutant d'autres éléments à la séquence TimeStampTokenEvidence conformément aux règles décrites à la Section 2. Bien sûr, chaque TimeStampToken supplémentaire doit être ajouté en temps utile (avant que le précédent soit expiré ou ait été révoqué).

La technique d'extension de validité décrite ci-dessus exige que la TSA qui signe les certificats puisse encore être vérifiée longtemps après qu'ils ont expiré, normalement en vérifiant une CRL. La CRL doit être capturée au moment convenable, parce que les certificats expirés sont normalement supprimés de la CRL sans considération de si ils ont été révoqués. La construction TimeStampAndCRL permet d'ajouter une CRL à la suite du TimeStampToken qui s'y rapporte, afin que le certificat de TSA soit encore vérifiable à tout moment futur. La CRL doit être capturée au moment où un autre élément est sur le point d'être ajouté à la séquence TimeStampTokenEvidence, ou même plus tard -- pour permettre qu'une demande de révocation de dernière minute soit traitée par la CA (voir la discussion sur les "périodes de grâce" dans la [RFC5126]).

6. Références normatives

[RFC2045] N. Freed et N. Borenstein, "Extensions de messagerie Internet multi-objets (MIME) Partie 1 : Format des corps de message Internet", novembre 1996. *(D. S., MàJ par* 2184, 2231, 5335.)

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997. DOI 10.17487/RFC2119, *(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.

[RFC3986] T. Berners-Lee, R. Fielding et L. Masinter, "Identifiant de ressource uniforme (URI) : Syntaxe générique", DOI 10.17487/RFC3986, STD 66, janvier 2005. *(P.S. ; MàJ par* RFC8820)

[RFC4998] T. Gondrom et autres, "Syntaxe d'enregistrement de preuve (ERS)", août 2007. *(P.S.)*

[RFC5280] D. Cooper et autres, "Profil de certificat d'infrastructure de clé publique X.509 et de liste de révocation de certificat (CRL) pour l'Internet", DOI 10.17487/RFC5280, mai 2008. (*Remplace les* RFC3280, RFC4325,

RFC4630) *(P.S. ; MàJ par* RFC8398, 8399)

[RFC5652] R. Housley, "Syntaxe de message cryptographique (CMS)", STD70, septembre 2009. *(Remplace* RFC3852)

7. Référence pour information

[RFC5126] D. Pinkas, N. Pope, J. Ross, "Signatures électroniques évoluées conformes à la syntaxe de message cryptographique (CAdES)", mars 2008. (*Remplace* RFC3126) *(Information)*

Appendice A. Module ASN.1

Le module ASN.1 contebu dans le présent appendice définit les structures qui sont nécessaires pour mettre en 0156uvre la présente spécification. Il est attendu qu'il soit utilisé en conjonction avec les modules ASN.1 des [RFC5652], [RFC3161], [RFC5280], et [RFC4998].

```
TimeStampedDataModule { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0) 35 }

ÉTIQUETTES IMPLICITES DE DÉFINITIONS ::=
DÉBUT
IMPORTE
-- Importe de la [RFC5652]
    Attribute
      FROM CryptographicMessageSyntax2004
       { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0) cms-2004(24) }

-- Importe de la [RFC3161]
    TimeStampToken
      FROM PKIXTSP
       { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0) id-mod-tsp(13)}

-- Importe de la [RFC5280]
   CertificateList
     FROM PKIX1Explicit88
    { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0) id-pkix1-explicit-
      88(18)}

-- Importe de la [RFC4998]
   EvidenceRecord
     FROM 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) };

-- Type de contenu et identifiant d'objet TimeStampedData

IDENTIFIANT D'OBJET id-ct-timestampedData ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) id-
smime(16) id-ct(1) 31 }

TimeStampedData ::= SEQUENCE {
       version              ENTIER { v1(1) },
       dataUri              IA5String FACULTATIF,
       metaData             MetaData FACULTATIF,
       content              CHAINE D'OCTETS FACULTATIF,
       temporalEvidence     Evidence
  }

MetaData ::= SEQUENCE {
```

```
        hashProtected       BOOLEEN,
        fileName            UTF8String FACULTATIF,
        mediaType           IA5String FACULTATIF,
        otherMetaData       Attributes FACULTATIF
    }

Attributes ::= ENSEMBLE TAILLE(1..MAX) DE Attribute      -- conformément à la RFC 5652

Evidence ::= CHOIX {
        tstEvidence     [0] TimeStampTokenEvidence,      -- voir la RFC 3161
        ersEvidence     [RFC2119] EvidenceRecord,        -- voir la RFC 4998
        otherEvidence   [LEGAL] OtherEvidence
    }

OtherEvidence ::= SEQUENCE {
        oeType          IDENTIFIANT D'OBJET,
        oeValue         TOUT DEFINI PAR oeType }

TimeStampTokenEvidence ::= SEQUENCE TAILLE(1..MAX) DE TimeStampAndCRL

TimeStampAndCRL ::= SEQUENCE {
        timeStamp   TimeStampToken,                      -- conformément à la RFC 3161
        crl         CertificateList FACULTATIF           -- conformément à la RFC 5280
    }

FIN
```

Appendice B. Remerciements

Merci à Stephen Kent pour avoir encouragé l'auteur dans les premières étapes de ce travail.

Merci à Russ Housley de sa relecture de ce mémoire, suggérant d'utiles amendements et allouant une valeur aux OID définis.

Des remerciements sont également dus aux autres personnes qui ont relu ce mémoire et aidé à l'améliorer, mais qui préfèrent ne pas être mentionnées.

Adresse de l'auteur

Adriano Santoni
Actalis S.p.A.
Via Taramelli 26
I-20124 Milano
Italy

mél : adriano.santoni@actalis.it