rfc4210

Protocole de gestion de certificat des infrastructures à clé publique X.509

Introduction

Ce document décris le protocole de gestion de certificat (CMP) des infrastructures à clé publique X.509. Les messages de protocole sont définis pour la création et la gestion des certificats. Le terme X.509 dans ce document réfère à un certificat X.509v3.

Définition des entités de PKI

Les entités impliqués dans la gestion de PKI incluent l'entité finale et l'autorité de certification. Une autorité d'enregistrement peut également être utilisé.

Sujet et entités finales

Le terme "subject" est utilisé ici pour référer à l'entité pour lequel le certificat est fournis, typiquement nommé dans le sujet ou le champ subjectAltName d'un certificat. Quand on souhaite distinguer les outils et/ou logiciels utilisés par le sujet (ex : un module de gestion de certificat local), on utilise le terme "subject equipment". En général, le terme "end entity" (EE), au lieu de "subject" est préféré pour éviter les confusions avec le nom du champ. Il est important de noter que les entités finales n'incluent pas seulement des utilisateurs humains d'applications, mais également les applications elles-mêmes. Ce facteur influence les protocoles que les opérations de gestion de PKI utilisent ; par exemple, une application est plus susceptibles de savoir exactement quelles extensions de certificat sont nécessaires que ne le sont les utilisateurs humains.

Les entités de gestion de PKI sont également des entités finales dans le sens qu'ils sont parfois nommés dans le sujet ou le subjectAltName d'un certificat ou un cross-certificate. Quand approprié, le terme "end entity" sera utilisé pour référer aux entités finales qui ne sont pas des entités de gestion de PKI.

Toute entité finale nécessite un accès local sécurisé à certaines informations – au minimum, son propre nom et clé privée, le nom d'une CA qui est directement trusté, et la clé publique de cette CA (ou une empreinte de la clé publique où une version d'un certificat auto-fournis est disponible quelque part). Les implémentations peuvent utiliser un stockage local sécurisé pour plus que ce minimum. La forme de stockage varie également – de simples fichier à des jetons cryptographiques inviolables. L'information stockée dans un tel stockage local est référé ici à l'environnement de sécurité personnel (PSE). Bien que les formats PSE sont au-delà du scope de ce document, un format générique d'échange pour les PSE est définis ici : un message réponse de certification peut être utilisé.

Autorité de certification

L'autorité de certification peut ou non être un vrai tier de confiance du point de vue de l'entité finale. Très souvent, la CA appartient à la même organisation que les entités finales qu'elle supporte.

De même, on utilise le terme CA pour référer à l'entité nommée dans le champ issuer d'un certificat. Quand il est nécessaire de distinguer les outils software ou hardware utilisé par la CA, on utilise le terme d'équipement CA. L'équipement CA inclus souvent un composant

offline et un composant online, avec la clé privée de la CA uniquement disponible au composant offline.

On utilise le terme "root CA" pour indiquer une CA qui est directement trusté par une entité finale ; c'est à dire, acquérir de manière sécurité la valeur d'une clé publique d'une CA root nécessite certaines étapes out-of-band. Ce terme n'implique pas qu'une CA root soit nécessairement tout en haut d'une hiérarchie, simplement que la CA en question est trustée directement.

Une CA subordonnée est une CA qui n'est pas une CA root pour l'entité finale. Souvent, une CA subordonnée ne sera une CA root pour aucune entité, mais ce n'est pas mandatoire.

En plus des entités finales et des CA, de nombreux environnements font appel à un autorité d'enregistrement (RA) séparé de l'autorité de certification. Les fonctions du RA varie d'un cas à un autre mais peut inclure une authentification personnelle, la distribution de jetons, reporter la révocation, assignement de nom, génération de clé, archivage de paires de clé, etc.

Ce document considère le RA comme composant optionnel : quand il n'est pas présent, la CA est supposé gérer les fonctions du RA, donc les protocoles de gestion de PKI sont les même du point de vue de l'entité finale. De même, on distingue le RA et les outils utilisés (les équipements RA).

Noter qu'un RA est lui-même une entité finale. On assume que tous les RA sont en fait des entités finales certifiés et que les RA ont des clés privées qui sont utilisable pour la signature. La manière dont un équipement CA identifie des entités finales comme RA est un problème d'implémentation (par ex : Ce document ne spécifie pas d'opération de certification RA spécial). On ne mandate pas que le RA est certifié par la CA avec laquelle il inter-agit (donc un RA peut travailler avec plus d'une CA alors qu'il a été certifié une seule fois).

Dans certaines circonstances, les entités finales vont communiquer directement avec une CA même quand un RA est présent. Par exemple, pour un enregistrement initial et/ou une certification, le sujet peut utiliser sont RA, mais communiquer directement avec le CA pour rafraîchir son certificat.

Pré-requis de gestion de PKI

Les protocoles donnés ici nécessitent les pré-requis suivant pour la gestion de PKI :

1. La gestion de la PKI doit se conformer aux standards ISO/IEC 9594-8/ITU-T X.509
2. Il doit être possible de mettre à jour régulièrement une paire de clé sans affecter une autre paire de clé.
3. L'utilisation de la confidentialité dans les protocoles de gestion de PKI doit être conservé à un minimum pour pouvoir simplifier l'acceptation dans les environnements où une forte confidentialité peut causer des problèmes réguliers.
4. Les protocoles de gestion de PKI doivent permettre d'utiliser différents algorithmes cryptographiques (incluant RSA, DSA, MD5 et SHA-1). Cela signifie qu'une CA, RA, ou entité finale donné, en principe, utilise les algorithmes qui lui conviennent pour ses propres paires de clés.
5. Les protocoles de gestion de PKI ne doivent pas pré-inclure la génération de paires de clé par l'entité finale concernée, par un RA, ou par une CA. La génération de clé peut également se produire n'importe où, mais dans le but de la gestion de PKI la clé est d'abord présente dans une entité finale, RA, ou CA.
6. Les protocoles de gestion de PKI doivent supporter la publication de certificats par l'entité finale concernée, par une RA, ou par une CA. Différences implémentations et différents environnements peuvent choisir n'importe quel approche ci-dessus.
7. Les protocoles de gestion de PKI doivent supporter la production de listes de révocation de certificat en permettant aux entités finales certifiées de créer des requêtes pour la révocation des certificats. Cela doit être fait de telle manière qu'une attaque DOS, qui est possible, ne soit pas simple.
8. Les protocoles de gestion de PKI doivent être utilisables sur une variété de mécanismes de transport, incluant les mails, http, TCP/IP et ftp.
9. L'autorité finale pour la création des certificats incombe à la CA. Aucune RA ou équipement d'entité finale ne peut supposer qu'un certificat fournis par une CA contienne ce qui a été demandé ; Une CA peut altérer les valeurs de champs de certificat ou peut en ajouter, supprimer, ou modifier les extensions en accord à ses stratégies. En d'autres terme, toutes les entités PKI (entités finales, CA et CA) doivent être capable de manipuler les réponses aux requêtes pour les certificats dans lequel le certificat émis est différent de la demande. Noter qu'une stratégie peut dicter qu'une CA ne publie pas ou ne distribue pas le certificat tant que l'entité qui a fait la demande n'a pas accepté le certificat créé (généralement par un message certConf).

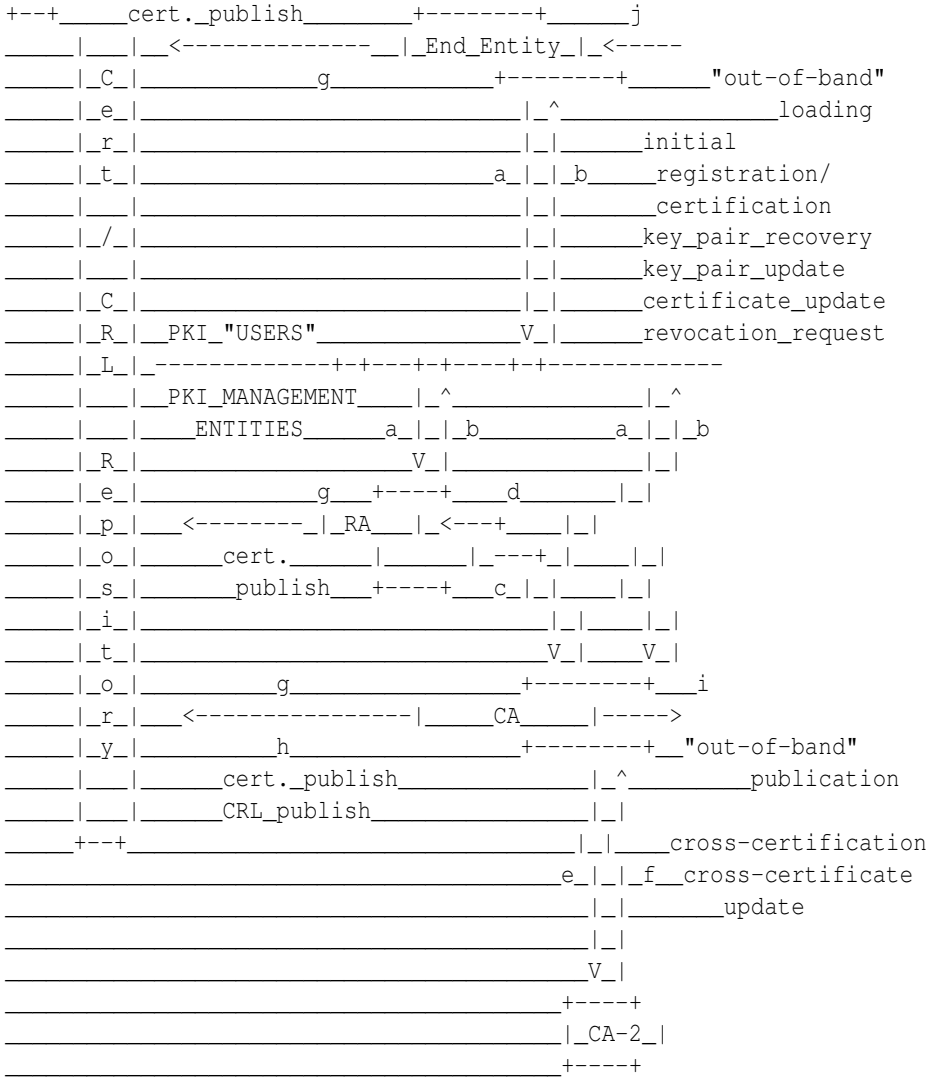
10. Une grâce, un changement planifié d'une paire de clé de CA non-compromise doit être supporté (noter que si la clé CA est compromise, la ré-initialisation doit être effectuée pour toutes les entités dans le domaine de la CA). Une entité finale dont le PSE contient la nouvelle clé publique CA doit également être capable de vérifier les certificats vérifiables en utilisant l'ancienne clé publique. Les entités finales qui trust directement l'ancienne paire de clé CA doivent également être capable de vérifier les certificat signés en utilisant la nouvelle clé privée de la CA.

11. Les fonctions d'un RA peut, dans certaines implémentations ou environnements, être géré par la CA elle-même. Les protocoles doivent être conçus pour que les entités finales utilisent le même protocole sans regarder si la communication est avec une CA ou une RA. Naturellement, l'entité finale doit utiliser la bonne clé public RA ou CA pour protéger la communication.

12. Pour une EE demandant un certificat contenant un valeur de clé publique donnée, celle-ci doit être prête à démontrer la possession de la clé privée correspondant.

Opérations de gestion de PKI

Le diagramme suivant affiche la relation entre les entités définies plus haut en terme d'opération de gestion de PKI. Les lettres dans le diagramme indiquent les protocoles dans le sens qu'un jeu définis de message de gestion de PKI peut être envoyé.



1. Établissement de CA : en établissant une nouvelle CA, certaines étapes sont requises (par ex. la production de crl initiales, l'export de la clé publique de la CA).

2. Initialisation de l'entité finale : inclus l'import d'une clé publique CA racine et la demande d'information sur les options supportées par une entité de gestion de PKI.

3. Certification : Diverses opérations résultent de la création de nouveaux certificats :

1. **Enregistrement/certification initiale :** C'est le processus par lequel une entité finale se fait connaître auprès d'une CA ou RA, avant que la CA ne fournisse un certificat pour l'entité finale. Le résultat final de ce processus (Quand il est réussit) est qu'une CA fournis un certificat pour une clé publique d'une entité finale, et retourne ce certificat à l'entité finale et/ou poste ce certificat dans un répertoire publique. Ce processus peut, et est typiquement, être fait de plusieurs étapes, incluant une initialisation de l'équipement de l'entité finale. Par exemple, l'équipement de l'entité finale doit être initialisé de manière sécurisé avec la clé publique d'une CA, pour être utilisé dans la validation de chemins de certification.

2. **Mise à jour des paires de clé :** Toute paire de clé doit être mis à jours régulièrement (par ex : remplacé par une nouvelle paire de clé), et un nouveau certificat doit être fournis.

3. **Mise à jour de certificat :** Vu que les certificats expirent, ils doivent être rafraîchis si rien n'a été changé dans l'environnement.

4. **Mise à jours de paire de clé de CA :** Comme pour les entités finales, les paires de clé CA doivent être mis à jours régulièrement ; cependant, différents mécanismes sont requis.

5. **Demande de cross-certification :** Une CA demande de fournir un cross-certificat pour une autre CA. Pour ce standard, les termes suivants sont définis. Un cross-certificat est un certificat dans lequel le sujet de la CA et le fournisseur de la CA sont distincts et subjectPublicKeyInfo contient une clé de vérification (par ex : le certificat a été fournis pour la signature de la paire de clé du sujet de la CA.). Quand il est nécessaire de distinguer plus finement, les termes suivants peuvent être utilisé : un cross-certificat est appelé un certification inter-domaine si le sujet et le fournisseur appartiennent à des domaines administratifs différents.

   1. **Note 1.** La définition ci-dessus de cross-certificat s'aligne avec le term définis CA-certificate dans X.509. Noter que ce terme ne doit pas être confondu avec le type d'attribut cACertificate X.500, qui n'est pas lié.

   2. **Note 2.** Dans de nombreux environnements, le terme cross-certificate, sera compris comme synonyme de cross-certificate inter-domaine.

   3. **Note 3.** L'émission de cross certificat peut être, mais pas nécessairement, mutuel ; c'est à dir, 2 CA peuvent fournir des cross certificats entre eux.

6. **Mise à jours de cross-certificats :** Similairement à une mise à jours de certificat.

4. **Opérations de découverte de certificat/CRL :** Certaines opérations de gestion de PKI résultent dans la publication de certificats ou de CRL :

   1. **Publication de certificat :** Après avoir produit un certificat, un moyen de publication est nécessaire. Ce moyen définis dans PKIX peut impliquer les messages spécifiés plus bas, ou d'autres moyen (ldap par exemple).

   2. **Publication de CRL :** Comme pour une publication de certificat.

5. **Opérations de récupération :** Certaines opérations de gestion de PKI sont utilisée quand une entité finale a perdu son PSE :

   1. **Récupération de paire de clé :** Comme option, Les clé clients utilisateurs (par exemple la clé privée de l'utilisateur utilisé pour le déchiffrement) peut être sauvegardé par une CA, une RA, ou un système de sauvegarde de clé associé avec une CA ou une RA. Si une entité a besoin de récupérer sa clé, un protocole d'échange peut être nécessaire pour supporter une telle récupération.

6. **Opérations de révocation :** Certaines opérations de PKI résultent en la création d'une nouvelle entrée de CRL et/ou une nouvelle CRL.

   1. **Demande de révocation :** Une personne autorisée avertis une CA d'une situation anormale nécessitant la révocation de certificat

7. **Opérations PSE :** Bien que la définition des opérations PSE soient en dehors du périmètre de ce document, on doit définir une PKIMessage (CertRepMessage) qui peut former la base de telles opérations.

Noter que les protocoles on-line ne sont par la seule manière d'implémenter les opérations ci-dessus. Pour toutes les opérations, il y à des méthodes offline qui accomplissent le même résultat, et cette spécification ne mandate pas l'utilisation de protocoles on-line. Par exemple, quand des tokens hardware sont utilisé, de nombreuses opération peuvent être accomplies comme partie de la livraison de jetons physique.

Hypothèses et restrictions

Initialisation de l'entité finale

La première étape pour une entité finale en dialoguant avec les entités de gestion de PKI est de demander les informations sur les fonctions de la PKI supportés et pour acquérir de manière sécurisé une copie des clé publiques de CA root.

Enregistrement/certification Initiale

Il y a de nombreux schéma qui peuvent être utilisé pour accomplir un enregistrement et une certification des entités finales. Aucune méthode n'est adaptée pour toutes les situations à cause de la plage de stratégie qu'une CA peut implémenter et la variation dans les types d'entité finales qui peuvent se produire.

Cependant, on peut classer les schémas d'enregistrement/certification qui sont supportés par cette spécification. Noter que le terme d'initial est crucial : on gère les situations où l'entité finale en question n'a jamais eu de contact avec la PKI. Quand l'entité finale possède déjà des clés certifiés, certaines simplification/alternatives sont possibles.

En ayant classé les schémas qui sont supportés par cette spécification on peut ainsi spécifier certains comme obligatoire et d'autres comme optionnels. Le but est que les schéma obligatoires couvrent suffisamment de cas qui se produisent en utilisation réelle, alors que les schémas optionnels se produisent moins fréquemment. De cette manière, on accomplis une balance entre la flexibilité et la simplicité d'implémentation.

Critères utilisés

Initialisation de l'enregistrement/certification

En terme de messages PKI qui sont produits, on peut regarder l'initialisation des échange d'enregistrement/certification initials comme se produisant chaque fois que le premier message de PKI lié à l'entité finale est produite. Noter que l'initialisation dans le monde réel de cette procédure peut se produire n'importe où.

Authentification du message originel de l'entité finale

Les messages on-line produits par l'entité finale qui nécessitent un certificat peuvent être authentifiés ou non. Les prérequis ici sont pour authentifier l'origine de tous messages de l'entité finale vers la PKI (CA/RA).

Dans cette spécification, une telle authentification est accomplie par la PKI (CA/RA) qui fournis à l'entité finale une valeur secrète (clé d'authentification initiale) et une valeur de référence (utilisée pour identifier la valeur secrète) via d'autres moyens. La clé d'authentification initiale peut ainsi être utilisée pour protéger les messages PKI important.

Donc, on peut classer le schéma d'enregistrement/certification initial en fonction de si l'entité finale est on-line ou non -> les messages PKI sont authentifiés on non.

Note 1 : on ne discute pas des messages d'authentification de la PKI -> entité finale ici, vu que c'est toujours requis. Dans tous les cas, il peut être simplement une fois que la clé publique de la CA root a été installée dans l'équipement de l'entité finale ou peut être basée sur la clé d'authentification initiale.

Note 2 : Une procédure d'enregistrement/certification initiale peut être sécurisé via d'autres moyens alternatifs.

Emplacement de la génération de clé

Dans cette spécification, la génération de clé est vu comme se produisant quand la clé publique ou privée se produit dans un PKIMessage. Noter que cela ne pré-inclus pas un service de génération de clé centralisée ; la paire de clé actuelle peut avoir été générée n'importe où et envoyé à l'entité finale, RA, ou CA en utilisant un protocole de requête/réponse de génération de clé. Il y a donc 3 possibilités pour

l'emplacement de la génération de clé : l'entité finale, une RA ou une CA.

Confirmation du succès de la certification

Après la création d'un certificat initial pour une entité finale, Une assurance supplémentaire peut être obtenu en retournant une confirmation du succès de l'opération à l'entité finale, contenant le certificat. Cela donne 2 possibilités : confirmé ou non.

Schéma mandatoire

Le critère ci-dessus permet un grand nombre de schéma d'enregistrement/certification initial. Cette spécification mandate qu'un équipement CA, RA, EE conformes doit supporter le schéma "Schéma authentifié de base", et peut supporter d'autres schémas additionnels.

Schéma centralisé

En terme de classification, ce schéma est, d'une certaine manière, le plus simple possible, où :

- L'initialisation se produit sur la CA certifiante
- Aucune authentification on-line n'est requis
- La génération de clé se produit sur la CA authentifiante
- Aucun message de confirmation n'est requis

En terme de flux de message, ce schéma signifie que le seul message requis est envoyé de la CA à l'entité finale. Le message doit contenir tout le PSE pour l'entité finale. Certains moyens externes doivent être fournis pour permettre à l'entité final d'authentifier le message reçu et pour décrypter et décrypter les valeurs.

Schéma authentifié de base

En terme de classification, ce schéma est où :

- Une initialisation se produit au niveau de l'entité finale
- un message d'authentification est requis
- La génération de clé se produit sur l'entité finale
- Un message de confirmation est requis

En terme de flux de message, ce schéma est comme suit : La génération de la clé, la création de la demande de certification et la protection de la clé d'authentification initiale (IAK) se produisent sur l'entité finale. L'autorité vérifie la requête, la traite et créé la réponse. L'entité final manipule cette réponse et créé la confirmation. L'autorité vérifie la confirmation et créé une réponse.

Preuve de possession de la clé privée

Pour empêcher certaines attaques et pour permettre à une CA/RA de vérifier proprement la validité du lien entre une entité finale et une paire de clé, les opérations de gestion de PKI spécifiés ici permettent à une entité finale de prouver qu'il possède la clé privée correspondant

à la clé publique pour laquelle un certificat est demandée. Une CA/RA est libre de choisir comment forcer le POP dans ses échanges de certification. Cependant, il est nécessaire qu'un CA/RA force le POP par n'importe quel moyen parce qu'il y a de nombreux protocoles opérationnels non-PKIX utilisés qui ne vérifient pas la liaison entre l'entité finale et la clé privée.

POP est accomplis de différente manières en fonction du type de clé pour laquelle un certificat est demandé. Si une clé peut être utilisée différents but, une méthode appropriée peut être utilisée. Par exemple, une clé qui peut être utilisée pour signer, aussi bien que d'autres but, ne devraient pas être envoyés à la CA/RA pour prouver la possession).

Cette spécification permet explicitement les cas où une entité finale fournis la preuve significative à une RA et que cette RA atteste à la CA que la preuve requise a été reçue et validée. Par exemple, une entité finale souhaite avoir une clé de signature certifiée, pourrait envoyer la signature appropriée à la RA, qui peut simplement notifier à la CA que l'entité finale a fournis la preuve requise. Bien sûr, une telle situation peut être interdite par stratégie (ex : les CA peuvent être les seules entités permises à vérifier le POP durant la certification).

Clés de signature

Pour les clés de signature, l'entité finale peut signer une valeur pour prouver la possession de la clé privée.

Clés de chiffrement

Pour les clés de chiffrement, l'entité finale peut fournir la clé privée à la CA/RA, ou peut être requise pour déchiffrer une valeur pour prouver sa possession de la clé privée. En déchiffrant une valeur qui peut être accomplie soit directement, soit indirectement.

La méthode directe sert à une RA/CA pour fournir un challenge aléatoire pour lequel une réponse immédiate par l'EE est requise.

La méthode indirect sert à fournir un certificat qui est chiffré afin que l'EE puisse démontrer sa capacité à déchiffrer le message.

Cette spécification encourage l'utilisation de la méthode indirecte parce qu'elle ne nécessite pas de messages supplémentaires à envoyer.

Clés d'agrément de clé

Pour les clés d'agrément de clé, l'entité finale et l'entité de gestion de PKI doit établir un clé secrète partagée pour pouvoir prouver que l'entité finale possède la clé privée.

Noter que cela nécessite de n'imposer aucune restriction sur les clés qui peuvent être certifiés par une CA donnée. En particulier, pour les clés Diffie-Hellman, l'entité finale peut librement choisir ses paramètres à condition que la CA puisse générer une paire de clé avec les paramètres appropriés si nécessaire.

Mise à jour de clé de la CA root

Cette discussion s'applique uniquement aux CA qui sont directement trustés par certaines entités finales. Les CA auto-signés devraient être considérés comme des CA directement trustés. Reconnaître si une CA non auto-signé est supposé être directement trusté par certaines entités finales est une question de stratégie de CA et est hors du scope de ce document.

La base de la procédure décrite ici est que la CA protège sa nouvelle clé publique en utilisant sa précédente clé privée et vice-versa. Donc, quand une CA met à jour sa paire de clé elle doit générer 2 valeurs d'attribut cACertificate supplémentaires si les certificats sont disponibles en utilisant un annuaire X.500 (pour un total de 4 : OldWithOld, OldWithNew, NewWithOld, et NewWithNew).

Quand une CA change sa paire de clé, ces entités qui ont acquis l'ancienne clé publique de la CA via des moyens tiers sont les plus affectés. Ce sont ces entités finales qui auront besoin d'accéder à la nouvelle clé publique CA protégée avec l'ancienne clé privée CA. Cependant, ils vont seulement avoir besoin de cela pour une période limitée (jusqu'à ce qu'ils aient acquis la nouvelle clé publique). Ce sera facile à accomplir quand les certificats des entités finales expirent.

La structure de données utilisé pour protéger la nouvelle et l'ancienne clé publique CA est un certificat standard (qui peut également contenir des extensions). Il n'y a pas de nouvelle structures de données requises.

Note 1. Ce schéma n'utilise aucune extension X.509 v3 vu qu'il doit être capable fonctionner avec les certificats version 1. La présente de l'extension KeyIdentifier serait cependant plus efficace.

Note 2. Bien que le schéma pourrait être généralisé pour couvrir des cas où les CA mettent à jours leur paire de clé plus d'une fois durant la période de validité du certificat d'une de ses entités finales, cette généralisation semble d'une valeur douteuse. Ne pas avoir cette généralisation signifie simplement que la période de validité des certificats émis avec l'ancienne paire de clé CA ne peut pas excéder la fin de la période de validité OldWithNew.

Note 3. Ce schéma s'assure que les entités finales vont acquérir la nouvelle clé publique CA, à la dernière par expiration du dernier certificat qu'ils possèdent qui a été signé avec l'ancienne clé privée de la CA. Les opération de certificat et/ou de mise à jours de clé se produisant à d'autre moments ne nécessitent pas nécessairement cela.

Actions de l'opérateur CA

Pour changer la clé de la CA, l'opérateur CA effectue :

1. Générer une nouvelle paire de clés
2. Créer un certificat contenant l'ancienne clé publique signée avec la nouvelle clé privée
3. Créer un certificat contenant la nouvelle clé publique signée avec l'ancienne clé privée
4. Créer un certificat contenant la nouvele clé publique CA signée avec la nouvelle clé privée
5. Publier ces nouveaux certificats via le dépôt et/ou d'autres moyens
6. Exporter la nouvelle clé publique CA pour que les entités finales puissent l'acquérir en utilisant un mécanisme externe (si nécessaire).

L'ancienne clé privée CA n'est ainsi plus requise. Cependant, l'ancienne clé publique CA va continuer à être utilisée pour un certain temps. L'ancienne clé publique CA n'est plus requise (autre que pour la non-répudiation) quand toutes les entités finale de cette CA on acquis la nouvelle clé publique CA.

Le certificat "old with new" doit avoir une période de validité commençant à la date de génération de l'ancienne paire de clé et se terminant à la date d'expiration de l'ancienne lé publique.

Le certificat "new with old" doit avoir une période de validité commençant à la date de génération de la nouvelle paire de clé et se terminant à la date à laquelle toutes les entités finales de cette CA vont posséder la nouvelle clé publique.

Le certificat 'new with new' doit avoir une période de validité commençant à la date de génération de la nouvelle paire de clé et se terminant à ou avant la date à laquelle la CA va mettre de nouveau à jour sa paire de clé.

Vérifier les certificats

En vérifiant une signature, le vérificateur vérifie le certificat contenant la clé publique du signataire. Cependant, une fois qu'une CA est autorisée à mettre à jours ses clés il y a de nouvelles possibilités :

- Pour les cas impaires, le signataire des certificats est protégé avec la nouvelle clé publique, dans les cas paires, ce certificat est protégé avec l'ancienne clé publique.
- 2 types de dépôts : le dépôt contient la nouvelle et l'ancienne clé publique ou le dépôt ne contient que l'ancienne clé (dû à un délay)
- 2 cas de possession de la clé : le PSE contient la nouvelle clé, ou le PSE contient l'ancienne clé.

Cas 1 : le dépôt et le PSE contiennent la nouvelle clé : c'est le cas standard où le vérificateur peut directement vérifier le certificat sans utiliser le dépôt.

Cas 2 : le dépôt et le PSE contiennent la nouvelle clé : Le vérificateur doit accéder au dépôt pour obtenir la valeur de l'ancienne clé publique

Cas 3 : le dépôt a la nouvelle clé, mais pas le PSE. Dans ce cas, le vérifier doit accéder au dépôt pour obtenir la valeur de la nouvelle clé publique.

Cas 4 : le dépôt a la nouvelle clé, mais pas le PSE. Le vérificateur peut directement vérifier le certificat sans utiliser le dépôt

Cas 5 : Le PSE a la nouvelle clé, mais pas le dépôt. Bien que l'opération CA n'a pas mis à jours le dépôt, le certificat peut être directement validé (identique au cas 1)

Cas 6 : Le PSE a la nouvelle clé, mais pas le dépôt. Le vérificateur pense qu'il s'agit d'un situation cas 2 et va accéder au dépôt, cependant, la vérification va échouer.

Cas 7 : le dépôt et le PSE contiennent l'ancienne clé : Dans ce cas l'opérateur CA n'a pas mis à jours le dépôt et la vérification va échouer

Cas 8 : le dépôt et le PSE contiennent l'ancienne clé : Bien que l'opérateur CA n'a pas mis à jours le dépôt, le vérificateur peut vérifier le certificat directement, (identique au cas 4)

Vérification dans les cas 1,4,5,8

Dans ces cas, le vérificateur n'a pas de copie locale de la clé publique qui peut être utilisée pour vérifier le certificat directement. C'est la même situation où aucune clé n'a été changée. Noter que le cas 8 peut se produire entre le moment où l'opérateur CA a généré la nouvelle paire de clé et le moment où l'opérateur CA stock les attributs mis à jours dans le dépôt. Le cas 5 peut seulement se produire si l'opérateur CA a émis les certificats du signataire et du vérificateur durant cet écart (L'opérateur CA devrait éviter cela).

Vérification dans le cas 2

Dans le cas 2, le vérificateur doit avoir accès à l'ancienne clé publique de la CA. Le vérificateur fait ceci :

1. Recherche l'attribut caCertificate dans le dépôt et prend le certificat OldWithNew (déterminé sur la période de validité ; noter que les champs subject et issuer doivent correspondre).
2. Vérifie que c'est correcte en utilisant la nouvelle clé CA (que le vérificateur a localement).
3. Si correct, vérifie le certificat du signataire en utilisant l'ancienne clé CA.

Le cas 2 se produit quand l'opérateur CA a émis le certificat du signataire, puis changé la clé, puis émis le certificat du vérificateur ; donc c'est un bon cas typique.

Vérification dans le cas 3

Dans le cas 3, le vérificateur doit avoir accès à la nouvelle clé publique de la CA. Le vérificateur fait ceci :

1. Recherche l'attribut CACertificate dans le dépôt et prend le certificat NewWithOld (déterminé sur la période de validité ; le subject et issuer doivent matcher.
2. Vérifie que c'est correct en utilisant l'ancienne clé

3. Si correct, vérifie le certificat du signataire en utilisant la nouvelle clé CA.

Le cas 3 se produit quand l'opérateur CA a émis le certificat du signataire, puis changé la clé, et ainsi émis le certificat du signataire ; donc c'est également un cas typique.

Érreur de vérification dans le cas 6

Dans ce cas, la CA a émis le PSE du vérificateur, qui contient la nouvelle clé, sans mettre à jours les attributs du dépôt. Cela signifie que le vérificateur n'a aucun moyen d'obtenir une version trustée de l'ancienne clé de la CA et donc la vérification échoue. Cette erreur est une faute de l'opérateur CA.

Érreur de vérification dans le cas 7

Dans ce cas, la CA a émis le certificat du signataire protégé avec la nouvelle clé sans mettre à jours les attributs du dépôt. Cela signifie que le vérificateur n'a aucun moyen d'obtenir une version trustée de l'ancienne clé de la CA et donc la vérification échoue. Cette erreur est une faute de l'opérateur CA.

Révocation - Changement de la clé CA

Comme vu plus haut, la vérification d'un certificat devient de plus en plus complexe une fois que la CA est autorisée à changer sa clé. C'est également vrai pour les vérifications de révocation vu que la CA peut avoir signé la CRL en utilisant une clé privée plus récente que celle dans le PSE de l'utilisateur.

L'analyse des alternatives est la même que pour la vérification du certificat.

Structures de données

Cette section contient des descriptions des structures de données requise pour les messages de gestion de PKI. La section suivante décris les contraintes sur leur valeurs et la séquence des évènements pour chaque opération de gestion de PKI.

Message PKI

Tous les messages utilisés dans cette spécification pour les buts de gestion de PKI utilisent la structure suivante :

```
PKIMessage ::= SEQUENCE {
 header PKIHeader,
 body PKIBody,
 protection [0] PKIProtection OPTIONAL,
 extraCerts [1] SEQUENCE SIZE (1..MAX) OF CMPCertificate OPTIONAL
}
PKIMessages ::= SEQUENCE SIZE (1..MAX) OF PKIMessage
```

PKIHeader contient les informations qui sont communes aux messages PKI

PKIBody contient des informations spécifiques au message

PKIProtection si utilisé, contient les bits qui protègent le message de PKI.

extraCerts peut contenir des certificats qui peuvent être utiles. Par exemple, cela peut être utilisé par une CA ou RA pour présente une entité finale avec les certificats qui sont nécessaires pour vérifier son propre nouveau certificat. (Si, par exemple, la CA qui a émis le certificat EE n'est pas une CA root pour l'EE). Noter que ce champ ne contient pas nécessairement un chemin de certification ; le destinataire peut avoir à trier, sélectionner, ou traiter les certificats supplémentaires pour pouvoir les utiliser.

En-tête de message de PKI

Tous les messages de PKI nécessitent certaines informations d'en-tête pour l'adressage et l'identification de la transaction. Certaines de ces informations vont également être présentes dans une enveloppe spécifique au transport. Cependant, si le message PKI est protégé, alors cette information est également protégée.

La structure de données suivante est utilisée pour contenir cette information :

```
PKIHeader ::= SEQUENCE {
 pvno INTEGER { cmp1999(1), cmp2000(2) },
 sender GeneralName,
 recipient GeneralName,
 messageTime [0] GeneralizedTime OPTIONAL,
 protectionAlg [1] AlgorithmIdentifier OPTIONAL,
 senderKID [2] KeyIdentifier OPTIONAL,
 recipKID [3] KeyIdentifier OPTIONAL,
 transactionID [4] OCTET STRING OPTIONAL,
 senderNonce [5] OCTET STRING OPTIONAL,
 recipNonce [6] OCTET STRING OPTIONAL,
 freeText [7] PKIFreeText OPTIONAL,
 generalInfo [8] SEQUENCE SIZE (1..MAX) OF InfoTypeAndValue OPTIONAL
}
PKIFreeText ::= SEQUENCE SIZE (1..MAX) OF UTF8String
```

pvno est fixé à 2 pour cette version de ce document

sender contient le nom de l'émetteur du PKIMessage. Ce nom (en conjonction avec senderKID, si fournis) devrait être suffisant pour indiquer la clé à utiliser pour vérifier la protection dans le message. Si rien sur l'émetteur n'est connus sur l'entité (ex : dans l'initialisation, un message req, où l'entité finale peut ne pas connaître son propre DN, e-mail, IP, etc), alors ce champ doit contenir une valeur NULL ; c'est à dire une séquence de rdn de longueur 0. Dans un tel cas, le champ senderKID doit contenir un identifiant (par ex : un numéro de référence) qui indique au destinataire les information de clé secrète partagée appropriée pour vérifier le message.

recipient Contient le nom du destinataire du PKIMessage. Ce nom (en conjonction avec recipKID, si fournis) devrait être utilisable pour vérifier la protection du message.

protectionAlg Spécifie l'algorithme utilisé pour protéger le message. Si aucun bits de protection n'est fournis (PKIProtection est optionnel), ce champ doit être omis ; si les bits de protection sont fournis, ce champ doit être fournis.

senderKID, recipKID Sont utilisables pour indiquer quelles clés ont été utilisées pour protéger le message. (RecipKID va normalement être requis seulement quand le message est protégé avec des clés DH). Ces champs doivent être utilisés si nécessaire pour identifier de manière unique une clé et devraient être omis sinon.

transactionID est utilisé pour permettre au destinataire du message de le corréler avec une transaction sortante. C'est nécessaire pour toutes les transactions qui consistent d'une simple paire demande/réponse. Pour ces transactions demande/réponse, les règles sont comme suit. Un client peut populer le champ transactionID de la requête. Si un serveur reçois une telle requête qui a le champ transactionID mis, alors il doit mettre le champ transactionID de la réponse à la même valeur. Si un serveur reçois une telle requête avec un champ transactionID manquant, alors il peut mettre le champ transactionID de la réponse.

Pour les transaction qui consistent de plus qu'une simple paire demande/réponse, les règles sont comme suit. Les client devraient générer une transactionID pour la première demande. Si un serveur reçoit une telle demande avec le champs transactionID mis, alors il doit mettre le champ transactionID de la réponse à la même valeur. Si un serveur reçois une telle demande sans le champs transactionID, il doit populer ce champs dans la réponse avec un ID généré par le serveur. Les requêtes et réponses suivantes doivent toutes mettre le champs

transactionID avec la valeur établie. Dans tous les cas où un transactionID est utilisé, un client ne doit pas avoir plus d'une transaction avec le même transactionID en cours vers un serveur donné. Les serveur sont libre d'imposer ou non l'unicité du transactionID, tant qu'ils sont capable d'associer correctement les messages avec la transaction correspondante.

Typiquement, cela signifie qu'un serveur va nécessiter que le couple client/transactionID soit unique, ou même le transactionID seul, s'il ne peut pas distinguer les clients basé sur les information au niveau du transport. Un serveur recevant le premier message d'un transaction (qui nécessite plus d'une paire demande/réponse) qui contient un transactionID qui ne permet pas de répondre aux contraintes ci-dessus (généralement parce que le transactionID est déjà utilisé) doit envoyer un ErrorMsgContent avec un PKIFailureInfo de transactionIdInUse. Il est recommandé que les client utilisent un transactionID avec une donnée pseudo-aléatoire 128-bits pour commencer la transaction pour réduire la probabilité d'avoir un transactionID utilisé par le serveur.

senderNonce, recipNonce protègent le PKIMessage avec les attaques replay. senderNonce est généralement une donnée pseudo-aléatoire 128-bits générée par l'émetteur, et recipNonce est copié de senderNonce du précédent message dans la transaction.

messageTime Contient l'horodatage du moment où l'émetteur a créé le message. Peut être utile pour permettre aux entités finales de corriger/vérifier l'heure local.

freeText Peut être utilisé pour envoyer des données additionnelles au destinataire. Le premier langage utilisé dans cette séquence indique la langue désirée pour les réponses.

generalInfo Peut être utilisé pour envoyer des données additionnelles traitable par la machine au destinataire. Les extentions generalInfo suivantes sont définies et peuvent être supportées :

ImplicitConfirm

C'est utilisé par l'EE pour informer la CA qu'elle ne souhaite pas envoyer une confirmation de certificat pour les certificats émis.

```
implicitConfirm OBJECT IDENTIFIER ::= {id-it 13}
ImplicitConfirmValue ::= NULL
```

Si la CA autorise la demande, elle doit placer la même extension dans le PKIHeader de la réponse. Si l'EE ne trouve pas l'extension dans la réponse, elle doit envoyer la confirmation du certificat.

ConfirmWaitTime

C'est utilisé par la CA pour informer l'EE du temps d'attente prévu pour la confirmation de certificat avant de révoque le certificat et supprimer la transaction.

```
confirmWaitTime OBJECT IDENTIFIER ::= {id-it 14}
ConfirmWaitTimeValue ::= GeneralizedTime
```

Corps du message PKI

```
PKIBody ::= CHOICE {
 ir [0] CertReqMessages, -Initialization Req
 ip [1] CertRepMessage, -Initialization Resp
 cr [2] CertReqMessages, -Certification Req
 cp [3] CertRepMessage, -Certification Resp
 p10cr [4] CertificationRequest, -PKCS #10 Cert. Req.
 popdecc [5] POPODecKeyChallContent -pop Challenge
 popdecr [6] POPODecKeyRespContent, -pop Response
 kur [7] CertReqMessages, -Key Update Request
 kup [8] CertRepMessage, -Key Update Response
```

```
 krr [9] CertReqMessages, -Key Recovery Req
 krp [10] KeyRecRepContent, -Key Recovery Resp
 rr [11] RevReqContent, -Revocation Request
 rp [12] RevRepContent, -Revocation Response
 ccr [13] CertReqMessages, -Cross-Cert. Request
 ccp [14] CertRepMessage, -Cross-Cert. Resp
 ckuann [15] CAKeyUpdAnnContent, -CA Key Update Ann.
 cann [16] CertAnnContent, -Certificate Ann.
 rann [17] RevAnnContent, -Revocation Ann.
 crlann [18] CRLAnnContent, -CRL Announcement
 pkiconf [19] PKIConfirmContent, -Confirmation
 nested [20] NestedMessageContent, -Nested Message
 genm [21] GenMsgContent, -General Message
 genp [22] GenRepContent, -General Response
 error [23] ErrorMsgContent, -Error Message
 certConf [24] CertConfirmContent, -Certificate confirm
 pollReq [25] PollReqContent, -Polling request
 pollRep [26] PollRepContent -Polling response
}
```

Les types spécifiques sont décris plus bas.

Protection de message PKI

Certains message PKI vont être protéger pour l'intégrité. (Noter que si un algorithme asymétrique est utilisé pour protéger un message et que le composant publique a déjà été certifié, l'origine du message peut également être authentifié).

Quand la protection est appliquée, la structure suivante est utilisée :

```
PKIProtection ::= BIT STRING
```

L'entrée pour le calcul de PKIProtection est un encodé DER de la structure de données suivante :

```
ProtectedPart ::= SEQUENCE {
 header PKIHeader,
 body PKIBody
}
```

Il peut y avoir des cas dans lesquels la chaîne de bit PKIProtection n'est délibérément pas utilisée pour protéger un message parce que d'autre protection externes sont appliqués à la place. Un tel choix est explicitement permit dans cette spécification. Des exemples de telles protections externes incluent les encapsulations PKCS#7 et Security Multiparts (rfc1847) du PKIMessage (ou simplement le PKIBody), si les informations PKIHeader sont gérés de manière sécurisés dans le mécanisme externe. Il est noté, cependant, que de nombreux mécanismes externes nécessitent que l'entité finale possède déjà un certificat et/ou un DN unique, et/ou d'autres informations liées à l'infrastructure.

Donc, ils peuvent être non appropriés pour un enregistrement initial, récupération de clé, ou tout autre processus avec des caractéristiques de 'boot-strapping'. Pour ces cas il peut être nécessaire que le paramètre PKIProtection soit utilisé. Dans le future, si et quand les mécanismes externes sont modifiés pour ces scénarios, l'utilisation de PKIProtection peut devenir rare ou non-existant.

En fonction des circonstance, les bits PKIProtection peuvent contenir un MAC ou une signature. Seul les cas suivants peut se produire.

Information de secret partagé

Dans ce cas, l'émetteur et le destinataire partages une information secrète (établie via un moyen externe ou depuis une opération de gestion PKI précédente). PKIProtection va maintenir une valeur MAC et protectionAlg sera :

```
id-PasswordBasedMac OBJECT IDENTIFIER ::= {1 2 840 113533 7 66 13}
PBMParameter ::= SEQUENCE {
 salt OCTET STRING,
 owf AlgorithmIdentifier,
 iterationCount INTEGER,
 mac AlgorithmIdentifier
}
```

Dans protectionAlg ci-dessus, la valeur salt est ajoutée à l'entrée du secret partagé. Le OWF est ainsi appliqué iterationCount fois, où le secret salé est l'entrée de la première itération et, pour chaque itération successive, l'entrée est la sortie de l'itération précédente. La sortie de l'itération finale (appelée BASEKEY avec une taille H) est ce qui est utilisé pour former la clé symétrique. Si l'algorithme MAC nécessite une clé K-Bits et K <= H, alors les bits K les plus signifiant de BASEKEY sont utilisés. Si K > H, alors tout BASEKEY est utilisé pour les bits H les plus signifiant de la clé, OWF('1' || BASEKEY) est utilisé pour les bits H les plus signifiant suivants de la clé, OWF('2' || BASEKEY) est utilisé pour les bits H les plus signifiant suivants de la clé et ainsi de suite, jusqu'à ce que tous les bits K aient été dérivés.

Note : il est recommandé que les champs de PBMParameter restent constants via les messages d'une simple transaction (ex : ir/ip/certConf/pkiConf) pour pouvoir réduire la charge associé avec le calcul de PasswordBasedMac.

Paires de clé DH

Où l'émetteur et le destinataire possèdent des certificats Diffie-Hellman avec des paramètres DH compatibles, pour pouvoir protéger le message l'entité finale doit générer une clé symétrique basée sur la clé privée DH et la clé publique DH du destinataire de message PKI. PKIProtection contient une valeur MAC protégé avec cette clé symétrique dérivée et protectionAlg sera :

```
id-DHBasedMac OBJECT IDENTIFIER ::= {1 2 840 113533 7 66 30}
DHBMParameter ::= SEQUENCE {
 owf AlgorithmIdentifier, - AlgId for a One-Way Function (SHA-1 recommended)
 mac AlgorithmIdentifier - the MAC AlgId (e.g., DES-MAC, Triple-DES-MAC [PKCS11],
}
```

Dans le protectionAlg ci-dessus, OWF est appliqué au résultat du calcul DH. La sortie OWF (appelée BASEKEY, avec une taille H) est ce qui est utilisé pour former la clé symétrique. Si l'algorithme MAC nécessite une clé K-bit et K <= H, les bits K les plus signifiant de BASEKEY sont utilisé. Si K > H, tout BASEKEY est utilisé pour les bits H les plus signifiants suivants, OWF('1' || BASEKEY) est utilisé pour les bits H les plus signifiant suivant, OWF('2' || BASEKEY) est utilisé pour les bits H les plus signifiant suivant, et ainsi de suite, jusqu'à ce que les bits K aient été dérivés.

Signature

Dans ce cas, l'émetteur possède une paire de clé de signature et signe simplement le message PKI. PKIProtection va contenir la valeur de signature et protectionAlg sera un AlgorithmIdentifier pour une signature numérique.

Protection Multiple

Dans les cas ou une entité finale envoie un message PKI protégé à une RA, la RA peut transférer ce message à une CA, attachant sa propre protection (qui peut être un MAC ou une signature, en fonction des informations et des certificats partagés entre la RA et la CA). C'est accomplis en imbriquant tout le message envoyé par l'entité finale dans un nouveau message PKI. La structure utilisée est :

```
NestedMessageContent ::= PKIMessages
```

L'utilisation des PKIMessages, une sequence de PKIMessage, laisse la RA automatiser les requêtes de nombreux EE dans un simple nouveau message. Par simplicité, tous les messages dans le batch doivent être de même type (ex : ir). Si la RA souhaite modifier les messages d'une certaine manière (ex : ajouter des valeur de champ particulier ou de nouvelles extensions), elle peut créer sont propre PKIBody. Le PKIMessage originel de l'EE peut être inclus dans le champ generalInfo de PKIHeader. l'infoType utilisé dans cette situation est {id-it 15} et infoValue est PKIMessages (le contenu doit être dans le même ordre que les demandes dans PKIBody).

Structures de données communes

Avant de spécifier les types spécifique qui peuvent être placés dans un PKIBody, on définis certaines structures qui sont utilisées dans plus d'un cas.

Contenu des certificats demandés

Divers messages de gestion de PKI nécessitent que l'initiateur du message indique certains des champs qui doivent être présent dans un certificat. La structure CertTemplate permet à une entité finale ou une RA de spécifier ce qu'elle souhaite dans le certificat. CertTemplate est identique à une certificat, mais avec tous les champs optionnels.

Noter que même si l'initiateur spécifie complètement le contenus d'un certificat, une CA est libre de modifier les champs dans le certificat qu'elle émet. Si le certificat modifié n'est pas acceptable pour le demandeur, le demandeur doit renvoyer un message certConf qui soit n'inclus pas ce certificat, ou inclus ce certificat (via un CertHash) avec un statut "rejected".

Valeurs chiffrées

Quand des valeurs chiffrées (restreintes, dans cette spécification, à des clés privées ou des certificats) sont envoyés dans les messages PKI, la structure de données EncryptedValue est utilisée.

L'utilisation de cette structure nécessite que le créateur et de destinataire soient capable de chiffrer et déchiffrer, respectivement. Typiquement, cela signifie que l'émetteur et de destinataire aient, ou soient capable de générer une clé secrète partagée.

Si le destinataire du PKIMessage possède déjà une clé privée utilisable pour le déchiffrement, alors le champ encSymmKey peut contenir une clé de session chiffrée en utilisant la clé publique du destinataire.

Codes de status et information d'erreur pour les messages PKI

Tous les messages de réponse incluent une information de status. Les valeurs suivantes sont définies.

```
PKIStatus ::= INTEGER {
 accepted (0),
 grantedWithMods (1),
 rejection (2),
 waiting (3),
 revocationWarning (4),
 revocationNotification (5),
 keyUpdateWarning (6)
}
```

Les répondeurs peuvent utiliser la syntaxe suivante pour fournir certaines informations sur les cas d'erreur

```
PKIFailureInfo ::= BIT STRING {
 badAlg (0),
 badMessageCheck (1),
 badRequest (2),
 badTime (3),
 badCertId (4),
 badDataFormat (5),
 wrongAuthority (6),
 incorrectData (7),
 missingTimeStamp (8),
 badPOP (9),
 certRevoked (10),
 certConfirmed (11),
 wrongIntegrity (12),
 badRecipientNonce (13),
 timeNotAvailable (14),
 unacceptedPolicy (15),
 unacceptedExtension (16),
 addInfoNotAvailable (17),
 badSenderNonce (18),
 badCertTemplate (19),
 signerNotTrusted (20),
 transactionIdInUse (21),
 unsupportedVersion (22),
 notAuthorized (23),
 systemUnavail (24),
 systemFailure (25),
 duplicateCertReq (26)
}

PKIStatusInfo ::= SEQUENCE {
 status PKIStatus,
 statusString PKIFreeText OPTIONAL,
 failInfo PKIFailureInfo OPTIONAL
}
```

Identification de certificat

Pour pouvoir identifier des certificats particuliers, la structure de données CertId est utilisée.

Clé publique CA out-of-band

Chaque CA root doit être capable de publier sa clé publique courante via des moyens externes. Bien que de tels mécanismes soient au-delà du scope de ce document, on définis des structures de données qui peuvent supporter de tels mécanismes.

Il y a généralement 2 méthodes disponibles : soit la CA publie directement son certificat auto-signé, ou cette information est disponible via l'annuaire (ou équivalent) et la CA publie un hash de cette valeur pour permettre la vérification de son intégrité avec utilisation.

```
OOBCert ::= Certificate
```

Les champs dans ce certificat sont restreints comme suit :

- Le certificat doit être auto-signé (par ex : la signature doit être vérifiable en utilisant le champ subjectPublicKeyInfo.
- Les champs subject et issuer doivent être identiques
- Si le champ subject est NULL, alors les extensions subjectAltNames et issuerAltNames doivent être présents et avoir la même valeur
- Les valeur de toutes les autres extensions doivent être correct pour un certificat auto-signé.

```
OOBCertHash ::= SEQUENCE {
 hashAlg [0] AlgorithmIdentifier OPTIONAL,
 certId [1] CertId OPTIONAL,
 hashVal BIT STRING
}
```

L'intension de la valeur de hash est que celui qui a reçu de manière sécurisée cette valeur peut vérifier un certificat auto-signé pour cette CA.

Options d'archive

Les demandeurs peuvent indiquer qu'ils souhaitent que la PKI archive une valeur de clé privée en utilisant la structure PKIArchiveOptions.

Information de publication

Les demandeurs peuvent indiquer qu'ils souhaitent que la PKI publient un certificat en utilisant la structure PKIPublicationInfo

Structures POP

Si la demande de certification concerne la signature d'une paire de clé, alors la preuve de possession de la clé privée est démontrée via l'utilisation de la structure POPOSigningKey.

```
POPOSigningKeyInput ::= SEQUENCE {
 authInfo CHOICE {
  sender [0] GeneralName,
  publicKeyMAC PKMACValue
 },
 publicKey SubjectPublicKeyInfo
}
```

D'une autre manière, si la demande de certification est pour une paire de clé de chiffrement, alors la preuve de possession de la clé de privée peut être démontrée d'une des 3 manières suivante :

Méthode indirecte

La CA ne retourne pas un certificat, mais un certificat chiffré (le certificat chiffré avec une clé symétrique générée aléatoirement, et la clé symétrique est chiffrée avec la clé publique pour laquelle la demande de certificat est faite). L'entité finale prouve sa connaissance de la clé privée à la CA en fournissant le CertHash correct pour ce certificat dans le message certConf. Cela démontre le POP parce que l'EE peut seulement calculer le CertHash correct s'il est capable de récupérer le certificat, et il peut seulement récupérer le certificat s'il est capable

déchiffrer la clé symétrique en utilisant sa propre clé privée. Pour que cela fonctionne, la CA ne doit pas publier le certificat tant que le message certConf n'est pas arrivé (où certHash est utilisé pour démontrer POP).

Protocole Challenge-Response

L'entité finale s'engage dans un protocole challenge-response (en utilisant les messages POPODecKeyChall et POPODecKeyResp) entre CertReqMessages et CertRepMessage – c'est la méthode directe décrite plus haut. (Cette méthode est généralement utilisée dans un environnement dans lequel une RA vérifie le POP et créé la demande de certification à la CA. Dans un tel scénario, la CA trust la RA pour avoir validé le POP correctement avant que la RA demande un certificat pour l'entité finale). Le protocole complet ressemble comme suit :

```
EE_____________RA_____________CA
_________---_req_--->
_________<--_chall_--
_________---_resp_-->
_____________________---_req'_-->
_____________________<--_rep_---
_____________________---_conf_-->
_____________________<--_ack_---
_________<--_rep_---
_________---_conf_-->
_________<--_ack_---
```

Ce protocole est plus long qu'un échange triple donné dans le choix 2 ci-dessus, mais permet à une RA d'être impliqué et a la propriété que le certificat n'est pas créé tant que le POS n'est pas complété. Dans certains environnements, un ordre différent peut être requis, tel que ceci (peut être déterminé par stratégie) :

```
EE_____________RA_____________CA
---_req_--->
<--_chall_--
---_resp_-->
_________________---_req'_-->
_________________<--_rep_---
<--_rep_---
---_conf_-->
_________________---_conf_-->
_________________<--_ack_---
<--_ack_---
```

Si la demande de certificat est pour une paire de clé d'agrément de clé (KAK), alors le POP peut être utilisé dans un échange triple décris plus haut pour encoder les paires de clé, avec les changement suivant : (1) le texte entre parenthèses 2) est remplacé avec "ex : le certificat chiffré avec la clé symétrique dérivé de la clé privée KAK de la CA et la clé publique pour laquelle la demande de certification est faite". (2) le premier texte entre parenthèses du champ challenge est remplacé avec "(PreferredSymmAlg et une clé symétrique dérivé de la clé privée KAK de la CA et la clé publique pour laquelle la demande de certificat est faite)". Alternativement, le POP peut utiliser la structure POPOSigningKey comme 4ème alternative pour démontrer le POP si la CA a déjà un certificat DH qui est connu de l'EE.

Les messages challenge-response pour POP d'une clé privée de chiffrement sont spécifié comme suit. Noter que cet échange est associé avec le message de demande de certification précédent by le transactionID utilisé dans le PKIHeader et par la protection appliquée au PKIMessage.

```
POPODecKeyChallContent ::= SEQUENCE OF Challenge
Challenge ::= SEQUENCE {
 owf AlgorithmIdentifier OPTIONAL,
 witness OCTET STRING,
 challenge OCTET STRING
}
```

Noter que la taille de Rand doit être approprié pour le chiffrement sous la clé publique de demandeur. Cet entier n'est généralement pas plus long que 64bits, laissant 100 octets pour le champ sender quand le modulo est 1024 bits. Si, dans certains environnements, les noms sont trop long, alors tout ce qui peut rentrer doit être utilisé (tant que cela inclus au moins un nom commun, et tant que le destinataire est capable de comprendre l'abréviation).

```
POPODecKeyRespContent ::= SEQUENCE OF INTEGER
```

Sommaire des options POP

Le texte dans cette section fournis de nombreuses options en respect des techniques POP. En utilisant SK pour Signing Key, EK pour Encryption Key, et KAK pour Key Agreement Key, les techniques peut être listés comme suit :

RAVerified
SKPOP
EKPOPThisMessage
KAKPOPThisMessage
KAKPOPThisMessageDHMAC
EKPOPEncryptedCert
KAKPOPEncryptedCert
EKPOPChallengeResp
KAKPOPChallengeResp

En donnant ce tableau d'options, il est naturel de demander comment une entité finale peut connaître ce qui est supporté par la CA/RA. Les guides suivants devraient clarifier cette situation.

RAVerified Ce n'est pas une décision de l'EE ; le RA l'utilise si et seulement si a vérifié le POP avant de transmettre la requête à la CA, donc il n'est pas possible pour l'EE de choisir cette technique.

SKPOP Si l'EE an une paire de clé de signature, c'est la seule méthode POP spécifiée pour l'utilisation dans la requête pour un certificat correspondant.

EKPOPThisMessage, KAKPOPThisMessage Donner on non sa clé privée à la CA/RA est une décision EE. Si l'EE décide de révéler sa clé, alors ce sont les seules méthodes POP disponible dans cette spécification pour l'accomplir.

KAKPOPThisMessageDHMAC L'EE peut seulement utiliser cette méthode si (1) la CA a un certificat DH disponible dans ce but, et (2) l'EE a déjà une copie de ce certificat. Si ces 2 conditions sont remplies, cette technique est clairement supportée et peut être utilisé par l'EE, si désiré.

EKPOPEncryptedCert, KAKPOPEncryptedCert, EKPOPChallengeResp, KAKPOPChallengeResp L'EE choisis une des technique dans le message de demande, en fonction des préférences et du type de clés. L'EE ne fait pas de POP à ce moment là ; il indique simplement quelle méthode il souhaite utilisé. Cependant, si la CA/RA répond avec une erreur badPOP, l'EE peut redemander en utilisant une autre méthode. Noter cependant que cette spécification encourage l'utilisation de EncryptedCert et, en outre, dit que le challenge-response devrait être utilisé quand une RA est impliquée et fait la vérification POP. Ainsi, l'EE devrait être capable de prendre une décision intelligente concernant la méthode POP à choisir dans la demande.

Initialisation de la demande

Un message d'initialisation de demande contient comme PKIBody la structure CertReqMessages, qui spécifie les certificats demandés. Typiquement, SubjectPublicKeyInfo, KeyId, et Validity sont les champs qui doivent être fournis pour chaque demande de certificat. Ce message est prévu pour être utilisé par les entité qui s'initialise pour la première fois dans la PKI.

Initialisation de la réponse

Un message d'initialisation de réponse contient comme PKIBody la structure CertRepMessage, qui a pour chaque attribut demandé un champ PKIStatusInfo, un sujet, et possiblement une clé privée (normalement chiffrée avec une clé de session, qui est elle-même chiffrée avec le protocolEncrKey). Noter que si la protection du message est de type secret partagé, tout certificat transporté dans le champ caPubs peut être directement validé comme certificat CA root par l'initiateur.

Demande de certification

Un message de demande de certification contient comme PKIBody une structure CertReqMessages, qui spécifie les certificats demandés. Ce message est prévu pour être utilisé pour les entité PKI existantes qui souhaitent obtenir des certificats additionnels. Alternativement, PKIBody peut être un CertificationRequest. Cette structure peut être requise pour les demande de certificat pour les paires de clé de signature quand l'intéropérabilité avec les anciens systèmes est désirée, mais son utilisation est fortement découragée.

Réponse de certification

Un message de réponse de certification contient comme PKIBody une structure CertRepMessage, qui a la valeur de status pour chaque certificat demandé, et optionnellement une clé publique de CA, des informations d'erreur, un sujet, et une clé privée chiffrée.

```
CertRepMessage ::= SEQUENCE {
 caPubs [1] SEQUENCE SIZE (1..MAX) OF Certificate OPTIONAL,
 response SEQUENCE OF CertResponse
}

CertResponse ::= SEQUENCE {
 certReqId INTEGER,
 status PKIStatusInfo,
 certifiedKeyPair CertifiedKeyPair OPTIONAL,
 rspInfo OCTET STRING OPTIONAL
   analogue à id-regInfo-utf8Pairs définis pour regInfo dans CertReqMsg
}

CertifiedKeyPair ::= SEQUENCE {
 certOrEncCert CertOrEncCert,
 privateKey [0] EncryptedValue OPTIONAL,
 publicationInfo [1] PKIPublicationInfo OPTIONAL
}

CertOrEncCert ::= CHOICE {
 certificate [0] Certificate,
 encryptedCert [1] EncryptedValue
}
```

Seul un champ failInfo (dans PKIStatusInfo) et certificate (dans CertifiedKeyPair) peuvent être présents dans chaque CertResponse (en fonction du statut). Pour certaines valeurs de statut (ex : waiting), aucun des champs optionnels ne sont présents.

En donnant un EncryptedCert et la clé de déchiffrement, le certificat peut êtr obtenu. Le but de cela est de permettre à une CA de retourner la valeur d'un certificat, mais avec la contrainte que seule le destinataire prévu peut obtenir le certificat. Le bénéfice de cette approche est qu'un CA peut répondre avec un certificat même en l'absence de preuve que le demandeur est l'entité finale qui peut utiliser la clé privée (noter que la preuve n'est pas obtenue jusqu'à ce que le message certConf soit reçu par la CA). Donc, la CA n'aura pas à révoquer ce certificat dans le cas où quelque-chose ne va pas avec le POP.

Contenu des demandes de mise à jour de clé

Pour les demandes de mise à jour de clé la syntaxe CertReqMessages est utilisée. Typiquement, SubjectPublicKeyInfo, KeyId, et Validity sont les champs fournis pour chaque clé à mettre à jours. Ce message est prévu pour être utilisé pour demanders des mises à jours de certificat existant non-révoqués et non-expirés. Une mise à jours est un remplacement de certificat contenant soit une nouvelle clé publique ou la clé publique courante.

Contenu des réponses de mise à jour de clé

Pour les réponses de mise à jour de clé, la syntaxe CertRepMessage est utilisée. La réponse est identique à la réponse d'initialisation.

Contenu de demande de récupération de clé

Pour les demandes de récupération de clé la syntaxe utilisée est identique à CertReqMessages des demandes d'initialisation. Typiquement, SubjectPublicKeyInfo et KeyId sont les champs qui peuvent être utilisé pour fournir une clé publique de signature pour laquelle un certificat est requis. Noter que si un historique de clé est requis, le demandeur doit fournir un contrôle de clé de chiffrement de protocole dans la demande.

Contenu de réponse de récupération de clé

Pour les réponses de récupération de clé, la syntaxe suivante est utilisée. Pour certaines valeurs de statut (ex : waiting), aucun champ optionnel n'est présent.

```
KeyRecRepContent ::= SEQUENCE {
 status PKIStatusInfo,
 newSigCert [0] Certificate OPTIONAL,
 caCerts [1] SEQUENCE SIZE (1..MAX) OF Certificate OPTIONAL,
keyPairHist [2] SEQUENCE SIZE (1..MAX) OF CertifiedKeyPair OPTIONAL
}
```

Contenu de demande de révocation

Pour une demande de révocation d'un ou plusieurs certificats, la structure de données suivante est utilisée. Le nom du demandeur est présent dans la structure PKIHeader

```
RevReqContent ::= SEQUENCE OF RevDetails
RevDetails ::= SEQUENCE {
 certDetails CertTemplate,
 crlEntryDetails Extensions OPTIONAL
}
```

Contenu de réponse de révocation

La réponse de révocation est envoyée au demandeur de la révocation :

```
RevRepContent ::= SEQUENCE {
 status SEQUENCE SIZE (1..MAX) OF PKIStatusInfo,
 revCerts [0] SEQUENCE SIZE (1..MAX) OF CertId OPTIONAL,
 crls [1] SEQUENCE SIZE (1..MAX) OF CertificateList OPTIONAL
}
```

Contenu de demande de cross-certification

Les demandes de cross-certification utilisent la même syntaxe (CertReqMessages) que pour les demandes de certification normales, avec comme restriction que la paire de clé doit avoir été générée par la CA demandeur et que la clé privée ne doit pas être envoyée. Cet requête peut également être utilisé par les CA subordonnées pour obtenir leur certificats signés par la CA parent.

Contenu de réponse de cross-certification

Les réponse de cross-certification utilisent la même syntaxe (CertRepMessage) que pour les réponse de certification normales, avec comme restriction qu'aucune clé privée de chiffrement ne peut être envoyée.

Contenu d'annonce de mise à jour de clé CA

Quand une CA met à jours sa propre paire de clé, la structure de données suivante peut être utilisé pour annoncer cet évènement :

```
CAKeyUpdAnnContent ::= SEQUENCE {
 oldWithNew Certificate,
 newWithOld Certificate,
 newWithNew Certificate
}
```

Annonce de certificat

Cette structure peut être utilisé pour annoncer l'existence de certificat. Noter que ce message est prévu pour être utilisé pour les cas (s'il y'en a) où il n'y a pas de méthode pré-existante pour la publication de certificats ; il n'est pas prévu pour être utilisé où, par exemple, X.500 est la méthode pour la publication de certificats

```
CertAnnContent ::= Certificate
```

Annonce de révocation

Quand une CA a été révoquée, ou est en train de révoquer un certificat particulier, elle peut émettre une annonce de cet évènement :

```
RevAnnContent ::= SEQUENCE {
 status PKIStatus,
 certId CertId,
 willBeRevokedAt GeneralizedTime,
 badSinceDate GeneralizedTime,
 crlDetails Extensions OPTIONAL
```

```
}
```

Une CA peut utiliser une telle annonce pour alerter ou notifier un sujet que son certificat est révoqué. C'est généralement utilisé quand la demande de révocation en vient pas du sujet concerné. Le champ willBeRevokedAt contient la date à laquelle une nouvelle entrée sera ajoutée aux CRLs concernées.

Contenu de confirmation PKI

Cette structure est utilisée dans le protocole d'échange comme PKIMessage final. Son contenu est le même dans tous les cas – actullement il n'y a aucun contenu vu que PKIHeader gère toutes les informations requises.

```
PKIConfirmContent ::= NULL
```

L'utilisation de ce message pour la confirmation de certificat n'est pas recommandée. certConf devrait être utilisé à la place. Une fois reçu un PKIConfirm pour une réponse de certificat, le destinataire peut le traiter comme un certConf avec tous les certificats acceptés.

Contenu de confirmation de certificat

Cette structure est utilisée par le client pour envoyer une confirmation à la CA/RA pour accepter ou rejeter le certificat.

```
CertConfirmContent ::= SEQUENCE OF CertStatus
CertStatus ::= SEQUENCE {
 certHash OCTET STRING,
 certReqId INTEGER,
 statusInfo PKIStatusInfo OPTIONAL
}
```

Pour un CertStatus particulier, l'omission du champ statusInfo indique l'acceptation du certificat spécifié. Alternativement, des détails de statut explicite peuvent être fournis dans le champ statusInfo.

Dans CertConfirmContent, l'omission d'une structure CertStatus correspond à un certificat fournis dans la réponse précédente indiquant le rejet du certificat. Donc, un CertConfirmContent vide peut être utilisé pour indiquer le rejet de tous les certificats émis.

Contenu du message général PKI

```
InfoTypeAndValue ::= SEQUENCE {
 infoType OBJECT IDENTIFIER,
 infoValue ANY DEFINED BY infoType OPTIONAL
} - où {id-it} = {id-pkix 4} = {1 3 6 1 5 5 7 4}
GenMsgContent ::= SEQUENCE OF InfoTypeAndValue
```

Certificat de chiffrement de protocole CA

Peut être utilisé par l'EE pour obtenir un certificat depuis la CA à utiliser pour protéger les informations sensibles durant le protocole :

```
GenMsg: {id-it 1}, < absent >
```

GenRep: {id-it 1}, Certificate | < absent >

Les EE doivent s'assurer que le certificat correct est utilisé pour ce but.

Types de paire de clé de signature

Peut être utilisé par l'EE pour obtenir la liste des algorithmes de signature dont les valeur de clé publique peuvent être certifiés par la CA. Noter que dans le but de cet échange, rsaEncryption et rsaWithSHA1, par exemple, sont considérés équivalents.
GenMsg: {id-it 2}, < absent >
GenRep: {id-it 2}, SEQUENCE SIZE (1..MAX) OF AlgorithmIdentifier

Types de paire de clé d'agrément de clé/chiffrement

Peut être utilisé par le client pour obtenir la liste des algorithmes d'agrément de clé/chiffrement dont les valeurs de clé publique peut être certifiés par la CA.
GenMsg: {id-it 3}, < absent >
GenRep: {id-it 3}, SEQUENCE SIZE (1..MAX) OF AlgorithmIdentifier

Algorithme symétrique préféré

Peut être utilisé par le client pour obtenir l'algorithme de chiffrement préféré de la CA pour les informations confidentielles qui nécessitent d'échanger entre l'EE et la CA.
GenMsg: {id-it 4}, < absent >
GenRep: {id-it 4}, AlgorithmIdentifier

Paire de clé CA mise à jour

Peut être utilisé par la CA pour annoncer une mise à jour de clé de CA
GenMsg: {id-it 5}, CAKeyUpdAnnContent

CRL

Peut être utilisé par le client pour obtenir une copie de la dernière CRL.
GenMsg: {id-it 6}, < absent >
GenRep: {id-it 6}, CertificateList

Identifiant d'objet non supporté

C'est utilisé par le serveur pour retourner une liste d'identifiants d'objets qu'il ne reconnaît ou supporte.

```
GenRep: {id-it 7}, SEQUENCE SIZE (1..MAX) OF OBJECT IDENTIFIER
```

Paramètres de paire de clé

Peut être utilisé par l'EE pour demander les paramètres de domaine à utiliser pour générer la paire de clé pour certains algorithmes à clé publique. Peut être utilisé, par exemple, pour demander P, Q et G appropriés pour générer la clé DH/DSA, ou pour demander un jeu de courbe elliptique.

```
GenMsg: {id-it 10}, OBJECT IDENTIFIER - (Algorithm object-id)
GenRep: {id-it 11}, AlgorithmIdentifier | < absent >
```

Un infoValue absent dans GenRep indique que l'algorithme spécifié dans GenMsg n'est pas supporté. Les EE doivent s'assurer que les paramètres sont acceptable et que le message GenRep est authentifié.

passphrase de révocation

Peut être utilisé par l'EE pour envoyer un passphrase à une CA/RA pour authentifier un demande de révocation.

```
GenMsg: {id-it 12}, EncryptedValue
GenRep: {id-it 12}, < absent >
```

Tags de langue supportés

Peut être utilisé pour déterminer le tag de langue approprié à utiliser dans les messages suivants. L'émeteur envoie sa liste de langages supportés (par ordre de préférence) ; le destinataire retourne celui qu'il souhaite utiliser. Si aucun des tags proposés ne sont supportés, une erreur doit être retournée.

```
GenMsg: {id-it 16}, SEQUENCE SIZE (1..MAX) OF UTF8String
GenRep: {id-it 16}, SEQUENCE SIZE (1) OF UTF8String
```

Contenu de réponse PKI Général

```
GenRepContent ::= SEQUENCE OF InfoTypeAndValue
```

Cette structure de données peut être utilisée par l'EE, CA ou RA pour transporter des informations d'erreur

```
ErrorMsgContent ::= SEQUENCE {
 pKIStatusInfo PKIStatusInfo,
 errorCode INTEGER OPTIONAL,
 errorDetails PKIFreeText OPTIONAL
}
```

Ce message peut être généré à tout moment durant une transaction PKI. Si le client envoie cette requête, le serveur doit répondre avec une réponse PKIConfirm, ou un autre ErrorMsg si une partie de l'en-tête n'est pas valide. Chaque partie doit traiter ce message comme la fin de

la transaction (si une transaction est en cours).

Si la protection est souhaitée dans le message, le client doit le protéger en utilisant la même technique que le message de début de la transaction. La CA doit toujours le signer avec une clé de signature.

Demande et réponse d'interrogation

Cette paire de message est prévue pour manipuler les scénario dans lequel le client a besoin d'interroger le serveur pour déterminer le statut d'une transaction ir, cr, ou kur en cours. (par exemple quand le PKIStatus "waiting" a été reçu).

```
PollReqContent ::= SEQUENCE OF SEQUENCE {
 certReqId INTEGER }

PollRepContent ::= SEQUENCE OF SEQUENCE {
 certReqId INTEGER,
 checkAfter INTEGER, - time in seconds
 reason PKIFreeText OPTIONAL }
```

Les clauses suivantes décrivent quand les messages d'interrogation sont utilisés, et comment ils sont utilisés. Il est assumé que plusieurs messages certConf peuvent être envoyés durant les transactions. Il y'en aura un envoyé en réponse à chaque ip, cp, kup qui contient un CertStatus pour un certificat émis.

1. En réponse à un message ip, cp, ou kup, un EE va envoyer un certConf pour tous les certificats émis et, suivant le ack, un pollReq pour tous les certificats en attente.
2. En réponse à un message à un pollReq, une CA/RA va retourner un ip, cp, kup si un ou plusieurs certificats sont prêts ; sinon, elle retourne un pollRep
3. Si l'EE reçois un pollRep, il attend au moins checkAfter avant d'envoyer un autre pollReq
4. Si nu ip, cd, ou kup est reçu en réponse à un pollReq, alors il sera traité de la même manière que la réponse initiale.

Dans l'échange suivant, l'EE s'enregistre pour 2 certificats en une seule demande :

```
Step__End_Entity_____________________PKI
____1___Format_ir
____2____________________->_ir______->
____3_____________________________Handle_ir
____4_____________________________Manual_intervention_is
_____________________________________required_for_both_certs.
____5____________________<-_ip______<-
____6___Process_ip
____7___Format_pReq
____8____________________->_pReq_____->
____9_____________________________Check_status_of_cert_requests
____10____________________________Certificates_not_ready
____11____________________________Format_pRep
____12___________________<-_pRep_____<-
____13__Wait
____14__Format_pReq
____15___________________->_pReq_____->
____16____________________________Check_status_of_cert_requests
____17____________________________One_certificate_is_ready
____18____________________________Format_ip
____19___________________<-_ip_______<-
____20__Handle_ip
____21__Format_certConf
____22___________________->_certConf_->
____23____________________________Handle_certConf
____24____________________________Format_ack
____25___________________<-_pkiConf___<-
____26__Format_pReq
____27___________________->_pReq_____->
____28____________________________Check_status_of_certificate
____29____________________________Certificate_is_ready
____30____________________________Format_ip
____31___________________<-_ip_______<-
____31__Handle_ip
____32__Format_certConf
____33___________________->_certConf_->
____34____________________________Handle_certConf
____35____________________________Format_ack
____36___________________<-_pkiConf__<-
```

Fonctions de gestion de PKI obligatoire

Cette section décris les fonctions qui sont obligatoires dans le sens que toutes les implémentations EE, CA et RA doivent être capable de fournir les fonctionnalités décrites.

Initialisation de la CA root

Une CA root nouvellement créée doit produire un certificat auto-émis, qui est une structure Certificate avec le profile définis pour le certificat "newWithNew" émis par la mise à jours de la clé CA.

Pour que le certificat de la CA soit utile aux entités finales qui ne l'obtiennent pas par des moyens tiers, la CA doit également produire une

empreinte pour son certificat. La structure de données utilisée pour gérer l'empreinte est OOBCertHash.

Mise à jours de la clé CA

Les clés CA (comme toutes les autres clés) on une durée de vie finie et doivent être mis à jours périodiquement. Les certificats NewWithNew, NewWithOld, et OldWithNew peuvent être émis par la CA pour aider les EE existantes qui possèdent l'ancien certificat auto-signé (OldWithOld) à passer de manière sécurisée au nouveau certificat (NewWithNew), et pour aider les nouvelles entités qui ont NewWithNew à obtenir OldWithOld pour la vérification de donné existante.

Initialisation de CA subordonnée

Du point de vue des protocoles de gestion de PKI, l'initialisation d'une CA subordonnée est la même que l'initialisation d'un entité finale. La seule différence est que la CA subordonnée doit également produire une liste de révocation initiale.

Production de CRL

Avant d'émettre des certificats, une CA nouvellement établie (qui émet des crls) doit produire des versions vides de chaque CRL qui sont périodiquement produites.

Demande d'information PKI

Quand une entité PKI (CA, RA, ou EE) souhaite acquérir des informations sur le status courant d'une CA, elle peut lui envoyer une demande pour de telles informations.

La CA doit répondre à le demande en fournissant toutes les informations demandées. Si certaines informations ne peuvent pas être fournies, une erreur doit être retournée.

Si des PKIMessages sont utilisés pour demander et fournir cette information, alors la requête doit être le message GenMsg, la réponse doit être le message GenRep, et l'erreur doit être le message Error. Ce messages sont protégés en utilisant un MAC basé sur une information de secret partagé, ou en utilisant un autre moyen authentifié (si l'EE a un certificat existant).

Cross Certification

Le demandeur CA est la CA qui va devenir le sujet du cross-certificat ; le répondeur CA va devenir l'émetteur du cross-certificat. Le demandeur CA doit être "up and running" avant d'initialiser l'opération de cross-certification.

Schéma demande-réponse une voie

Le schéma de cross-certification est essentiellement une opération one-way ; c'est à dir, quand réussit, cette opération résulte en la création d'un nouveau cross-certificat. Si le pré-requis est que ce cross-certificat soit créé dans les 2 direction, alors chaque CA, en retour,

doit initialiser une opération de cross-certification (ou utiliser un autre schéma).

Ce schéma est prévu quand les 2 CA en question peuvent déjà se vérifier mutuellement leur signature. Description détaillée :

La cross certification est initiée à une CA connue comme répondeur. L'administrateur CA pour le répondeur identifie la CA qu'il veut cross certifier et l'équipement du répondeur génère un code d'autorisation. L'administrateur du répondeur passe ce code d'autorisation à la CA demandeuse via un moyen externe à l'administrateur de la CA demandeuse, qui utilise ce code pour initier l'échange on-line.

Le code d'autorisation est utilisé pour l'authentification et l'intégrité. C'est fait en générant une clé symétrique basée sur le code d'autorisation et en utilisant la clé symétrique pour générer des MAC pour tous les messages échangés. (Une authentification peut alternativement être faite en utilisant les signature au lieu des MAC, si les CA sont capable de récupérer et valider les clés publiques requise.)

La CA demandeuse initie l'échange en générant un demande de cross-certification (ccr) avec un nouveau nombre aléatoire. La CA demandeuse envoie le message ccr au répondeur. Les champs dans ce message sont protégés des modification avec un MAC basé sur le code d'autorisation.

À la réception du message ccr, le répondeur valide le message et le MAC, sauvegarde le nombre aléatoire du demandeur, et génère son propre nombre aléatoire. Il génère ainsi un nouveau certificat qui contient la clé publique de la CA demandeuse et est signée avec la clé privée du répondeur. Le répondeur répond avec un message ccp. Les champs dans ce message sont protégés des modification avec un MAC basé sur le code d'autorisation.

À la réception du message ccp, la CA demandeuse valide le message et le MAC. la CA demandeuse répond avec le message certConf. Les champs dans ce message sont protégés des modifications avec un MAC basé sur le code d'autorisation. La CA demandeuse peut écrire le certificat dans le dépôt comme aide pour la construction de future chemins de certificats.

Une fois le message certConf reçu, le répondeur valide le message et le MAC, et envoi un accusé en utilisant le message PKIConfirm. Il peut également publier le certificat.

Notes

1. Le message ccr doit contenir une demande de certification complète : c'est à dire, tous les champs exceptés le numéro de série doivent être spécifiés par la CA demandeuse.
2. Le message ccp devrait contenir le certificat de vérification du répondeur ; si présent, la CA demandeuse doit vérifier ce certificat.

Une version plus simple et non-interactive de la cross-certification peut également être envisagé, dans lequel la CA émettrice obtient la clé publique de la CA depuis un dépôt, la vérifie via un mécanisme externe, et créé et publie le cross-certificat sans le concours explicite de la CA.

Initialisation de l'entité finale

Comme avec les CA, les entités finales doivent être initialisées. L'initialisation des entités finales nécessite au moins 2 étapes :

- Acquisition des informations de la PKI
- Vérification tiers d'une clé publique de CA root

Acquisition d'information PKI

Les informations requises sont :

- La clé publique de la CA racine courante
- (Si la CA certifiante n'est pas la CA racine) le chemin de certification depuis la CA racine jusqu'à la CA certifiante, avec les listes de révocation appropriés.
- Les algorithmes et paramètres d'algorithme que la CA certifiante supporte pour chaque utilisation.

Des informations additionnelles peuvent être requises (ex : extensions supportées ou informations de stratégie de CA) pour pouvoir produire une demande de certification qui sera réussie. Cependant, par simplicité, on n'oblige par l'EE d'obtenir ces information via les messages PKI. Le résultat final est simplement que certaines demandes de certification peuvent échouer (ex : si l'EE veut générer sa propre clé de chiffrement, mais que la CA ne le permet pas).

Vérification Out-of-Band de la clé CA racine

Une EE doit posséder de manière sécurisée la clé publique de sa CA racine. Une méthode consiste à fournir à l'EE une empreinte du certificat de la CA via un moyen externe sécurisé. L'EE peut ainsi utiliser le certificat de la CA.

Demande de certificat

Une EE initialisée peut demander un certificat additionnel à tout moment. Cette demande sera faite en utilisant le message de demande de certification (cr). Si l'EE possède déjà une paire de clé de signature (avec un certificat de vérification correspondant), alors ce message cr sera protégé par la signature numérique de l'entité. La CA retourne le nouveau certificat dans un CertRepMessage.

Mise à jour de clé

Quand une paire de clé va expirer, l'entité finale peut demander une mise à jour de clé ; c'est à dire, qu'elle peut demander à la CA d'émettre un nouveau certificat pour une nouvelle paire de clé (ou, dans certaines circonstances, un nouveau certificat pour la même paire de clé (. La demande est faite en utilisant un message de demande de mise à jour de clé (kur). Si l'EE possède déjà une paire de clé de signature (avec le certificats de vérification correspondant), alors ce message sera typiquement protégé par la signature numérique de l'entité. La CA retourne le nouveau certificat (si la demande est réussie) dans une réponse de mise à jour de clé (kup), qui est syntaxiquement identique à un CertRepMessage.

Négociation de version

Cette section définis la négociation de version utilisée pour supporter d'anciens protocoles entre client et serveurs.

Si un client connaît les versions de protocoles supportés par le serveur (ex : depuis un précédent échange PKIMessage ou via un moyen externe), il doit envoyer un PKIMessage avec la version la plus haute supportée par lui et le serveur. Si un client ne connaît pas les versions supportées par le serveur, il doit envoyer un PKIMessage en utilisant la version la plus élevée qu'il supporte.

Si un serveur reçoit un message avec une version qu'il supporte, alors la version du message de réponse doit être la même que la version reçue. Si un serveur reçoit un message avec une version plus élevée ou moins élevée qu'il ne supporte, il doit envoyer en ErrorMsg avec le bit unsupportedVersion (dans le champ failureInfo de pKIStatusInfo). Si la version reçue est supérieur à la versions supportée, la version dans le message d'erreur doit être la plus haute version que le serveur supporte ; si la version reçue est inférieur à la version la plus basse supportée par le serveur alors la version dans le message d'erreur doit être la version la plus basse que le serveur supporte.

Si un client reçoit un ErrorMsgContent avec le bit unsupportedVersion et une version qu'il supporte, il peut retenter la requête avec cette version.

Support des implémentation rfc2510

La rfc2510 ne spécifie pas le comportement des implémentation recevant des versions qu'il ne comprend pas vu qu'il ne comprend qu'une seule version. Avec l'introduction de cette révision, le comportement suivant sur le versionning est recommandé.

Clients parlant aux serveurs rfc2510

Si, après avoir envoyé un message cmp2000, un client reçois un ErrorMsgContent avec une version de cmp1999, alors il doit annuler la transaction. Il peut ensuite retenter la transaction en utilisant des message cmp1999.

Si un client reçois un PKIMessage non-erreur avec une version de cmp1999, il peut décider de continuer la transaction en utilisant les sémantiques rfc2510. S'il choisis de ne pas le faire et que la transaction n'est pas terminée, il doit annuler la transaction et envoyer un ErrorMsgContent avec une version de cmp1999.

Serveurs reçevant des PKIMessage cmp1999

Si un serveur reçoit un message cmp1999, il peut revenir à un comportement rfc2510 et répondre avec des messages cmp1999. S'il choisi de ne pas le faire, il doit envoyer un ErrorMsgContent.

Considérations de sécurité

POP avec une clé de déchiffrement

Dans les protocoles spécifiés avant, quand une entité finale doit prouver la possession d'un clé de déchiffrement, il est effectivement challengé pour déchiffrer quelque-chose (son propre certificat). Ce schéma (et beaucoup d'autres) peut être vulnérable à une attaque si le propriétaire de la clé de déchiffrement en question peut être dupé en déchiffrant un challenge arbitraire et en retournant le texte en clair à un attaquant. Bien que dans cette spécification, d'autres problèmes de sécurité sont requis pour que cette attaque réussise, il est concevable que des services futures pourraient être vulnérable à de telles attaques. Pour cette raison, on réitère la règle générale que les implémentations devraient suivre sur le déchiffrement arbitraire et la révélation du texte récupéré.

POP en exposant la clé privée

Noter également qu'exposer une clé priée à la CA/RA comme technique POP peut exposer à des risques de sécurité. Les implémenteurs doivent faire preuve de prudence en sélectionnant et utilisant ce mécanisme POP particulier.

Appendice A. Raisons de la présence des RA

Les raisons qui justifient la présence d'un RA peut être séparés en ceux, d'un part, qui sont dûs à des facteurs techniques, et ceux qui d'autre part sont de nature organisationnelles. Les raisons techniques sont les suivante :

- Si des jetons hardwares sont utilisé, toutes les entités finales n'ont pas l'équipement nécessaire à les initialiser ; l'équipement RA peut inclure les fonctionnalités nécessaires.
- Certaines entités finales peuvent ne pas avoir la capacité de publier les certificats ; de même, la RA peut être utilisé pour cela.
- La RA sera capable d'émettre des demandes de révocation signées à la demande des entités finales associées avec elle, alors que l'entité final peut ne pas être en mesure de le faire (Si la paire de clé est complètement perdue).

Certaines raisons organisationnelles qui demandent la présente d'une RA sont :

- Il peut y avoir une raison de coût de concentrer les fonctionnalités dans un équipement RA plutôt que de fournir ces fonctionnalités à tous les EE.
- Établir des RA dans une organisation peut réduire le nombre de CA requises, ce qui est parfois désirable
- Pour beaucoup d'applications, Il y aura déjà en place une structure administrative pour que les candidats pour le rôle de RA soit facile à trouver.

Appendice B. Utilisation de passphrase de révocation

Une demande de révocation doit incorporer des mécanismes de sécurité prévu, incluant une authentification propre, pour réduire la probabilité d'attaques DOS réussies. Une signature numérique dans la requête peut fournir l'authentification requise, mais il y a des circonstances sous lesquelles un mécanisme alternatif peut être souhaité (par ex : quand une clé privée n'est plus accessible et que l'entité souhaite demander une révocation avant de re-certifier une autre paire).

Pour de telles circonstances, un PasswordBasedMac dans la demande est également obligatoire pour supporter cette spécification si les demandes de révocations sont également supportées et si les informations de secret partagé peuvent être établis entre le demandeur et le répondeur avant le besoin de révocation.

Un mécanisme qui est utilisé dans certains environnement est la passphrase de révocation, dans lequel une valeur d'entropy suffisante (ex : une passphrase relativement plus longue qu'un mot de passe court) est partagé entre (seulement) l'entité et la CA/RA à un certain moment avant la révocation ; cette valeur est ensuite utilisé pour authentifier la demande de révocation.

Dans cette spécification, la technique suivante pour établir une information à secret partagé est optionnelle. Son utilisation précise dans les messages CMP sont comme suit.

L'OID et la valeur spécifié dans la section "passphrase de révocation" peuvent être envoyé dans un message GenMsg à tout moment, ou peut être envoyé dans le champ generalInfo du PKIHeader d'un PKIMessage a tout moment. (En particulier, EncryptedValue peut être envoyé dans l'en-tête du message CertConf qui confirme l'acceptation des certificats requis dans une demande d'initialisation ou une demande de certification.). Cela transmet une passphrase de révocation choisie par l'entité (ex : les octets déchiffrés du champ encValue) à la CA/RA ; en outre, le transfert est accomplis avec les caractéristiques de confidentialité appropriés (parce que la passphrase est chiffrée avec le protocolEncryptionKey de la CA/RA).

Si une CA/RA reçois la passphrase de révocation dans un GenMsg, elle doit construire et envoyer un message GenRep qui inclus l'OID (avec une valeur absente) spécifiée dans la section "passphrase de révocation". Si la CA/RA reçois la passphrase de révocation dans le champ generalInfo d'un PKIHeader d'un PKIMessage, elle doit inclure l'OID (avec une valeur absente) dans le champ generalInfo du PKIHeader du PKIMessage de réponse correspondant. Si la CA/RA n'arrive pas à retourner le message de réponse approprié, elle doit envoyer un message d'erreur avec un status de rejet et optionnellement, un jeu de raison failInfo.

Le champ valueHint de EncryptedValue peut contenir un identifiant de clé (choisi par l'entité finale, avec la passphrase elle-même) pour assister à la récupéraion de la passphrase correcte (ex, quand la demande de révocation est construite par l'entité et reçue par la CA/RA).

Le messages de demande de révocation est protégé par un PasswordBasedMac, avec la passphrase de révocation comme clé. Si approprié, le champ senderKID dans le PKIHeader peut contenir la valeur précédemment transmise dans valueHint.

En utilisant la technique spécifiée ci-dessus, la passphrase de révocation peut être initialement établie et mise à jour à tout moment sans nécessiter de messages supplémentaires ou d'échange externes. Par exemple, le message de demande de révocation lui-même (protégé et authentifié via un MAC qui utilise la passphrase de révocation comme clé) peut contenir, dans le PKIHeader, une nouvelle passphrase de

révocation à utiliser pour authentifier de futures demandes de révocation pour tout autre certificat de l'entité. Dans certains environnements cela peut être préférable aux mécanismes qui révèlent la passphrase dans le message de demande de révocation, vu que cela peut permettre une attaque DOS dans laquelle la passphrase révélée est utilisée par un tier non-autorisé pour authentifier les demandes de révocation pour les autres certificat de l'entité. Cependant, parce que la passphrase n'est pas révélée dans la requête, il n'y a pas de pré-requis pour que la passphrase soit toujours mise à jours quand la demande de révocation est faite (c'est à dire, la même passphrase peut être utilisée par une entité pour authentifier les demandes de révocation pour différents certificats à différents moments).

De plus, la technique ci-dessus peut fournir une protection cryptographique forte sur tout le message de demande de révocation même quand une signature numérique n'est pas utilisée. Les techniques qui font de l'authentification de la demande de révocation en révélant simplement la passphrase de révocation ne fournissent pas de protection cryptographique sur les champs du message (donc une demande de révocation d'un certificat peut être modifié par un tier non autorisé pour révoquer un autre certificat pour cette entité).

Appendice C - Clarification du conportement des messages de demande

Dans le cas des mises à jours de la rfc4211, qui apporte des problème d'interprétation ou d'intéropérabilité, la rfc4211 devrait être le document normatif. Les définitions suivante viennent de la rfc4211. Elles sont inclues ici pour codifier les clarifications de comportement ; sinon toutes les syntaxes et sémantiques sont identiques à la rfc4211.

```
CertRequest ::= SEQUENCE {
 certReqId INTEGER,
 certTemplate CertTemplate,
 controls Controls OPTIONAL }

- Si certTemplate est une séquence vide, alors les contrôles peuvent contenir le contrôle
- id-regCtrl-altCertTemplate, spécifiant un template pour un certificat autre qu'un
- certificat à clé-publique x509v3. Inversement, si certTemplate n'est pas vide (au moins
- un champ est présent), alors les contrôles ne doivent pas contenir id-regCtrl-altCertTemplate.
- Le nouveau contrôle est définis comme suit:

id-regCtrl-altCertTemplate OBJECT IDENTIFIER ::= {id-regCtrl 7}
AltCertTemplate ::= AttributeTypeAndValue

POPOSigningKey ::= SEQUENCE {
 poposkInput [0] POPOSigningKeyInput OPTIONAL,
 algorithmIdentifier AlgorithmIdentifier,
 signature BIT STRING }

- La signature (en utilisant AlgorithmIdentifier) est dans la valeur encodée DER de poposkInput.
- Note: si CertReqMsg certReq certTemplate ( ou le contrôle altCertTemplate) contient les valeurs
- subject et publicKey, alors poposkInput doit être omis et la signature doit être calculée dans
- la valeur encodée DER de CertReqMsg certReq ( ou la valeur encodée DER de AltCertTemplate).
- Si certTemplate/altCertTemplate ne contient ni le sujet ni la clé publique, poposkInput doit
- être présent et doit être signé.

POPOPrivKey ::= CHOICE {
 thisMessage [0] BIT STRING,

- Le type de "thisMessage" est donné en chaîne de bit dans la rfc4211; mais devrait être un
- "EncryptedValue", en accord avec cette spécification.

 subsequentMessage [1] SubsequentMessage,
 dhMAC [2] BIT STRING }
```

Appendice D. profils de messages de gestion

Cet appendice contient les profils détaillés pour ces PKIMessages qui doivent être supportés par les implémentations conformes. Les profils pour les PKIMessages utilisés dans les opérations de gestion PKI suivant sont fournis :

- Enregistrement/certification initiale
- Schéma authentifié de base
- Demande de certificat
- Mise à jour de clé

Règles générales pour l'interprétation de ces profils

1. Où les champs OPTIONAL ou DEFAULT ne sont pas mentionnés dans les profiles, ils doivent être absent du message. Les champs obligatoire ne sont pas mentionnés s'il ont une valeur évidente. (ex dans la version de cette spécification, pvno est toujours 2).
2. Où les structures se trouvent dans plus d'un message, elles sont profilés séparément comme appropriées.
3. Les algorithmIdentifier des structures PKIMessage sont profilés séparément.
4. Un DN X.500 spécial est appelé le "NULL-DN" ; cela signifie un DN contenant une séquence RelativeDistinguishedNames de longueur 0 (son encodé DER est alors '3000'H).
5 Où un GeneralName est requis pour un champ, mais qu'aucune valeur prévue n'est disponible (ex : une entité produit une demande avant de connaître son nom), alors le GeneralName doit être un NULL-DN. Cette valeur spéciale peut être appelée NULL-GeneralName.
6. Où un profil omit de spécifier la valeur pour un GeneralName, la valeur NULL-GeneralName doit être présente dans le champs PKIMessage concerné.
7. Où toute ambiguïté peut se produire à cause du nommage des champs, le profile les nomme en utilisant une notation "point" (ex : certTemplate.subject signifie le champ subject dans un champ appelé certTemplate).
8. Où une séquence de types fait partie d'un message, une notation en tableau basé sur des 0 est utilisée pour décrire les champs dans la séquence (ex : crm[0].certReq.certTemplate.subject réfère à un sous-champ du premier CertReqMsh contenu dans un message de demande).
9. Tous les échanges de message PKI décris plus bas dans cet appendice nécessitent qu'un message certConf soit envoyé par l'entité qui initie et un PKIConfirm par l'entité qui répond. Le PKIConfirm n'est pas inclus dans certains profiles donnés vu que sont contenu est NULL et son header peut être utilisé pour le protectionAlg.

Utilisation des algorithmes

La table suivante contient les définitions de l'utilisation d'algorithme dans les protocoles de gestion PKI. Les colonnes dans la table sont :

Name : Un identifiant utilisé pour les profiles de message

Use : Une description d'ou et pourquoi l'algorithme est utilisé

Mandatory : Un AlgorithmIdentifier qui doit être supporté par les implémentations conformes.

others : Alternatives à AlgorithmIdentifier.

Name	Use	Mandatory	Others
MSG_SIG_ALG	Protection des messages en utilisant une signature	DSA/SHA-1	RSA/MD5,ECDSA,...
MSG_MAC_ALG	Protection des messages en utilisant MAC	PasswordBasedMac	HMAC, X9.9,...
SYM_PENC_ALG	Chiffrement symétrique de la clé privée de l'EE	3-DES (3-key-EDE,CBC mode)	AES,RC5,CAST-128,...
PROT_ENC_ALG	Chiffrement asymétrique de la clé privée	D-H	RSA,ECDH,...
PROT_SYM_ALG	Chiffrement symétrique des bits de la clé privée	3-DES (3-key-EDE,CBC mode)	AES,RC5,CAST-128,...

AlgorithmIdentifier mandatoire et spécifications :

```
DSA/SHA-1:
 AlgId: {1 2 840 10040 4 3};

Digital Signature Standard [FIPS-186]
 Public Modulus size: 1024 bits.

PasswordBasedMac:
 AlgId: {1 2 840 113533 7 66 13}, avec SHA-1 {1 3 14 3 2 26} et HMAC-SHA1 {1 3 6 1 5 5 8 1 2};

Secure Hash Standard [FIPS-180] and [RFC2104]
 HMAC key size: 160 bits (i.e., "K" = "H" in Section 5.1.3.1, "Shared secret information")

3-DES:
 AlgId: {1 2 840 113549 3 7}; (used in RSA's BSAFE and in S/MIME).

D-H:
 AlgId: {1 2 840 10046 2 1};

[ANSI-X9.42]
 Public Modulus Size: 1024 bits.
 DomainParameters ::= SEQUENCE {
  p INTEGER, - odd prime, p=jq +1
  g INTEGER, - generator, g^q = 1 mod p
  q INTEGER, - prime factor of p-1
  j INTEGER OPTIONAL, - cofactor, j>=2
  validationParms ValidationParms OPTIONAL
 }
 ValidationParms ::= SEQUENCE {
  seed BIT STRING, - seed for prime generation
  pGenCounter INTEGER - parameter verification
 }
```

Profile POP

Les champs POP utilisé (dans le champ signature du champ pop de la structure ProofOfPossession) en prouvant la possession d'une clé privée de signature qui correspond à une clé publique de vérification pour laquelle un certificat a été demandé.

```
Field_____Value_____Comment
algorithmIdentifier_MSG_SIG_ALG___Seul la protection de la signature est permise pour cette preuve
signature_____present_____bits calculés en utilisant MSG_SIG_ALG
```

La preuve de possession d'une clé privée de déchiffrement qui correspond à une clé public de chiffrement pour laquelle un certificat a été demandé n'utilise pas ce profile ; le champ CertHash du message certConf est utilisé à la place.

Toutes les CA/RA ne font pas de POP dans le protocole de demande de certification (comment POP est faite peut être un problème de stratégie explicite pour une CA). Cependant, cette spécification mandate que les entités CA/RA doivent faire une POP comme partie du processus de certification. Toutes les entités finales doivent être préparée à fournir une POP.

Enregistrement/certification initial - schéma authentifié de base

Une entité finale (non initialisée) demande un (premier) certificat à une CA. Quand la CA répond avec une message contenant un

certificat, l'EE répond avec une confirmation de certificat. La CA envoie en PKIConfirm en retour, terminant la transaction. Tous les message sont authentifiés.

Ce schéma permet à l'EE de demander une certification d'une clé publique générée localement (typiquement une clé de signature). L'EE peut également choisir de demander la génération et la certification centralisée d'une autre paire de clé (typiquement une paire de clé de chiffrement). La certification peut seulement être demandée pour en clé publique générée localement.

L'EE doit supporter le POP d'une clé privée, associée avec la clé publique générée localement.

Préconditions :

1. L'EE peut authentifier la signature de la CA via un moyen externe
2. L'EE et la CA partage une clé MAC symétrique

Flux de messages :

```
Step#_End_entity_____________________________PKI
______1___format_ir
______2_____________________->___ir______->
______3_____________________________________handle_ir
______4_____________________________________format_ip
______5_____________________<-___ip______<-
______6___handle_ip
______7___format_certConf
______8_____________________->___certConf_->
______9_____________________________________handle_certConf
_____10_____________________________________format_PKIConf
_____11_____________________<-___PKIConf__<-
_____12___handle_PKIConf
```

Pour ce profile, on mandate que l'EE doit inclure tous CertReqMsg dans un simple PKIMessage, et que le PKI (CA) doit produire une seul réponse PKIMessage qui contient la réponse complète (ex : incluant la seconde paire de clé optionnelle, s'il elle a été demandée et si la génération de la clé centralisée est supportée). Par simplicité, on mandate également que ce message doit être le final (ex : pas d'utilisation du status "waiting").

L'EE a une interaction out-of-band avec la CA/RA. Cette transaction établis le secret partagé, le referenceNumber et optionnellement le dn utilisé pour sender et subject dans le template du certificat. Il est recommandé que le secret partagé ait au moins 12 caractères de long.

Demande de certificat

Une EE (initialisé) demande un certificat depuis une CA. Quand la CA répond avec un message contenant un certifica, l'EE répond avec une confirmation de certificat. La CA répond avec un PKIConfirm, pour terminer la transaction. Tous les messages sont authentifiés. Le profile pour cet échange est identique à celui donné précédemment excepté :

- Le nom de l'émetteur devrait être présent
- protectionAlg de MSG_SIG_ALG doit être supporté (MSG_MAC_ALG peut l'être également) dans le demande, la réponse, certConfirm et les PKIMessages.
- senderKID et recipKID sont seulement présents si requis pour la vérification du message
- body est cr ou cp
- body peut contenir une ou 2 structures CertReqMsg, mais CertReqMsg peut être utilisé pour demander une certification d'une clé publique générée localement ou une clé publique générée centralement.
- Les bits de protection sont calculés en accord avec le champ protectionAlg

Demande de mise à jour de clé

Une entité finale (initialisée) demande un certificat à une CA (pour mettre à jours la paire de clé et/ou le certificat correspondant qu'il possède déjà). Quand la CA répond avec un message contenant un certificat, l'EE répond avec une confirmation de certificat. La CA répond avec un PKIConfirm, pour terminer la transaction. Tous les messages sont authentifiés. Le profile pour cet échange est identique à celui donné précédemment excepté :

- Le nom de l'émetteur devrait être présent
- protectionAlg de MSG_SIG_ALG doit être supporté (MSG_MAC_ALG peut l'être également) dans le demande, la réponse, certConfirm et les PKIMessages.
- senderKID et recipKID sont seulement présents si requis pour la vérification du message
- body est kur ou kup
- body peut contenir une ou 2 structures CertReqMsg, mais CertReqMsg peut être utilisé pour demander une certification d'une clé publique générée localement ou une clé publique générée centralement.
- Les bits de protection sont calculés en accord avec le champ protectionAlg
- regCtrl OldCertId devrait être utilisé

Appendice E : Profile de message de gestion PKI

Cet appendice contient les profile pour les PKIMessages qui peuvent être supportés par les implémentations. Les profiles pour les PKIMessages utilisé dans les opération de gestion PKI suivant sont fournis :

- Mise à jours de clé CA racine
- Demande/réponse d'informations
- Demande/réponse de cross-certification
- Initialisation in-band en utilisant en certificat d'identité externe.

Certificats auto-signés

Le profile sur la manière dont un certificat peut être auto-signé. Ces structures sont utilisées pour les distributions de clés publiques de CA. Cela peut se produire d'une des 3 manières.

```
Type_____Fonction
newWithNew__Une vrai certificat auto-signé; la clé publique contenue doit être utilisable pour vérifier la signature
oldWithNew__La précédente clé CA Root signée avec la nouvelle clé privée.
newWithOld__La nouvelle clé publique CA root signée avec la précédente clé privée.
```

De tels certificats doivent contenir les valeurs sensibles pour tous les champs. Par exemple, quand présent, subjectAltName doit être identique à issuerAltName, et, quand présent, keyIdentifier doit contenir les valeurs appropriées, etc.

Mise à jours de clé Root

Une CA racine met à jour sa paire de clé. Elle produit ainsi un message d'annonce de mise à jours de clé CA qui peut être disponible (via certains mécanismes de transport) aux EE concernées. Un message de confirmation n'est pas requis pour les EE.

Demande/réponse d'information PKI

L'entité finale envoie un message général à la PKI demandant des détails qui sont requis pour d'autres opérations de gestion PKI. La RA/CA répond avec une réponse générale. Si une RA génère la réponse, alors il va simplement transférer le message équivalent qui a été précédemment reçus de la CA, en ajoutant possiblement les certificats dans les champs extraCerts du PKIMessage. Un message de confirmation n'est pas requis pour les EE.

Demande/réponse de cross-certification

La création d'un simple cross-certificat (ex : pas 2 en même temps). La CA demandeuse peut choisir qui est responsable pour la publication du cross-certificat créé par la CA répondante via l'utilisation du contrôle PKIPublicationInfo.

Préconditions :

1. La CA répondante peut vérifier l'origine de la demande avant de traiter la demande.
2. La CA demandeuse peut authentifier l'authenticité de l'origine de la réponse avant de traiter la réponse.

L'utilisation de confirmation de certification et la confirmation du serveur correspondant est déterminé par le champ generalInfo dans le PKIHeader. Le profile suivant ne mandate pas le supporte pour ces confirmations.

Initialisation in-band via certificat externe

Une EE (non-initialisée) souhaite s'initialiser dans la PKI avec une CA, CA-1. Elle utilise, pour l'authentification, un certificat d'identité pré-existant émis par une autre CA (externe), CA-X. Une relation de confiance doit déjà avoir été établis entre CA-1 et CA-X pour que CA-1 puisse valider le certificat de l'EE signé par CA-X. De plus, certains mécanismes doivent déjà avoir été établis dans le PSE de l'EE qui lui permet d'authentifier et vérifier les PKIMessage signés par CA-1.

L'EE envoie une demande d'initialisation pour démarrer la transaction. Quand CA-1 répond avec un message contenant le nouveau certificat, l'EE répond avec une confirmation de certificat. CA-1 répond avec un PKIConfirm pour terminer la transaction. Tous les messages sont signés (les messages EE sont signés en utilisant la clé privée qui correspond à la clé publique dans son certificat d'identité externe, les messages CA-1 sont signés en utilisant la clé privée qui correspond à la clé publique dans un certificat qui peut être chaîné à une ancre de confiance dans le PSE de l'EE). Le profile pour cet échange est identique à celui donné précédemment excepté :

- L'EE et CA-1 ne partagent pas de clé MAC symétrique
- Le nom de l'émetteur dans ir doit être présent
- protectionAlg de MSG_SIG_ALG doit être utilisé dans tous les messages
- le certificat d'identité externe doit être géré dans le champ extraCerts de ir
- senderKID et recipKID ne sont pas utilisés.
- body est ir ou ip
- Les bits de protections sont calculés en accord avec le champ protectionAlg