rfc5208

Syntaxe et conteneur pour les informations de clé privée

Définitions

Les informations de clé privée incluent une clé privée pour un algorithme à clé publique spécifié, et un jeu d'attributs. La syntaxe de message cryptographique, dénifis dans la rfc5652, peut être utilisé pour signer numériquement, hasher, authentifier ou chiffrer le type de contenu de clé asymétrique.

La liste suivante sont les mises à jours de la rfc5208 :

- PrivateKeyInfo a été changé en OneAsymmetricKey. Cela reflète l'ajout du champ publicKey pour permettre aux 2 parties de la clé asymétrique d'être transportés séparément.
- Définition du type de contenu CMS asymmetric Key Package
- Suppression de la redondance IMPLICIT des attributs
- Ajout de publicKey à OneAsymmetricKey et mise à jour du numéro de version
- Ajout du support des attributs PKCS #9
- Ajout d'une discussion sur la compatibilité avec d'autres formats de clé privée
- Ajout des pré-requis pour le jeu de règle d'encodage
- Changement des imports depuis PKCS #5
- Remplacement de ALGORITHM-IDENTIFIER avec ALGORITHM depuis la rfc 5912
- Enregistrement du type de média application/pkcs8 et de l'extension de fichier .p8

Asymmetric Key Package CMS Content Type

Le type de contenu CMS paquet de clé asymétrique est utilisé pour transférer une ou plusieurs clé asymétrique d'un partie à un autre. Un paquet de clé asymétrique peut être encapsulé dans un ou plusieurs types de contenu CMS de protection. Les anciennes versions de cette spécification de spécifiaient pas de jeu de règle d'encodage particulier, mais les générateurs devraient utiliser DER et les destinataires doivent supporter BER, qui inclus également DER.

Ce type de contenu a la syntaxe suivante :

```
ct-asymmetric-key-package CONTENT-TYPE ::=
 { AsymmetricKeyPackage IDENTIFIED BY id-ct-KP-aKeyPackage }

id-ct-KP-aKeyPackage OBJECT IDENTIFIER ::=
 { joint-iso-itu-t(2) country(16) us(840) organization(1)
  gov(101) dod(2) infosec(1) formats(2)
  key-package-content-types(78) 5
 }

AsymmetricKeyPackage ::= SEQUENCE SIZE (1..MAX) OF OneAsymmetricKey

OneAsymmetricKey ::= SEQUENCE {
 version Version,
```

```
 privateKeyAlgorithm PrivateKeyAlgorithmIdentifier,
 privateKey PrivateKey,
 attributes [0] Attributes OPTIONAL,
 ...,
 [[2: publicKey [1] PublicKey OPTIONAL ]],
 ...
}

PrivateKeyInfo ::= OneAsymmetricKey

 - PrivateKeyInfo est utilisé par [P12]. Si un élément taggé en version 2 est utilisé,
 - La version doit être v2, sinon la version devrait être 1. Quand v1 est utilisé,
 - PrivateKeyInfo est le même que dans la [RFC5208].

Version ::= INTEGER { v1(0), v2(1) } (v1, ..., v2)

PrivateKeyAlgorithmIdentifier ::= AlgorithmIdentifier
 { PUBLIC-KEY,
  { PrivateKeyAlgorithms } }

PrivateKey ::= OCTET STRING
 - Le contenu varie en fonction du type de clé. L'identifiant d'algorithme dicte le format de la clé

PublicKey ::= BIT STRING
 - Le contenu varie en fonction du type de clé. L'identifiant d'algorithme dicte le format de la clé

Attributes ::= SET OF Attribute { { OneAsymmetricKeyAttributes } }
```

AsymmetricKeyPackage contient un ou plusieurs élément OneAsymmetricKey.

La syntaxe de OneAsymmetricKey accueille un numéro de version, une indication de l'algorithme asymétrique à utiliser avec la clé privée, une clé privée, des attributs optionnels (par ex. userCertificate de X.520), et une clé publique optionnelle. En général, doit la clé publique, soit le certificat est présent. Dans de très rare cas les 2 sont présents, vu qu'ils incluent chacun une copie de la clé publique. OneAsymmetricKey renomme la syntaxe PrivateKeyInfo définie dans la rfc5208. Le nouveau nom reflète mieux la capacité de s'occuper des composant clé publique et privée. La compatibilité avec PrivateKeyInfo est préservée via le numéro de version. Les champs dans OneAsymmetricKey sont utilisé comme suit :

version Identifie la version de OneAsymmetricKey. Si publicKey est présent, la version est v2 sinon v1.

privateKeyAlgorithm Identifie l'algorithme de clé privée et optionnellement les paramètres associés avec la paire de clé asymétrique. L'algorithme est identifiée par un OID et le format des paramètres dépend de cet OID, mais le jeu d'objets privateKeyAlgorithms restreins les OID permis. La valeur placée dans PrivateKeyAlgorithmIdentifier est la valeur d'un algorithme utilisé avec la clé privée.

privateKey Est une chaîne d'octets qui contiennent la valeur de la clé privée. L'interprétation du contenu est définis dans l'enregistrement de l'algorithme de clé privée. Par exemple, une clé DSA est un entier, une clé RSA est représentée en RSAPrivateKey comme définis dans la rfc3447, et une clé ECC est représentée comme ECPrivateKey comme définie dans la rfc5915.

attributes est optionnel et contient les informations correspondantes à la clé publique (par ex : les certificats). Les champs attributes utilise la classe ATTRIBUTE qui est restreins par le jeu d'objet OneAsymmetricKeyAttributes. Les attributs de la rfc2985 peuvent être supportés.

publicKey est optionnel. Si présent, il contient la clé publique encodée en chaîne de bits. L structure dans cette chaîne dépend de privateKeyAlgorithm. Par exemple, une clé DSA est un entier. Noter que les clé publiques RSA sont inclues dans RSAPrivateKey, comme dans rfc 3447, et les clé publiques ECC sont inclues dans ECPrivateKey, comme dans rfc 5915.

Informations de clé privée chiffrée

Cette section donne la syntaxe pour les informations de clé privée chiffrée qui sont utilisées par [P12] :

```
EncryptedPrivateKeyInfo ::= SEQUENCE {
 encryptionAlgorithm EncryptionAlgorithmIdentifier,
 encryptedData EncryptedData }

EncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
 { CONTENT-ENCRYPTION,
  { KeyEncryptionAlgorithms } }

EncryptedData ::= OCTET STRING
```

encryptionAlgorithm Identifie l'algorithme sous lequel les informations de clé privée sont chiffrées.

encryptedData Est le résulta du chiffrement des informations de clé privée.

Le processus de chiffrement est constitué des étapes suivantes :

1. Les informations de clé privée sont encodées. Les générateurs devraient utiliser DER et les destinataires doivent supporter BER, qui inclus également DER
2. Le résultat de la première étape est chiffré avec la clé secrète pour donner une chaîne d'octets, le résultat du processus de chiffrement.

Protection de AsymmetricKeyPackage

Les types de contenu de protection CMS peuvent être utilisés pour fournir une sécurité à AsymmetricKeyPackage :

SignedData Peut être utilisé pour appliquer une signature numérique à AsymmetricKeyPackage

EncryptedData Peut être utilisé pour chiffrer AsymmetricKeyPackage avec un chiffrement symétrique, où l'émetteur et le destinataire partagent déjà la clé de chiffrement nécessaire.

EnvelopedData Peut être utilisé pour chiffrer AsymmetricKeyPackage avec un chiffrement symétrique, où l'émetteur et le destinataire ne partagent pas la clé de chiffrement nécessaire.

AuthenticatedData Peut être utilisé pour protéger AsymmetricKeyPackage avec MAC, où les informations de gestion de clé sont manipulées d'une manière similaire à EnvelopedData.

AuthEnvelopedData Peut être utilisé pour protéger AsymmetricKeyPackage avec des algorithmes qui supportent de chiffrement authentifié, où les informations de gestion de clé sont manipulées d'une manière similaire à EnvelopedData.

Considérations d'autres format de clé privée

Ce document définis la syntaxe et les sémantiques pour un type de contenu qui échange des clé privées asymétriques. Il y a 2 autres formats qui ont été utilisés pour le transport de clé privée asymétiques :

Personal Information Exchange Est une syntaxe transfert pour les informations d'identité personnelle, incluant les clés privées, certificats, divers autres secrets, et des extensions. OneAsymmetricKey, PrivateKeyInfo, et EncryptedPrivateKeyInfo peuvent être inclus dans un message p12. Les informations de clé privée, OneAsymmetricKey et PrivateKeyInfo, sont inclus dans le KeyBag BAG-TYPE P12. EncryptedPrivateKeyInfo est inclu dans le pkcs8ShroudedKeyBag BAG-TYPE P12. Dans les implémentations courantes, les extensions pfx et p12 peuvent être utilisées.

Microsoft's private-key l'extension pvk est utilisé pour des stockages locaux. pvk et p12 ne sont pas interchangeables.

Pour extraire les information de clé privée du AsymmetricKeyPackage les couches d'encapsulations doivent être enlevées. Au minimum, la couche ContentInfo doit être enlevée. Si le AsymmetricKeyPackage est encapsulée dans une donnée signée, puis les couches SignedData et EncapsulatedContentInfo doivent également être supprimées. C'est la même chose pour EnvelopedData, EncryptedData, et

AuthenticatedData. Une fois toutes les couches supprimées, il y a autant de jeux d'information de clé privée qu'il y a de structure OneAsymmetricKey. OneAsymmetricKey et PrivateKeyInfo ont la même structure, donc peuvent être sauvées en tant que fichier p8 ou copiés dans un BAG-TYPE KeyBag p12. Supprimer les couches de sécurité encapsulantes va invalider toute signature et peut exposer la clé.

Les fichiers p8 sont parfois encodés PEM, et dans ce cas utilisent l'extension pem. L'encodage PEM est soit l'encodage Base64 ou de type EncryptedPrivateKeyInfo encodé DER inclus entre :

```
---BEGIN ENCRYPTED PRIVATE KEY---
---END ENCRYPTED PRIVATE KEY---
```

ou l'encodage Base64 du PrivateKeyInfo encodé DER entre :

```
---BEGIN PRIVATE KEY---
---END PRIVATE KEY---
```

Media application/pkcs8

Type name : application

Subtype name : pkcs8

Required parameters : None

Optional parameters : None

Encoding considerations : binary

Security considerations : Carries a cryptographic private key

Interoperability considerations : The PKCS #8 object inside this media type MUST be DER-encoded PrivateKeyInfo.

Published specification : RFC 5958

Applications which use this media type : Any MIME-compliant transport that processes asymmetric keys.

Additional information : Magic number(s) : None

File extension(s) : .p8

Macintosh File Type Code(s) : Person & email address to contact for further information : Sean Turner <turners@ieca.com>

Restrictions on usage : none