Groupe de travail Réseau
Request for Comments : 5035
RFC mise à jour : 2634
Catégorie : Sur la voie de la normalisation

J. Schaad, Soaring Hawk Consulting
août 2007

Traduction Claude Brière de L'Isle

Mise à jour des services de sécurité améliorée (ESS) : ajout du choix d'algorithme CertID

Résumé
Dans le document d'origine des services de sécurité améliorés pour S/MIME [RFC2634] a été introduit une structure pour lier cryptographiquement à la signature le certificat à utiliser dans la validation ; cette structure incorporait l'utilisation de SHA-1. Le présent document permet à la structure de choisir son algorithme et définit un nouvel attribut à cette fin.

Table des Matières

1. Introduction

Dans le document original des services de sécurité améliorés (ESS, *Enhanced Security Services*) pour S/MIME [RFC2634], a été définie une structure pour lier cryptographiquement avec la signature le certificat à utiliser dans la validation. Cette structure, appelée ESSCertID, identifie un certificat par son hachage. La structure est conçue pour utiliser une valeur de hachage SHA-1. Les récentes attaques contre SHA-1 exigent qu'on définisse un nouvel attribut qui permette l'utilisation d'algorithmes différents. Le présent document s'y emploie.

Le présent document définit la structure ESSCertIDv2 avec un nouvel attribut SigningCertificateV2, qui utilise la structure mise à jour. Le présent document permet à la structure d'avoir une souplesse d'algorithme en incluant un identifiant d'algorithme et en définissant un nouvel attribut signé pour utiliser la nouvelle structure.

Le présent document spécifie la poursuite de l'utilisation de ESSCertID pour assurer la compatibilité quand SHA-1 est utilisé pour qu'il n'y ait pas d'ambiguïté sur le certificat.

1.1 Notation

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" dans le présent document sont à interpréter comme décrit dans la [RFC2119].

1.2 Mise à jour à la RFC2634

Le présent document met à jour le paragraphe 5.4 de la RFC2634. une fois la mise à jour appliquée, le paragraphe révisé aura la structure suivante :

5.4 Définitions d'attribut de certificat de signature
5.4.1 Définition d'attribut de certificat de signature, version 2
5.4.1.1 Identification de certificat version 2
5.4.2 Définition d'attribut de certificat de signature, version 1
5.4.2.1 Identification de certificat version 1

De plus, le module ASN.1 de l'Appendice A est remplacé.

2. Remplacement du paragraphe 5.4

5.4 Définitions d'attribut de certificat de signature

L'attribut de certificat de signature est conçu pour empêcher des attaques en simple substitution et répétition, et pour permettre qu'un ensemble restreint de certificats soit utilisé pour vérifier une signature.

Deux attributs différents existent pour cela du fait d'une faute dans la conception originale. La seule différence substantielle entre les deux attributs est que SigningCertificateV2 permet la souplesse de l'algorithme de hachage, tandis que SigningCertificate force l'utilisation de SHA-1. Avec les récentes avancées de la capacité de créer des collisions de hachage pour SHA-1, il est sage d'avancer plutôt le plus vite que trop tard.

Quand la fonction de hachage SHA-1 est utilisée, l'attribut SigningCertificate DOIT être utilisé. L'attribut SigningCertificateV2 DOIT être utilisé si un algorithme autre que SHA-1 est utilisé et NE DEVRAIT PAS être utilisé pour SHA-1. Les applications DEVRAIENT reconnaître les deux attributs si elles considèrent que SHA-1 est capable de faire la distinction entre deux certificats différents (c'est-à-dire, si la possibilité de collision est suffisamment faible). Si les deux attributs existent dans un seul message, ils sont évalués indépendamment.

Quatre cas existent qui doivent être pris en compte quand on utilise cet attribut, pour un traitement correct :

1. La signature se valide et les hachages correspondent : c'est le cas de succès.

2. La signature se valide et les hachages ne correspondent pas : dans ce cas, le certificat contenait la clé publique correcte, mais le certificat contenant la clé publique n'est pas celui que le signataire avait prévu d'utiliser. Dans ce cas, l'application devrait tenter une recherche d'un certificat différent avec la même clé publique et pour lequel les hachages correpondent. Si aucun certificat ne peut être trouvé, c'est un cas d'échec.

3. La signature échoue à la validation et les hachages correspondent : dans ce cas, on peut supposer que la signature a été modifiée d'une certaine façon. C'est un cas d'échec.

4. La signature échoue à la validation et les hachages ne correspondent pas : dans ce cas, ce peut être que la signature a été modifiée, ou que le mauvais certificat a été utilisé. Les applications devraient tenter une recherche d'un certificat différent qui corresponde à la valeur du hachage dans l'attribut et utiliser le nouveau certificat pour réessayer la validation de la signature.

3. Insertion d'un nouveau paragraphe 5.4.1

5.4.1 Définition de l'attribut Certificat de signature, version 2

L'attribut Certificat de signature est conçu pour empêcher les attaques de simple substitution et répétition, et pour permettre à un ensemble restreint de certificats d'être utilisé pour vérifier une signature.

SigningCertificateV2 est identifié par l'OID :

IDENTIFIANT D'OBJET id-aa-signingCertificateV2 ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) id-aa(2) 47 }

L'attribut a la définition ASN.1 :

```
SigningCertificateV2 ::=  SEQUENCE {
        certs      SEQUENCE DE ESSCertIDv2,
        policies  SEQUENCE DE PolicyInformation FACULTATIF
}
```

certs : contient la liste des certificats qui sont à utiliser pour valider le message. Il DOIT contenir au moins un élément. Le premier certificat identifié dans la séquence d'identifiants de certificats DOIT être le certificat utilisé pour vérifier la signature. Le codage de ESSCertIDv2 pour ce certificat DEVRAIT inclure le champ issuerSerial. Si d'autres contraintes assurent que issuerAndSerialNumber va être présent dans les SignerInfo, le champ issuerSerial PEUT être omis. Le certificat identifié est utilisé durant le processus de vérification de signature. Si le hachage du certificat ne corespond pas au certificat utilisé pour vérifier la signature, la signature DOIT être considérée comme invalide.

Si plus d'un identifiant de certificat est présent dans la séquence de ESSCertIDv2, tous les certificats référencés DOIVENT faire partie de la chaîne de validation de signature. Les certificats peuvent être des certificats d'attribut (limitant les autorisations) ou des certificats de clé publique (limitant la validation de chemin). Le champ issuerSerial (dans la structure ESSCertIDv2) DEVRAIT être présent pour ces certificats, sauf si le client qui valide la signature est supposé avoir un accès facile à tous les certificats nécessaires pour la validation. Si seul le certificat signant est présent dans la séquence, il n'y a pas de restriction sur l'ensemble de certificats utilisés pour valider la signature.

policies : contient une séquence de termes d'informations de politique qui identifie les politiques de certificat que le signataire affirme appliquer au certificat, et selon lesquels le certificat devrait être tenu pour bon. Cette valeur suggère une valeur de politique à utiliser dans la validation du chemin de certification par le consommateur d'assertions. On trouvera la définition de PolicyInformation dans la [RFC3280].

Si il est présent, le certificat SigningCertificateV2 DOIT être un attribut signé; il NE DOIT PAS être un attribut non signé. La CMS définit "SignedAttribute" comme un "SET OF Attribute" *(ensemble d'attributs)*. Un objet SignerInfo NE DOIT PAS inclure plusieurs instances de l'attribut SigningCertificateV2. La CMS définit la syntaxe ASN.1 pour les attributs signés comme incluant des ensembles de valeurs d'attribut de AttributeValue. Un attribut SigningCertificateV2 DOIT inclure seulement une instance de AttributeValue. Il NE DOIT PAS y avoir zéro, une, ou plusieurs instances de AttributeValue présentes dans les ensembles de valeurs d'attribut de AttributeValue.

4. Insertion d'un nouveau paragraphe 5.4.1.1

Insèrer le texte qui suit comme un nouveau paragraphe.

5.4.1.1 Identification de Certificat, version 2

La meilleure façon d'identifier les certificats est un sujet souvent discuté. La structure ESSCertIDv2 fournit deux champs différents qui sont utilisés à cette fin.

Le hachage du certificat entier permet à un vérificateur de voir si le certificat utilisé dans le processus de vérification est le même que celui prévu par le signataire. Les hachages sont pratiques en ce qu'ils sont aussi fréquemment utilisés par les magasins de certificats comme méthode d'indexation et de restitution des certificats. L'utilisation du hachage est exigée par cette structure car la détection de certificats substitués se fonde sur le fait qu'ils vont se transposer en des valeurs de hachage différentes.

La paire producteur/numéro de de série est la méthode d'identification des certificats utilisée dans la [RFC3280]. Le présent document impose une restriction pour les certificats qui est que le nom distinctif du producteur doit être présent. La paire producteur/numéro de série devrait donc normalement être suffisante pour identifier le certificat signant correct. (Cela suppose que le même nom de producteur n'est pas réutilisé d'après l'ensemble d'ancres de confiance.) La paire producteur/numéro de de série peut être mémorisée dans le champ "sid" de l'objet SignerInfo. Cependant, le champ sid n'est pas couvert par la signature. Dans les cas où la paire producteur/numéro de série n'est pas utilisée dans le sid ou si la paire producteur/numéro de série a besoin d'être signée, elle DEVRAIT être placée dans le champ issuerSerial de la structure ESSCertIDv2.

Les certificats d'attribut et les certificats supplémentaires de clé publique contenant des informations n'ont pas une paire producteur/numéro de série représentée dans l'objet SignerInfo. Quand un certificat d'attribut ou un certificat de clé publique supplémentaire n'est pas inclus dans l'objet SignedData, il devient plus difficile d'obtenir l'ensemble correct de certificats sur la seule base du hachage du certificat. Pour cette raison, ces certificats DEVRAIENT être identifiés par l'objet IssuerSerial.

Le présent document définit un identifiant de certificat comme :

```
ESSCertIDv2 ::=  SEQUENCE {
   hashAlgorithm          AlgorithmIdentifier
        DEFAULT {algorithm id-sha256},
   certHash               Hash,
   issuerSerial           IssuerSerial FACULTATIF
}

Hash ::= CHAINE D'OCTETS
IssuerSerial ::= SEQUENCE {
   issuer                 GeneralNames,
   serialNumber       CertificateSerialNumber
}
```

Les champs de ESSCertIDv2 sont définis comme suit :

hashAlgorithm : contient l'identifiant de l'algorithme utilisé pour calculer certHash.

certHash : est calculé sur le certificat entier codé en DER (incluant la signature) en utilisant l'algorithme spécifié par hashAlgorithm.

issuerSerial : contient l'identification du certificat. Le issuerSerial va normalement être présent sauf si la valeur peut être déduite d'autres informations (par exemple, le champ sid de l'objet SignerInfo).

Les champs de IssuerSerial sont définis comme suit :

issuer : contient le nom du producteur du certificat. Pour les certificats non d'attribut, le producteur DOIT contenir seulement le nom de producteur du certificat codé dans le choix directoryName de GeneralNames. Pour les certificats d'attribut, le producteur DOIT contenir le champ Nom de producteur provenant du certificat d'attribut.

serialNumber : contient le numéro de série qui identifie de façon univoque le certificat pour le producteur.

5. Insertion d'un nouveau paragraphe 5.4.2

(Note : cette Section ne présente pas de nouveau matériel. Elle contient le paragraphe original du paragraphe 5.4 de la [RFC2634], qui est conservé avec des changements mineurs pour assurer la rétro compatibilité.)

Insérer le texte suivant comme un nouveau paragraphe :

5.4.2 Définition de l'attribut Certificat de signature, version 1

L'attribut de certificat de signature est conçu pour empêcher la simple attaque de substitution et de reproduction, et pour permettre d'utiliser un ensemble restreint de certificats d'autorisation dans la vérification de signature.

La définition de SigningCertificate est :

```
SigningCertificate ::=  SEQUENCE {
   certs        SEQUENCE OF ESSCertID,
   policies     SEQUENCE OF PolicyInformation FACULTATIF
}
```

id-aa-signingCertificate IDENTIFIANT D'OBJET ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) id-aa(2) 12 }

Le premier certificat identifié dans la séquence d'identifiants de certificats DOIT être le certificat utilisé pour vérifier la signature. Le codage de ESSCertID pour ce certificat DEVRAIT inclure le champ issuerSerial. Si d'autres contraintes assurent que issuerAndSerialNumber sera présent dans les SignerInfo, le champ issuerSerial PEUT être omis. Le certificat identifié est utilisé durant le processus de vérification de signature. Si le hachage du certificat ne correspond pas au certificat utilisé pour vérifier la signature, la signature DOIT être considérée comme invalide.

Si plus d'un certificat est présent dans la séquence de ESSCertID, les certificats après le premier limitent l'ensemble des certificats qui sont utilisés durant la validation. Les certificats peuvent être des certificats d'attribut (limitant les autorisations) ou des certificats de clé publique (limitant la validation de chemin). Le champ issuerSerial (dans la structure ESSCertID) DEVRAIT être présent pour ces certificats, sauf si le client qui valide la signature est supposé avoir un accès facile à tous les certificats exigés pour la validation. Si seul le certificat de signature est présent dans la séquence, il n'y a pas de restriction sur l'ensemble des certificats d'autorisation utilisés pour valider la signature.

La séquence des termes d'informations de politique identifie les politiques de certificat que le signataire affirme appliquer au certificat, et sur lesquelles le certificat devrait s'appuyer. Cette valeur suggère une valeur de politique à utiliser dans la validation du chemin de certification du consommateur d'assertions.

Si il est présent, l'attribut SigningCertificate DOIT être un attribut signé ; il NE DOIT PAS être un attribut non signé. La CMS définit SignedAttributes comme un SET OF Attribute *(ensemble d'attributs)*. Un SignerInfo NE DOIT PAS inclure plusieurs instances de l'attribut SigningCertificate. La CMS définit la syntaxe ASN.1 pour les attributs signés comme incluant des attrValues SET OF AttributeValue. Un attribut SigningCertificate DOIT inclure seulement une instance de AttributeValue. Il NE DOIT PAS y avoir zéro, une ou plusieurs instances de AttributeValue présentes dans le attrValues SET OF AttributeValue.

6. Insertion d'un nouveau paragraphe 5.4.2.1

(Note : cette Section ne présente pas de nouveau matériel. Elle contient le paragraphe original du paragraphe 5.4.1 de la [RFC2634], qui est conservé avec des changements mineurs pour assurer la rétro compatibilité.)

Supprimer l'ancien paragraphe 5.4.1 et insérer ce qui suit comme nouveau texte.

5.4.2.1 Identification de Certificat, version 1

Les certificats sont identifiés de façon univoque en utilisant les informations de la structure ESSCertID. On trouvera les explications au paragraphe 5.4.1.1.

Le présent document définit un identifiant de certificat comme :

```
ESSCertID ::=  SEQUENCE {
    certHash              Hash,
    issuerSerial          IssuerSerial FACULTATIF
}
```

Les champs de ESSCertID sont définis comme suit :

certHash : est calculé sur le certificat codé en DER entier (incluant la signature) en utilisant l'algorithme SHA-1.

issuerSerial : contient l'identification du certificat. Ce champ va normalement être présent sauf si la valeur peut être déduite d'autres informations (par exemple, le champ sid de l'objet SignerInfo).

Les champs de IssuerSerial sont expliquéss au paragraphe 5.4.1.1

7. Considérations pour la sécurité

Le présent document est conçu pour taiter le problème de sécurité d'un certificat substitué utilisé par le valideur. Si le valideur utilise un certificat différent de celui du signataire, le valideur ne peut pas obtenir le résultat correct. Un exemple en serait que le certificat original a été révoqué et qu'un nouveau certificat avec la même clé publique a été produit pour un individu différent. Comme le champ producteur/numéro de série n'est pas protégé, l'attaquant pourrait le remplacer et le faire pointer sur le nouveau certificat pour que la validation réussisse.

Les attributs définis dans le présent document sont à placer dans des endroits qui sont protégés par la signature. Cet attribut ne donne aucune sécurité supplémentaire si il est placé dans une localisation non signée ou non authentifiée.

Les attributs définis dans ce document permettent à un signataire de choisir un algorithme de hachage pour identifier un certificat. Un algorithme de hachage mal choisi peut fournir une protection inadéquate contre la substitution de certificat ou résulter en un déni de service pour cette protection. En employant les attributs définis dans la présente spécification avec le même algorithme de hachage utilisé pour la signature de message, l'envoyeur peut s'assurer que ces attributs fournissent une sécurité en rapport.

Comme les receveurs doivent prendre en charge l'algorithme de hachage pour vérifier la signature, choisir le même algorithme de hachage augmente aussi la probabilité que l'algorithme de hachage soit supporté dans le contexte de l'identification de certificat. Noter qu'un algorithme de hachage non pris en charge pour l'identification de certificat n'empêche pas de valider le message, mais dénie au receveur du message la protection contre la substitution de certificat.

Pour assurer que les mises en œuvre traditionnelles fournissent la protection contre la substitution de certificat, il est permis aux clients d'inclure à la fois ESScertID et ESScertIDv2 dans le même message. Comme ces attributs sont générés et évalués indépendamment, les contenus pourraient éventuellement être en conflit. Spécifiquement, lorsque un signataire a plusieurs certificats contenant la même clé publique, les deux attributs pourraient spécifier des certificats de signature différents. Le résultat du traitement de la signature peut varier selon le certificat utilisé pour valider la signature.

Les receveurs qui tentent d'évaluer les deux attributs peuvent choisir de rejeter un tel message.

8. 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)

[RFC2634] P. Hoffman, éd., "Services de sécurité améliorés pour S/MIME", juin 1999. (*MàJ par* RFC5035) *(P.S.)*

[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)

[RFC3629] F. Yergeau, "UTF-8, un format de transformation de la norme ISO 10646", STD 63, novembre 2003.

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

Appendice A. Module ASN.1

Remplacer le module ASN.1 de la RFC 2634 par celui-ci :

```
ExtendedSecurityServices-2006
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-ess-2006(30) }
DEFINITIONS DES ÉTIQUETTES IMPLICITES ::=
DÉBUT

IMPORTATIONS
-- Syntaxe de message cryptographique (CMS) [RFC3852]
   ContentType, IssuerAndSerialNumber, SubjectKeyIdentifier
```

```
 DE CryptographicMessageSyntax2004 { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
                    modules(0) cms-2004(24)}
-- Certificat PKIX et profil de CRL, paragraphe A.1 Module explicitement étiqueté
--  Syntaxe 1988          [RFC3280]
   AlgorithmIdentifier, CertificateSerialNumber
 DE PKIX1Explicit88 { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
                    id-pkix1-explicit(18) }

-- Certificat PKIX et profil de CRL, paragraphe A.2 Module implicitement étiqueté
--  Syntaxe 1988          [RFC3280]
   PolicyInformation, GeneralNames
 DE PKIX1Implicit88 {iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
                    id-pkix1-implicit(19)};

-- Services de sécurité étendus
-- La construction "SEQUENCE SIZE (1..MAX) OF" apparaît dans plusieurs constructions ASN.1 de ce module. Une
SEQUENCE ASN.1 valide peut avoir zéro, une ou plusieurs entrées. La construction SIZE (1..MAX) contraint la
SEQUENCE à avoir au moins une entrée. MAX indique que la limite supérieure est non spécifiée. Les mises en œuvre sont
libres de choisir une limite supérieure qui convient à leur environnement.

-- UTF8String ::= [UNIVERSAL 12] CHAINE D'OCTETS IMPLICITE

-- Les contenus sont formatés comme décrit dans la [RFC3629]

-- paragraphe 2.7

ReceiptRequest ::= SEQUENCE {
  signedContentIdentifier ContentIdentifier,
  receiptsFrom ReceiptsFrom,
  receiptsTo SEQUENCE SIZE (1..ub-receiptsTo) OF GeneralNames
}

ub-receiptsTo ENTIER ::= 16

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

ContentIdentifier ::=  CHAINE D'OCTETS

id-aa-contentIdentifier IDENTIFIANT D'OBJET ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
                    smime(16) id-aa(2) 7}

ReceiptsFrom ::= CHOIX {
 allOrFirstTier [0] AllOrFirstTier,                    -- anciennement "allOrNone [0]AllOrNone"
 receiptList [1] SEQUENCE OF GeneralNames
}

AllOrFirstTier ::= ENTIER {                           -- anciennement AllOrNone
 allReceipts (0),
 firstTierRecipients (1)
}

-- paragraphe 2.8

Receipt ::= SEQUENCE {
  version ESSVersion,
  contentType ContentType,
  signedContentIdentifier ContentIdentifier,
  originatorSignatureValue CHAINE D'OCTETS
}
```

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

ESSVersion ::= ENTIER { v1(1) }

-- paragraphe 2.9

ContentHints ::= SEQUENCE {
  contentDescription UTF8String (SIZE (1..MAX)) FACULTATIF,
  contentType ContentType
}

id-aa-contentHint IDENTIFIANT D'OBJET ::= { iso(1) member-body(2) us(840)  rsadsi(113549) pkcs(1) pkcs-9(9)
                        smime(16) id-aa(2) 4}

-- paragraphe 2.10

MsgSigDigest ::= CHAINE D'OCTETS

id-aa-msgSigDigest IDENTIFIANT D'OBJET ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
                        smime(16) id-aa(2) 5}

-- paragraphe 2.11

ContentReference ::= SEQUENCE {
  contentType ContentType,
  signedContentIdentifier ContentIdentifier,
  originatorSignatureValue  CHAINE D'OCTETS
}

id-aa-contentReference IDENTIFIANT D'OBJET ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
                        smime(16) id-aa(2) 10 }

-- paragraphe 3.2

ESSSecurityLabel ::= SET {
  security-policy-identifier SecurityPolicyIdentifier,
  security-classification SecurityClassification FACULTATIF,
  privacy-mark ESSPrivacyMark FACULTATIF,
  security-categories SecurityCategories FACULTATIF
}

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

SecurityClassification ::= ENTIER {
  unmarked (0),
  unclassified (1),
  restricted (2),
  confidential (3),
  secret (4),
  top-secret (5)
}(0..ub-integer-options)

ub-integer-options ENTIER ::= 256

ESSPrivacyMark ::= CHOIX {
   pString      PrintableString (SIZE (1..ub-privacy-mark-length)),
```

```
    utf8String   UTF8String (SIZE (1..MAX))
}

ub-privacy-mark-length ENTIER ::= 128

SecurityCategories ::= SET SIZE (1..ub-security-categories) OF SecurityCategory

ub-security-categories ENTIER ::= 64

SecurityCategory ::= SEQUENCE {
  type    [0] IDENTIFIANT D'OBJET,
  valeur [1] ANY DEFINED BY type
}

-- Note : La syntaxe SecurityCategory susmentionnée produit des codages hexadécimaux identiques à ceux de la syntaxe de
          SecurityCategory suivante qui est documentée dans la spécification X.411 :
--
-SecurityCategory ::= SEQUENCE {

--     type    [0]  SECURITY-CATEGORY,
--     valeur [1]  ANY DEFINED BY type }
--
--MACRO SECURITY-CATEGORY :: =
--DÉBUT
--TYPE NOTATION ::= type | vide
--VALUE NOTATION ::= valeur (VALEUR D'IDENTIFIANT D'OBJET)
--FIN

-- paragraphe 3.4

EquivalentLabels ::= SEQUENCE OF ESSSecurityLabel

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

-- paragraphe 4.4

MLExpansionHistory ::= SEQUENCE
     SIZE (1..ub-ml-expansion-history) OF MLData

id-aa-mlExpandHistory IDENTIFIANT D'OBJET ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
                        smime(16) id-aa(2) 3 }

ub-ml-expansion-history ENTIER ::= 64  MLData ::= SEQUENCE {
  mailListIdentifier EntityIdentifier,
  expansionTime GeneralizedTime,
  mlReceiptPolicy MLReceiptPolicy FACULTATIF
}

EntityIdentifier ::= CHOIX {
  issuerAndSerialNumber IssuerAndSerialNumber,
  subjectKeyIdentifier SubjectKeyIdentifier
}

MLReceiptPolicy ::= CHOIX {
  none [0] NUL,
  insteadOf [1] SEQUENCE SIZE (1..MAX) OF GeneralNames,
  inAdditionTo [2] SEQUENCE SIZE (1..MAX) OF GeneralNames
}
```

```
-- paragraphe 5.4

SigningCertificate ::=  SEQUENCE {
   certs        SEQUENCE OF ESSCertID,
   policies     SEQUENCE OF PolicyInformation FACULTATIF
}

id-aa-signingCertificate IDENTIFIANT D'OBJET ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
                        smime(16) id-aa(2) 12 }

SigningCertificateV2 ::=  SEQUENCE {
   certs        SEQUENCE OF ESSCertIDv2,
   policies     SEQUENCE OF PolicyInformation FACULTATIF
}

id-aa-signingCertificateV2 IDENTIFIANT D'OBJET ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
                        pkcs9(9) smime(16) id-aa(2) 47 }

id-sha256  IDENTIFIANT D'OBJET ::=  { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3)
                        nistalgorithm(4) hashalgs(2) 1 }

ESSCertIDv2 ::= SEQUENCE {
    hashAlgorithm           AlgorithmIdentifier
         DEFAULT {algorithm id-sha256},
    certHash                Hash,
    issuerSerial            IssuerSerial FACULTATIF
}

ESSCertID ::=  SEQUENCE {
    certHash                Hash,
    issuerSerial            IssuerSerial FACULTATIF
}

Hash ::= CHAINE D'OCTETS
IssuerSerial ::= SEQUENCE {
    issuer                  GeneralNames,
    serialNumber            CertificateSerialNumber
}

FIN -- de ExtendedSecurityServices-2006
```

Adresse de l'auteur

Jim Schaad
Soaring Hawk Consulting
PO Box 675
Gold Bar, WA 98251
USA
mél : jimsch@exmsft.com

Déclaration complète de droits de reproduction

Propriété intellectuelle