Groupe de travail Réseau
Request for Comments : 4210
RFC rendue obsolète : 2510
Catégorie : Sur la voie de la normalisation
Traduction Claude Brière de L'Isle

C. Adams, University of Ottawa
S. Farrell, Trinity College Dublin
T. Kause, SSH
T. Mononen, SafeNet
septembre 2005

Protocole de gestion de certificat (CMP) d'infrastructure de clé publique X.509 de l'Internet

Résumé
Le présent document décrit le protocole de gestion de certificats (CMP, *Certificate Management Protocol*) de l'infrastructure de clé publique (PKI, *Public Key Infrastructure*) X.509 de l'Internet. Les messages du protocole sont définis pour la création et la gestion de certificat X.509v3. CMP fournit des interactions en ligne entre les composants de PKI, incluant un échange entre une autorité de certification (CA, *Certification Authority*) et un système client.

Table des matières

1. Introduction

Le présent document décrit le protocole de gestion de certificats (CMP, *Certificate Management Protocol*) de l'infrastructure de clé publique (PKI, *Public Key Infrastructure*) X.509 de l'Internet. Les messages du protocole sont définis pour la création et la gestion de certificat. Le terme de "certificat" dans le présent document se réfère à un certificat X.509v3 comme défini dans [X509].

La présente spécification rend obsolète la RFC 2510. La présente spécification diffère de la RFC 2510 dans les domaines suivants :

La section sur le profil du message de gestion de PKI est partagée entre deux appendices : le profil exigé et le profil facultatif. Certaines des fonctionnalités anciennement obligatoires ont été passées au profil facultatif.

Le mécanisme de confirmation de message a changé substantiellement.

Un nouveau mécanisme d'interrogation est introduit, l'ancienne méthode d'interrogation étant déconseillée au niveau du transport de CMP.

Les questions de protocole de transport de CMP sont traitées dans un document distinct [RFC6712], et donc la Section Transports est supprimée.

Une nouvelle méthode implicite de confirmation est introduite pour réduire le nombre de messages de protocole échangés dans une transaction.

La nouvelle spécification contient des améliorations moins importantes du protocole et du texte explicatif amélioré sur plusieurs questions.

2. Exigences

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

3. Vue d'ensemble de la gestion de PKI

La PKI doit être structurée de façon à être cohérente avec les types d'individus qui doivent l'administrer. Fournir à ces administrateurs des choix illimités non seulement complique le logiciel requis, mais aussi augmente les chances qu'une faute subtile d'un administrateur ou d'un développeur de logiciel résulte en un plus large compromis. De même, restreindre les administrateurs avec des mécanismes compliqués les amènerait à ne pas utiliser la PKI.

Il est EXIGÉ des protocoles de gestion qu'ils prennent en charge les interactions en ligne entre les composants d'infrastructure de clé publique (PKI). Par exemple, un protocole de gestion peut être utilisé entre une autorité de certification (CA) et un système client auquel une paire de clés est associée, ou entre deux CA qui produisent des certificats croisés l'un pur l'autre.

3.1 Modèle de gestion de PKI

Avant de spécifier des formats et procédures de message particuliers, on définit d'abord les entités impliquées dans la gestion de PKI et leurs interactions (en termes de fonctions de gestion de PKI requises). On groupe ensuite ces fonctions afin de s'accommoder des différents types identifiables d'entités d'extrémité.

3.1.1 Définitions des entités de PKI

Les entités impliquées dans la gestion de PKI incluent l'entité d'extrémité (c'est-à-dire, l'entité à laquelle le certificat est produit) et l'autorité de certification (c'est-à-dire, l'entité qui produit le certificat). Une autorité d'enregistrement PEUT aussi être impliquée dans la gestion de PKI.

3.1.1.1 Entités sujettes et d'extrémité

Le terme "sujet" est utilisé ici pour se référer à l'entité à laquelle le certificat est produit, normalement citée dans le champ "subject" ou "subjectAltName" d'un certificat. Quand on souhaite distinguer les outils et/ou le logiciel utilisé par le sujet (par exemple, un module local de gestion de certificat) on va utiliser le terme "équipement sujet". En général, le terme "entité d'extrémité" (EE), plutôt que "sujet", est préféré pour éviter la confusion avec le nom du champ. Il est important de noter que les entités d'extrémité ne vont ici inclure aucun utilisateur humain des applications, mais aussi les applications elles-mêmes (par exemple, pour la sécurité IP). Ce facteur influence les protocoles qu'utilisent les opérations de gestion de PKI ; par exemple, un logiciel d'application va très probablement savoir exactement quelles extensions de certificat sont requises contrairement aux utilisateurs humains. Les entités de gestion de PKI sont aussi des entités d'extrémité dans le sens où elles sont parfois désignées dans le champ "subject" ou "subjectAltName" d'un certificat ou certificat croisé. Lorsque approprié, le terme "entité d'extrémité" sera utilisé pour se référer aux entités d'extrémité qui ne sont pas des entités de gestion de PKI.

Toutes les entités d'extrémité exigent un accès local sûr à certaines informations -- au minimum, leur propre nom et leur clé privée, le nom d'une CA qui est directement de confiance pour cette entité, et la clé publique de cette CA (ou une empreinte digitale de la clé publique lorsque une version auto certifiée est disponible ailleurs). Les mises en œuvre PEUVENT utiliser une mémorisation locale sûre pour plus que ce minimum (par exemple, le propre certificat de l'entité d'extrémité ou des informations spécifiques de l'application). La forme de mémorisation va aussi varier – du fichier au jeton cryptographique résistant à l'altération. Les informations mémorisées dans une telle mémorisation locale de confiance sont appelées ici l'environnement de sécurité personnel (PSE, *Personal Security Environment*) de l'entité d'extrémité.

Bien que les formats de PSE sortent du domaine d'application du présent document (ils dépendent de l'équipement, et cetera) un format générique d'inter change des PSE est défini ici : un message de réponse de certification PEUT être utilisé.

3.1.1.2 Autorité de certification

L'autorité de certification (CA) peut être ou non un réel "tiers" du point de vue de l'entité d'extrémité. Assez souvent, la CA va en fait appartenir à la même organisation que les entités d'extrémité qu'elle prend en charge.

Là encore, on utilise le terme "CA" pour se référer à l'entité désignée dans le champ de producteur d'un certificat. Lorsque il est nécessaire de distinguer les outils logiciels ou matériels utilisés par la CA, on utilise le terme "équipement de CA".

L'équipement de CA sera souvent inclus à la fois dans le composant "hors ligne" et dans un composant "en ligne", avec la clé privée de CA disponible seulement au composant "hors ligne". Ceci est, cependant, une affaire de mise en œuvre (bien que ce soit aussi pertinent comme question de politique).

On utilise le terme "CA racine" pour indiquer une CA qui est directement de confiance pour une entité d'extrémité ; c'est-à-dire, que l'acquisition en toute sécurité de la valeur d'une clé publique de CA racine exige des étapes hors bande . Ce terme n'est pas destiné à impliquer qu'une CA racine est nécessairement au sommet d'une certaine hiérarchie, mais simplement que la CA en question est directement de confiance.

Une CA "subordonnée" est celle qui n'est pas une CA racine pour l'entité d'extrémité en question. Souvent, une CA subordonnée ne sera pas une CA racine pour une entité, mais ceci n'est pas obligatoire.

3.1.1.3 Autorité d'enregistrement

En plus des entités d'extrémité et des CA, de nombreux environnements appellent l'existence d'une autorité d'enregistrement (RA, *Registration Authority*) séparée de l'autorité de certification. Les fonctions que l'autorité d'enregistrement peut assumer peuvent varier d'un cas à l'autre, mais PEUVENT inclure authentification personnelle, la distribution de jetons, les rapports de révocation, les allocations de noms, la génération de clé, l'archivage d'une paire de clés, et cetera.

Le présent document voit la RA comme un composant FACULTATIF : quand il n'est pas présent, la CA est supposée être capable d'assurer les fonctions de RA de sorte que les protocoles de gestion PKI sont les mêmes du point de vue des entités d'extrémité.

Là encore, on distingue, lorsque nécessaire, entre la RA et les outils utilisés ("l'équipement de RA").

Noter qu'une RA est elle-même une entité d'extrémité. On suppose de plus que toutes les RA sont en fait des entités d'extrémité certifiées et que les RA ont des clés privées utilisables pour signer. Comment un équipement de CA particulier identifie des entités d'extrémité comme RA est une question de mise en œuvre (c'est-à-dire, le présent document ne spécifie pas de fonctionnement de certification de RA particulier). On ne rend pas obligatoire que la RA soit certifiée par la CA avec laquelle elle interagit sur le moment (une RA peut fonctionner avec plus d'une CA tout en n'étant certifiée qu'une fois).

Dans certaines circonstances, les entités d'extrémité vont communiquer directement avec une CA même lorsque une RA est présente. Par exemple, pour l'enregistrement et/ou certification initial, le sujet peut utiliser sa RA, mais communiquer directement avec la CA afin de rafraîchir son certificat.

3.1.2 Exigences d'enregistrement de PKI

Le protocole qu'on donne ici satisfait aux exigences suivantes à l'égard de la gestion de PKI :

1. La gestion de PKI doit se conformer aux normes ISO/CEI 9594-8/UIT-T X.509.
2. Il doit être possible de mettre à jour de façon régulière toute paire de clés sans affecter une autre paire de clés.
3. L'utilisation de la confidentialité dans les protocoles de gestion PKI doit être gardée à un minimum afin de faciliter son acceptation dans les environnements où une forte confidentialité peut causer des problèmes réglementaires.
4. Les protocoles de gestion PKI doivent permettre l'utilisation de différents algorithmes cryptographiques standard de l'industrie (incluant spécifiquement RSA, DSA, MD5, et SHA-1). Cela signifie que toute CA, RA, ou entité d'extrémité peut, en principe, utiliser tout algorithme qui lui convient pour sa ou ses propres paires de clés.
5. Les protocoles de gestion PKI ne doivent pas empêcher la génération de paires de clés par l'entité d'extrémité concernée, par une RA, ou par une CA. La génération de clé peut aussi se produite ailleurs, mais pour les besoins de la gestion de PKI, on peut regarder la génération de clés comme se produisant partout où la clé est d'abord présente à une entité d'extrémité, RA, ou CA.
6. Les protocoles de gestion PKI doivent prendre en charge la publication des certificats par les entités d'extrémité concernées, par une RA, ou par une CA. Des mises en œuvre et des environnements différents peuvent choisir une des approches ci-dessus
7. Les protocoles de gestion PKI doivent prendre en charge la production de listes de révocation de certificat (CRL, *Certificate Revocation List*) en permettant aux entités d'extrémité certifiées de faire des demandes sur la révocation de certificats. Ceci doit être fait d'une façon telle que les attaques de déni de service, qui sont possibles, ne soient pas rendues plus simples.
8. Les protocoles de gestion PKI doivent être utilisables sur divers mécanismes de "transport", incluant spécifiquement la messagerie, http, TCP/IP et ftp.
9. L'autorité finale pour la création de la certification appartient à la CA. Aucune RA ou équipement d'entité d'extrémité ne peut supposer qu'un certificat produit par une CA va contenir ce qui était demandé ; une CA peut altérer des valeurs de champ de certificat ou peut ajouter, supprimer ou altérer des extensions selon sa politique de fonctionnement. En d'autres termes, toutes les entités de PKI (entités d'extrémité, RA, et CA) doivent être capables de traiter les réponses aux demandes sur les certificats dans lesquels le certificat réel produit est différent de celui demandé (par exemple, une CA peut raccourcir la période de validité demandée). Noter que la politique peut imposer que la CA ne doive pas

publier ou distribuer d'autre façon le certificat jusqu'à ce que l'entité demandeuse ait revu et accepté le nouveau certificat créé (normalement par l'utilisation du message certConf).

10. Un changement en douceur programmé d'une paire de clé de CA non compromise à la prochaine mise à jour de clé de CA doit être prise en charge (noter que si la clé de CA est compromise, la réinitialisation doit être effectuée pour toutes les entités dans le domaine de cette CA). Une entité d'extrémité dont le PSE contient la nouvelle clé publique de CA (suite à la mise à jour de clé de CA) doit aussi être capable de vérifier les certificats vérifiables en utilisant la vieille clé publique. Les entités d'extrémité qui font directement confiance à la vieille paire de clés de CA doivent aussi être capables de vérifier les certificats signés en utilisant la nouvelle clé privée de CA (c'est exigé pour les situations où la vieille clé publique de CA est "incorporée" dans l'équipement cryptographique de l'entité d'extrémité).
11. Les fonctions d'une RA peuvent, dans certaines mises en œuvre ou environnements, être effectuées par la CA elle-même. Les protocoles doivent être conçus de telle sorte que les entités d'extrémité utilisent le même protocole sans considérer si la communication est avec une RA ou une CA. Naturellement, l'entité d'extrémité doit utiliser la RA correcte de la clé publique de CA pour protéger la communication.
12. Lorsque une entité d'extrémité demande un certificat contenant une certaine valeur de clé publique, l'entité d'extrémité doit être prête à démontrer la possession de la valeur de clé privée correspondante. Cela peut se faire de diverses façons, selon le type de demande de certification. Voir au paragraphe 4.3 les détails des méthodes dans la bande définies pour les messages PKIX-CMP (c'est-à-dire, du protocole de gestion de certificat).

3.1.3 Opérations de gestion de PKI

Le diagramme suivant montre les relations entre les entités définies ci-dessus en termes de fonctionnement de la gestion de PKI. Les lettres dans le diagramme indiquent les "protocoles" en ce sens qu'un ensemble défini de messages de gestion de PKI peut être envoyé le long de chaque ligne marquée d'une lettre.

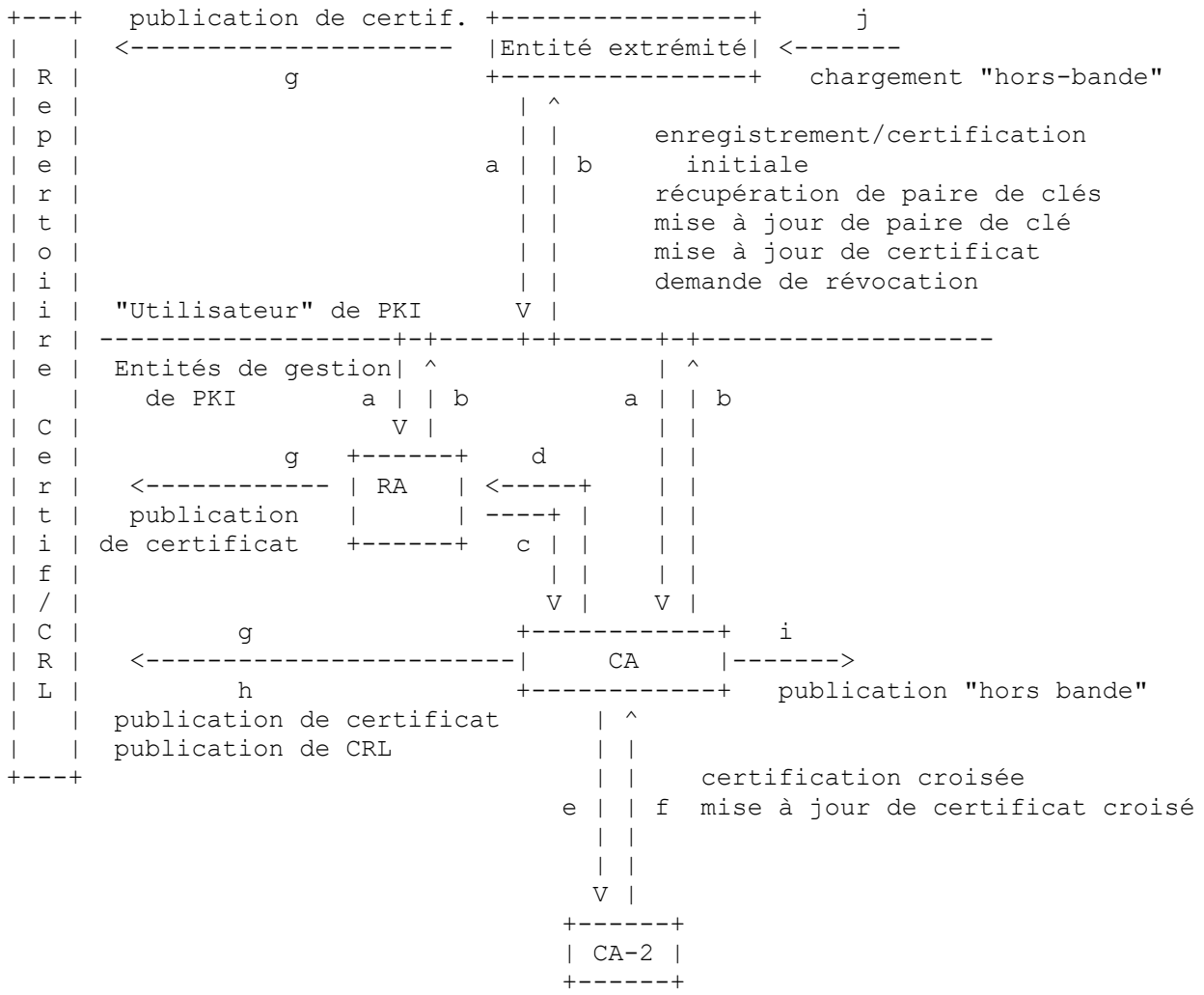


Figure 1 – Entités PKI

En gros, les ensembles d'opérations pour lesquelles des messages de gestion sont définis peuvent être groupés comme suit.

1. Établissement de CA : Lors de l'établissement d'une nouvelle CA, certaines étapes sont exigées (par exemple, production des CRL initiales, exportation de la clé publique de CA).

2. Initialisation d'entité d'extrémité : cela inclut d'importer la clé publique d'une CA racine et de demander des informations sur les options prises en charge par une entité de gestion de PKI.

3. Certification : diverses opérations résultent en la création de nouveaux certificats :
   1. Enregistrement/certification initial : c'est le processus par lequel une entité d'extrémité se fait d'abord connaître d'une CA ou RA, avant que la CA ne produise un ou des certificats pour cette entité d'extrémité. Le résultat final de ce processus (quand il réussit) est qu'une CA produit un certificat pour la clé publique d'une entité d'extrémité, et retourne ce certificat à l'entité d'extrémité et ou envoie ce certificat dans un répertoire public. Ce processus peut, et normalement va, impliquer plusieurs "étapes", incluant éventuellement une initialisation de l'équipement de l'entité d'extrémité. Par exemple, l'équipement de l'entité d'extrémité doit être initialisé de façon sûre avec la clé publique d'une CA, pour être utilisé à valider les chemins de certificats. De plus, une entité d'extrémité a normalement besoin d'être initialisée avec sa ou ses propres paires de clés.
   2. Mise à jour de paire de clés : chaque paire de clés doit être mise à jour régulièrement (c'est-à-dire, remplacée par une nouvelle paire de clés) et un nouveau certificat doit être produit.
   3. Mise à jour de certificat : lorsque les certificats arrivent à expiration, ils peuvent être "rafraîchis" si rien de pertinent n'a changé dans l'environnement.
   4. Mise à jour de paire de clés de CA : comme avec les entités d'extrémité, les paires de clés de CA doivent être mises à jour régulièrement ; cependant, différents mécanismes sont nécessaires.
   5. Demande de certification croisée : une CA demande la production d'un certificat croisé par une autre CA. Pour les besoins de la présente norme, on définit les termes suivants. Un "certificat croisé" est un certificat dans lequel la CA sujette et la CA productrice sont distinctes et SubjectPublicKeyInfo contient une clé de vérification (c'est-à-dire que le certificat a été produit pour la paire de clés signataire de la CA sujette). Quand il est nécessaire de faire une distinction plus fine, les termes suivants peuvent être utilisés : un certificat croisé est appelé un "certificat croisé inter domaine" si les CA sujette et productrice appartiennent à des domaines administratifs différents ; autrement, il est appelé un "certificat croisé intra domaine".

   Note 1. La définition de "certificat croisé" ci-dessus s'aligne sur le terme défini de "certificat de CA" dans X.509. Noter que ce terme ne doit pas être confondu avec le type d'attribut X.500 "cACertificate", qui n' pas de rapport.

   Note 2. Dans de nombreux environnements, le terme "certificat croisé", sauf mieux qualifié, sera compris comme synonyme de "certificat croisé inter-domaine" comme défini ci-dessus.

   Note 3. La production de certificats croisés peut être mutuelle, mais pas nécessairement ; c'est-à-dire que deux CA peuvent produire des certificats croisés l'une pour l'autre.

   6. Mise à jour de certificat croisé : Similaire à une mise à jour normale de certificat, mais impliquant un certificat croisé.

4. Opérations de découverte de certificat/CRL : certaines opérations de gestion de PKI résultent en la publication de certificats ou de CRL :
   1. Publication de certificat : après avoir passé l'épreuve de la production d'un certificat, il faut des moyens de le publier. Les "moyens" définis dans PKIX PEUVENT impliquer les messages spécifiés aux paragraphes 5.3.13 à 5.3.16, ou PEUVENT impliquer d'autres méthodes (LDAP, par exemple) comme décrit dans les [RFC2559], [RFC2585] (les documents de "protocoles opérationnels" de la série des spécifications PKIX).
   2. Publication de CRL : comme pour une publication de certificat.

5. Opérations de récupération : certaines opérations de gestion de PKI sont utilisées lorsque une entité d'extrémité a "perdu" son PSE :
   1. Récupération de paire de clés : en option, les matériels de clé de client utilisateur (par exemple, la clé privée d'un utilisateur utilisée pour le déchiffrement) PEUVENT être sauvegardés par une CA, une RA, ou un système de sauvegarde de clé associé à la CA ou RA. Si une entité a besoin de récupérer ces matériels de clé sauvegardés (par exemple, par suite de l'oubli d'un mot de passe ou la perte du fichier d'une chaîne de clé) un échange de protocole peut être nécessaire pour prendre en charge une telle récupération.

6. Opérations de révocation : certaines opérations de PKI résultent en la création de nouvelles entrées de CRL et/ou de nouvelle CRL :
   1. Demande de révocation : une personne autorisée informe une CA d'une situation anormale exigeant une révocation de certificat.

7. Opérations de PSE : bien que la définition des opérations de PSE (par exemple, déplacement d'un PSE, changement d'un PIN, etc.) sorte du domaine d'application de la présente spécification, on définit un message PKI (CertRepMessage) qui peut former la base de telles opérations.

Noter que les protocoles en ligne ne sont pas la seule façon de mettre en œuvre les opérations ci-dessus. Pour toutes les opérations, il y a des méthodes hors ligne qui arrivent au même résultat, et la présente spécification ne rend pas obligatoire les protocoles en ligne. Par exemple, quand des jetons matériels sont utilisés, beaucoup d'opérations PEUVENT être réalisées au titre de la livraison du jeton physique.

Les sections qui suivent définissent un ensemble standard de messages qui prennent en charge les opérations ci-dessus. Les protocoles de transport pour porter ces échanges dans différents environnements (fondés sur le fichier, en ligne, par messagerie électronique, et sur la Toile mondiale) sortent du domaine d'application du présent document et sont spécifiés séparément.

4. Hypothèses et restrictions

4.1 Initialisation d'entité d'extrémité

La première étape du traitement des entités de gestion de PKI par une entité d'extrémité est de demander des informations sur les fonctions de PKI prises en charge et pour acquérir en toute sécurité une copie de la ou des clés publiques de la CA racine pertinente .

4.2. Enregistrement/certification initial

Il y a plusieurs schémas qui peuvent être utilisés pour réaliser l'enregistrement et la certification initiaux des entités d'extrémité. Aucune méthode ne convient pour toutes les situations du fait de la gamme des politiques qu'une CA peut mettre en œuvre et de la diversité des types d'entité d'extrémité qui peuvent se présenter.

Cependant, on peut classer les schémas d'enregistrement/certification initiaux qui sont pris en charge par la présente spécification. Noter que le mot "initial", ci-dessus, est crucial : on traite de la situation où l'entité d'extrémité en question n'a eu aucun contact antérieur avec la PKI. Lorsque l'entité d'extrémité possède déjà des clé certifiées, d'autres simplifications ou solutions de remplacement sont possibles.

Ayant classé les schémas qui sont pris en charge par la présente spécification, on peut alors en spécifier certains comme obligatoires et d'autres comme facultatifs. Le but est que les schémas obligatoires couvrent un nombre suffisant des cas qui se présentent en utilisation réelle, tandis que les schémas facultatifs sont disponibles pour les cas particuliers qui surviennent moins fréquemment. De cette façon, on réalise un équilibre entre flexibilité et facilité de mise en œuvre.

On décrit maintenant la classification des schémas d'enregistrement/certification initial.

4.2.1 Critères utilisés

4.2.1.1 Initiation d'enregistrement/certification

En termes de production de messages PKI, on peut regarder l'initiation des échanges initiaux d'enregistrement/certification comme arrivant chaque fois que le premier message PKI relatif à l'entité d'extrémité est produit. Noter que l'initiation réelle de la procédure d'enregistrement/certification peut se produire ailleurs (par exemple, un service du personnel peut téléphoner à l'opérateur de RA).

Les localisations possibles sont à l'entité d'extrémité, à une RA, ou une CA.

4.2.1.2. Authentification d'origine de message d'entité d'extrémité

Les messages en ligne produits par l'entité d'extrémité qui exige un certificat peuvent être authentifiés ou non. L'exigence est ici d'authentifier l'origine de tout message provenant de l'entité d'extrémité à la PKI (CA/RA).

Dans cette spécification, une telle authentification est réalisée par la PKI (CA/RA) fournissant à l'entité d'extrémité une valeur secrète (clé authentification initiale) et une valeur de référence (utilisée pour identifier la valeur secrète) via des moyens hors bande. La clé initiale d'authentification peut alors être utilisée pour protéger les messages de PKI pertinents.

Donc, on peut classer le schéma initial d'enregistrement/certification selon que les messages de PKI en ligne de l'entité d'extrémité à la PKI sont authentifiés ou non.

Note 1 : On ne discute pas ici de authentification des messages de la PKI vers l'entité d'extrémité, car c'est toujours EXIGÉ. Dans tous les cas, cela peut se faire simplement une fois que la clé publique de la CA racine a été installée dans l'équipement de l'entité d'extrémité ou elle peut se fonder sur la clé d'authentification initiale.

Note 2 : Une procédure d'enregistrement/certification initiale peut être sûre quand les messages provenant de l'entité d'extrémité sont authentifiés via des moyens hors-bande (par exemple, une visite ultérieure).

4.2.1.3 Localisation de la génération de clé

Dans la présente spécification, la "génération de clé" est considérée comme se produisant chaque fois que le composant public ou privé d'une paire de clés apparaît pour la première fois dans un message de PKI. Noter que cela n'empêche pas un service centralisé de génération de clé ; les paires de clés réelles PEUVENT avoir été générées ailleurs et transportées à l'entité d'extrémité, RA, ou CA en utilisant un protocole (propriétaire ou normalisé) de demande/réponse de génération de clé (hors du domaine d'application de cette spécification).

Donc, il y a trois possibilités pour la localisation d'une "génération de clé" : l'entité d'extrémité, une RA, ou une CA.

4.2.1.4 Confirmation de la réussite de la certification

Suite à la création d'un certificat initial pour une entité d'extrémité, des assurances supplémentaires peuvent être obtenues en faisant que l'entité d'extrémité confirme explicitement la réception réussie du message contenant le certificat (ou indiquant sa création). Naturellement, ce message de confirmation doit être protégé (sur la base de la clé initiale d'authentification ou par d'autres moyens).

Cela donne deux possibilités supplémentaires : confirmé ou non.

4.2.2 Schémas obligatoires

Les critères ci-dessus permettent un grand nombre de schémas initiaux d'enregistrement/certification. La présente spécification rend obligatoire que les équipements conformes de CA, de RA, et d'EE prennent en charge le second schéma de la liste du paragraphe 4.2.2.2. Toute entité PEUT de plus prendre en charge d'autres schémas, si elle le désire.

4.2.2.1 Schéma centralisé

Selon les termes de la classification donnée ci-dessus, ce schéma est, d'une certaine façon, le plus simples possible, où :
- o l'initialisation se produit à la CA qui certifie ;
- o aucune authentification en ligne de message n'est exigée ;
- o la "génération de clé" se produit à la CA qui certifie (voir au paragraphe 4.2.1.3) ;
- o aucun message de confirmation n'est nécessaire.

En termes de flux de messages, ce schéma signifie que le seul message requis est envoyé de la CA à l'entité d'extrémité. Le message doit contenir le PSE entier pour l'entité d'extrémité. Des moyens hors bande doivent être fournis pour permettre à l'entité d'extrémité d'authentifier le message reçu et de déchiffrer toutes les valeurs chiffrées.

4.2.2.2 Schéma authentifié de base

Selon les termes de la classification donnée ci-dessus, ce schéma est celui où :
- o l'initialisation se produit à l'entité d'extrémité ;
- o le message d'authentification est EXIGÉ ;
- o la "génération de clé" se produit à l'entité d'extrémité (voir le paragraphe 4.2.1.3) ;
- o un message de confirmation est EXIGÉ.

En termes de flux de message, le schéma de base authentifié est le suivant :

Entité d'extrémité **RA/CA**

distribution hors bande de la clé d'authentification initiale (IAK)
et de la valeur de référence (RA/CA -> EE)

Génération de clé
Création de demande de certification
Protection de demande avec IAK

-->>-- demande de certification --->>--

vérifie la demande
Traite la requête
crée la réponse

--<<-- réponse de certification --<<--

Traite la réponse

Crée la confirmation

-->>-- message de confirmation de certificat --->>--

vérifie la confirmation
crée la réponse

--<<-- accusé de réception de confirmation (facultatif) --<<--

Traite la réponse

(Lorsque la vérification du message de confirmation de certificat échoue, le RA/CA DOIT révoquer le certificat nouvellement produit si il a été publié ou rendu disponible de quelque façon que ce soit.

4.3 Preuve de possession (POP) de clé privée

Afin de prévenir certaines attaques et de permettre à une CA/RA de vérifier correctement la validité du lien entre une entité d'extrémité et une paire de clés, les opérations de gestion de PKI spécifiées ici rendent possible à une entité d'extrémité de prouver qu'elle est en possession de (c'est-à-dire, est capable d'utiliser) la clé privée correspondant à la clé publique pour laquelle un certificat est demandé. Une CA/RA est libre de choisir comment appliquer la POP (par exemple, des moyens de procédure hors bande ou des messages PKIX-CMP dans la bande) dans ses échanges de certification (c'est-à-dire, ce peut être une question de politique). Cependant, il est EXIGÉ que les CA/RA appliquent la POP avec des moyens parce qu'il y a actuellement de nombreux protocoles opérationnels non PKIX en usage (les divers protocoles de messagerie électronique en sont un exemple) qui ne vérifie pas explicitement le lien entre l'entité d'extrémité et la clé privée. Jusqu'à ce qu'il existe des protocoles opérationnels qui vérifient le lien (pour la signature, le chiffrement, et les paires de clés d'accord de clé) et qu'ils soient répandus partout, ce lien ne peut être supposé avoir été vérifié par la CA/RA. Donc, si le lien n'est pas vérifié par la CA/RA, les certificats dans l'infrastructure de clé publique de l'Internet vont finir par perdre de leur signification.

La POP est réalisée de différentes façons qui dépendent du type de clé pour lequel un certificat est demandé. Si une clé peut être utilisée à de multiples fins (par exemple, une clé RSA) toute méthode appropriée PEUT être utilisée (par exemple, une clé qui peut être utilisée pour signer, aussi bien qu'à d'autres fins, NE DEVRAIT PAS être envoyée à la CA/RA afin de prouver la possession).

La présente spécification permet explicitement les cas où une entité d'extrémité fournit la preuve pertinente à une RA et où la RA atteste ensuite à la CA que la preuve requise a été reçue (et validée !). Par exemple, une entité d'extrémité qui souhaite avoir une clé de signature certifiée pourrait envoyer la signature appropriée à la RA, qui notifie alors simplement à la CA pertinente que entité d'extrémité a fourni la preuve requise. Bien sûr, une telle situation peut être interdite par certaines politiques (par exemple, les CA peuvent être les seules entités à qui il est permis de vérifier la POP durant la certification).

4.3.1 Clés de signature

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

4.3.2 Clés de chiffrement

Pour les clés de chiffrement, l'entité d'extrémité peut fournit la clé privée à la CA/RA, ou peut être obligée de déchiffrer une valeur afin de prouver la possession de la clé privée (voir au paragraphe 5.2.8). Le déchiffrement d'une valeur peut être réalisé directement ou indirectement.

La méthode directe est que la RA/CA produise un défi aléatoire auquel est exigée une réponse immédiate de la part de l'EE.

La méthode indirecte est de produire un certificat qui est chiffré pour l'entité d'extrémité (et l'entité d'extrémité doit démontrer sa capacité à déchiffrer ce certificat dans le message de confirmation). Cela permet à une CA de produire un certificat sous une forme qui ne peut être utilisée que par l'entité d'extrémité prévue.

La présente spécification encourage l'usage de la méthode indirecte parce que elle n'exige pas l'envoi de messages supplémentaires (c'est-à-dire, la preuve peut être démontrée en utilisant le triplet de messages {demande, réponse, confirmation}).

4.3.3 Clés d'accord de clé

Pour les clés d'accord de clé, l'entité d'extrémité et l'entité de gestion de PKI (c'est-à-dire, la CA ou RA) doivent établir une clé secrète partagée afin de prouver que l'entité d'extrémité est en possession de la clé privée.

Noter que cela n'a pas besoin d'imposer de restrictions aux clés qui peuvent être certifiées par une certaine CA. En particulier, pour les clés Diffie-Hellman, l'entité d'extrémité peut librement choisir ses paramètres d'algorithme pourvu que la CA puisse générer une paire de clés à court terme (ou à usage unique) avec les paramètres appropriés quand nécessaire.

4.4 Mise à jour de clé de CA racine

Cette discussion ne s'applique qu'aux CA qui sont directement de confiance pour certaines entités d'extrémité. Les CA auto signées DEVRONT être considérées comme directement de confiance pour les CA. Reconnaître si une CA non auto signée est supposée être directement de confiance pour certaines entités d'extrémité est une affaire de politique de CA et sort donc du domaine d'application du présent document.

La base de la procédure qu'on décrit ici est que la CA protège sa nouvelle clé publique en utilisant ses clés privées précédentes et vice versa. Donc, quand une CA met à jour sa paire de clés elle doit générer des valeurs supplémentaires d'attribut cACertificate si les certificats sont disponibles en utilisant un répertoire X.500 (pour un total de quatre : VieilleAvecVieille, VieilleAvecNouvelle, NouvelleAvecVieille, et NouvelleAvecNouvelle).

Quand une CA change sa paire de clés, les entités qui ont acquis la vieille clé publique de la CA via des moyens "hors bande" sont le plus affectés. Ce sont ces entités d'extrémité qui vont avoir besoin d'accéder à la nouvelle clé publique de CA protégée avec la vieille clé privée de CA. Cependant, elles n'auront besoin de cela que pour un temps limité (jusqu'à ce qu'elles aient acquis la nouvelle clé publique de CA via le mécanisme "hors bande"). Cela sera normalement facilement réalisé quand les certificats de ces entités d'extrémité arrivent à expiration.

La structure de données utilisée pour protéger la nouvelle clé publique de CA et la vieille est dans un certificat standard (qui peut aussi contenir des extensions). Aucune nouvelle structure de données n'est requise.

Note 1. Ce schéma n'utilise aucune des extensions de X.509 v3 car il doit être capable de fonctionner même pour les certificats de version 1. La présence de l'extension KeyIdentifier améliorerait l'efficacité.

Note 2. Bien que le schéma puisse être généralisé pour couvrir les cas où la CA met à jour sa paire de clés plus d'une fois durant la période de validité des certificats d'une de ses entités d'extrémité, Cette généralisation semble d'une valeur douteuse. Ne pas avoir cette généralisation signifie simplement que les périodes de validité des certificats produits avec la vieille paire de clés de CA ne peuvent pas excéder la fin de la période de validité de VieilleAvecNouvelle.

Note 3. Ce schéma assure que les entités d'extrémité vont acquérir la nouvelle clé publique de CA, au plus tard à l'expiration du dernier certificat en leur possession qui a été signé avec la vieille clé privée de la CA (via les moyens "hors bande"). Les opérations de mise à jour de certificat et/ou de clé survenant à d'autres moments n'exigent pas nécessairement cela (selon l'équipement de l'entité d'extrémité).

4.4.1 Actions de l'opérateur de CA

Pour changer la clé de la CA, l'opérateur de CA fait ce qui suit :

1. Générer une nouvelle paire de clés ;
2. Créer un certificat contenant la vieille clé publique de CA signée avec la nouvelle clé privée (le certificat "vieux avec nouveau") ;
3. Créer un certificat contenant la nouvelle clé publique de CA signée avec la vieille clé privée (le certificat "nouveau avec vieux") ;
4. Créer un certificat contenant la nouvelle clé publique de CA signée avec la nouvelle clé privée (le certificat "nouveau avec nouveau") ;
5. Publier ces nouveaux certificats via le répertoire et/ou autre moyen (peut être en utilisant un message CAKeyUpdAnn) ;
6. Exporter la nouvelle clé publique de CA afin que les entités d'extrémité puissent l'acquérir en utilisant le mécanisme "hors bande" (si nécessaire).

La vieille clé privée de CA n'est alors plus requise. Cependant, la vieille clé publique de CA va rester en usage pendant quelques temps. La vieille clé publique de CA n'est plus requise (sauf pour une non répudiation) quand toutes les entités d'extrémité de cette CA ont en toute sécurité acquis la nouvelle clé publique de CA.

Le certificat "vieux avec nouveau" doit avoir une période de validité qui commence au moment de la génération de la vieille paire de clés et se termine à la date d'expiration de la vieille clé publique.

Le certificat "nouveau avec vieux" doit avoir une période de validité commençant au moment de la génération de la nouvelle paire de clés et se terminant au moment où toutes les entités d'extrémité de cette CA vont posséder en toute sécurité la nouvelle clé publique de CA (au plus tard, à la date d'expiration de la vieille clé publique.

Le certificat "nouveau avec nouveau" doit avoir une période de validité qui commence au moment de la génération de la nouvelle paire de clés et qui se termine au moment, ou avant le moment auquel la CA va ensuite mettre à jour sa paire de clés.

4.4.2 Vérification des certificats

Normalement quand il vérifie une signature, le vérificateur vérifie (entre autres choses) le certificat qui contient la clé publique du signataire. Cependant, une fois qu'une CA est autorisée à mettre à jour sa clé, il y a toute une gamme de nouvelles possibilités. Elles sont montrées dans le tableau ci-dessous.

	Le répertoire contient les clés publiques nouvelles et vieilles		Le répertoire contient seulement la vieille clé publique (à cause, par exemple, du délai de publication)	
	PSE contient la nouvelle clé publique	PSE contient la vieille clé publique	PSE contient la nouvelle clé publique	PSE contient la vieille clé publique
Certificat du signataire protégé avec la nouvelle clé publique	Cas 1 : c'est le cas standard où le vérificateur peut directement vérifier le certificat sans utiliser le répertoire	Cas 3 : dans ce cas le vérificateur doit accéder au répertoire afin d'obtenir la valeur de la nouvelle clé publique	Cas 5 : bien que l'opérateur de CA n'ait pas mis à jour le répertoire, le vérificateur peut vérifier directement le certificat - c'est donc le même que le cas 1.	Cas 7 : Dans ce cas, l'opérateur de CA n'a pas mis à jour le répertoire et la vérification va donc échouer.
Certificat du signataire protégé avec la vieille clé publique.	Cas 2 : le vérificateur doit accéder au répertoire pour obtenir la valeur de la vieille clé publique.	Cas 4 : le vérificateur peut vérifier directement le certificat sans utiliser le répertoire.	Cas 6 : le vérificateur pense que c'est la situation du cas 2 et va accéder au répertoire ; cependant, la vérification va échouer.	Cas 8 : bien que l'opérateur de CA n'ait pas mis à jour le répertoire, le vérificateur peut vérifier le certificat directement - c'est donc comme le cas 4.

4.4.2.1 Vérification dans les cas 1, 4, 5, et 8

Dans ces cas, le vérificateur a une copie locale de la clé publique de CA qui peut être utilisée pour vérifier directement le certificat. C'est comme la situation où il n'y a pas eu de changement de clé.

Noter que le cas 8 se produira entre le moment où l'opérateur de CA a généré la nouvelle paire de clés et le moment où l'opérateur de CA mémorise les attributs mis à jour dans le répertoire. Le cas 5 ne peut se produire que si l'opérateur de CA a produit les deux certificats du signataire et du vérificateur dans ce "créneau" (l'opérateur de CA DEVRAIT éviter cela comme conduisant aux cas d'échec décrits ci-dessous).

4.4.2.2 Vérification dans le cas 2

Dans le cas 2, le vérificateur doit obtenir l'accès à la vieille clé publique de la CA. Le vérificateur fait ce qui suit :

1. Chercher l'attribut caCertificate dans le répertoire et prendre le certificat VieuxAvecNouveau (déterminé sur la base des périodes de validité ; noter que les champs de sujet et de producteur doivent correspondre) ;
2. Vérifier que c'est correct en utilisant la nouvelle clé de CA (que le vérificateur a en local) ;
3. Si c'est correct, vérifier le certificat du signataire en utilisant la vieille clé de CA.

Le cas 2 va arriver quand l'opérateur de CA a produit le certificat du signataire, puis changé la clé, puis produit le certificat du vérificateur ; c'est donc un cas assez typique.

4.4.2.3 Vérification dans le cas 3

Dans le cas 3, le vérificateur doit obtenir l'accès à la nouvelle clé publique de la CA. Le vérificateur fait ce qui suit :

1. Chercher l'attribut CACertificate dans le répertoire et prendre le certificat NouveauAvecVieux (déterminé sur la base des périodes de validité ; noter que les champs de sujet et de producteur doivent correspondre) ;
2. Vérifier que c'est correct en utilisant la vieille clé de CA (que le vérificateur a mémorisée en local) ;
3. Si c'est correct, vérifier le certificat du signataire en utilisant la nouvelle clé de CA.

Le cas 3 va arriver quand l'opérateur de CA a produit le certificat du vérificateur, puis changé la clé, puis produit le certificat du signataire ; c'est donc aussi un cas assez typique.

4.4.2.4 Échec de vérification dans le cas 6

Dans ce cas, la CA a produit le PSE du vérificateur, qui contient la nouvelle clé, sans mettre à jour les attributs du répertoire. Cela signifie que le vérificateur n'a pas de moyen d'obtenir une version digne de confiance de la vieille clé de la CA et donc la vérification échoue.

Noter que l'échec est de la faute de l'opérateur de la CA.

4.4.2.5 Échec de vérification dans le cas 7

Dans ce cas, la CA a produit le certificat du signataire protégé par la nouvelle clé sans mettre à jour les attributs du répertoire. Cela signifie que le vérificateur n'a pas de moyen pour obtenir une version digne de confiance de la nouvelle clé de la CA et donc la vérification échoue.

Noter que l'échec est encore de la faute de l'opérateur de la CA.

4.4.3 Révocation – changement de clé de CA

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

L'analyse des solutions de remplacement est la même que pour la vérification de certificat.

5. Structures des données

Cette section contient les descriptions des structures de données requises pour les messages de gestion de PKI. La Section 6 décrit les contraintes sur leurs valeurs et la séquence des événements pour chacune des diverses opérations de gestion de PKI.

5.1 Message PKI global

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

```
PKIMessage ::= SEQUENCE {
   en-tête       PKIHeader,
   corps          PKIBody,
   protection   [0] PKIProtection  FACULTATIF,
   extraCerts   [1] SEQUENCE TAILLE (1..MAX) DE CMPCertificate  FACULTATIF
}

PKIMessages ::= SEQUENCE TAILLE (1..MAX) DE PKIMessage
```

Le PKIHeader contient les informations qui sont communes à de nombreux messages de PKI.

Le PKIBody contient des informations spécifiques du message.

Le PKIProtection, quand il est utilisé, contient des bits qui protègent le message de PKI.

Le champ extraCerts peut contenir des certificats qui peuvent être utiles au receveur. Par exemple, ce peut être utilisé par une CA ou RA pour présenter à une entité d'extrémité les certificats dont elle a besoin pour vérifier son propre nouveau certificat (si, par exemple, la CA qui a produit le certificat de l'entité d'extrémité n'est pas une CA racine pour l'entité d'extrémité). Noter que ce champ ne contient pas nécessairement un chemin de certification ; le receveur peut devoir trier, choisir, ou autrement traiter les certificats supplémentaires afin de les utiliser.

5.1.1 En-tête de message PKI

Tous les messages de PKI requièrent des informations d'en-tête pour l'identification d'adressage et de transaction. Certaines de ces informations vont aussi être présentes dans une enveloppe spécifique du transport. Cependant, si le message de PKI est protégé, alors ces informations sont aussi protégées (c'est-à-dire, on ne fait pas l'hypothèse d'un transport sûr).

La structure de données suivante est utilisée pour contenir ces informations :

```
PKIHeader ::= SEQUENCE {
   pvno              ENTIER     { cmp1999(1), cmp2000(2) },
   envoyeur          GeneralName,
   receveur           GeneralName,
   heure du message          [0] GeneralizedTime          FACULTATIF,
   Alg. de protection        [1] AlgorithmIdentifier      FACULTATIF,
   ID de clé d'envoyeur      [2] KeyIdentifier            FACULTATIF,
   ID de clé de receveur     [3] KeyIdentifier            FACULTATIF,
   ID de transaction         [4] CHAÎNE D'OCTETS    FACULTATIF,
   Nom occas. d'envoyeur     [5] CHAÎNE D'OCTETS    FACULTATIF,
   Nom occas. de receveur    [6] CHAÎNE D'OCTETS    FACULTATIF,
   Texte libre               [7] PKIFreeText              FACULTATIF,
   Info générales            [8] SEQUENCE TAILLE (1..MAX) DE InfoTypeAndValue     FACULTATIF
}

PKIFreeText ::= SEQUENCE TAILLE (1..MAX) DE  chaîne UTF8
```

Le champ pvno *(numéro de version de protocole)* est fixe (de 2) pour cette version de cette spécification.

Le champ envoyeur contient le nom de l'envoyeur du message de PKI. Ce nom (en conjonction avec l'identifiant de clé d'envoyeur, s'il est fourni) devrait être suffisant pour indiquer la clé à utiliser pour vérifier la protection sur le message. Si rien n'est connu sur l'envoyeur par l'entité qui envoie (par exemple, dans le message de demande d'initialisation, où l'entité d'extrémité peut ne pas connaître son propre nom distinctif (DN), le nom de la boîte aux lettres, l'adresse IP, etc.) le champ "envoyeur" DOIT contenir une valeur "NUL" ; c'est-à-dire que la SÉQUENCE DE relative aux noms distinctifs est de longueur zéro. Dans ce cas, le champ Identifiant de clé d'envoyeur DOIT contenir un identifiant (c'est-à-dire, un numéro de référence) qui indique au receveur les informations appropriées de secret partagé à utiliser pour vérifier le message.

Le champ receveur contient le nom du receveur du message de PKI. Ce nom (en conjonction avec l'identifiant de clé de receveur, s'il est fourni) est utilisable pour vérifier la protection sur le message.

Le champ Algorithme de protection spécifie l'algorithme utilisé pour protéger le message. Si aucun bit de protection n'est fourni (noter que PKIProtection est FACULTATIF) ce champ DOIT alors être omis ; si des bits de protection sont fournis, ce champ DOIT alors être fourni.

Identifiant de clé d'envoyeur et Identifiant de clé de receveur sont utilisables pour indiquer quelles clés ont été utilisées pour protéger le message (recipKID ne va normalement être exigé que lorsque la protection du message utilise des clés Diffie-Hellman (DH)).

Ces champs DOIVENT être utilisés si c'est exigé pour identifier de façon univoque une clé (par exemple, si plus d'une clé est associée à un certain nom d'envoyeur) et DEVRAIENT être omis autrement.

Le champ Identifiant de transaction au sein de l'en-tête de message est à utiliser pour permettre au receveur d'un message de le corréler avec une transaction en cours. C'est nécessaire pour toutes les transactions qui consistent en plus qu'une seule paire de demande/réponse. Pour une transaction qui consiste en une seule paire demande/réponse, la règle est la suivante : un client PEUT remplir le champ Identifiant de transaction de la demande. Si un serveur reçoit une telle demande qui a le champ Identifiant de transaction établi, il DOIT alors régler le champ Identifiant de transaction de la réponse à la même valeur. Si un serveur reçoit une telle demande avec le champ Identifiant de transaction manquant, il PEUT alors établir le champ Identifiant de transaction de la réponse.

Pour les transactions qui consistent en plus que juste une seule paire de demande/réponse, la règle est la suivante : les clients DEVRAIENT générer un identifiant de transaction pour la première demande. Si un serveur reçoit une telle demande qui a le champ Identifiant de transaction établi, il DOIT alors établir le champ Identifiant de transaction de la réponse à la même valeur. Si un serveur reçoit une telle demande où manque le champ Identifiant de transaction, il DOIT alors remplir le champ Identifiant de transaction de la réponse avec un identifiant généré par le serveur. Les demandes et réponses suivantes DOIVENT toutes établir le champ Identifiant de transaction à la valeur ainsi établie. Dans tous les cas où un identifiant de transaction est utilisé, le client NE DOIT PAS avoir plus d'une transaction avec le même identifiant de transaction en cours à un instant donné (avec ce serveur). Les serveurs ont toute liberté pour exiger ou non l'unicité de l'identifiant de transaction, pour autant qu'ils soient capables d'associer correctement les messages à la transaction correspondante. Normalement, cela signifie qu'un serveur va exiger que le couple {client, identifiant de transaction} soit unique, ou même que l'identifiant de transaction seul soit unique, si il ne peut pas distinguer les clients sur la base des informations de niveau transport. Un serveur qui reçoit le premier message d'une transaction (ce qui exige plus qu'une seule

paire de demande/réponse) contenant un identifiant de transaction qui ne lui permet pas de satisfaire la contrainte susmentionnée (normalement parce que l'identifiant de transaction est déjà utilisé) DOIT renvoyer un ErrorMsgContent avec un PKIFailureInfo de transactionIdInUse. Il est RECOMMANDÉ que les clients remplissent le champ Identifiant de transaction avec 128 bits de données (pseudo-) aléatoires pour le début d'une transaction pour réduire la probabilité d'avoir un identifiant de transaction déjà utilisé au serveur.

Les champs senderNonce et recipNonce protègent le message PKI contre les attaques en répétition. Le nom occasionnel d'envoyeur va normalement avoir 128 bits de données (pseudo-)aléatoires générées par l'envoyeur, tandis que le nom occasionnel de receveur est copié du nom occasionnel d'envoyeur du message précédent dans la transaction.

Le champ Heure du message contient l'heure à laquelle l'envoyeur a créé le message. Cela peut être utile pour permettre aux entités d'extrémité de corriger/vérifier leur heure locale pour la cohérence avec l'heure d'un système central.

Le champ Texte libre peut être utilisé pour envoyer des messages lisibles par l'homme au receveur (dans tous les langages désirés). Le premier langage utilisé dans cette séquence indique le langage désiré pour les réponses.

Le champ Informations générales peut être utilisé pour envoyer des données supplémentaires traitables par la machine au receveur. Les extensions d'informations générales suivantes sont définies et PEUVENT être prises en charge.

5.1.1.1 ImplicitConfirm *(confirmation implicite)*

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

```
IDENTIFIANT D'OBJET  implicitConfirm ::= {id-it 13}
      ImplicitConfirmValue ::= NULL
```

Si la CA accepte la demande de l'EE, elle DOIT mettre la même extension dans l'en-tête PKI de la réponse. Si l'EE ne trouve pas l'extension dans la réponse, elle DOIT envoyer la confirmation de certificat.

5.1.1.2 ConfirmWaitTime *(temps d'attente de confirmation)*

Ceci est utilisé par la CA pour informer l'EE du temps pendant lequel elle entend attendre la confirmation de certificat avant de le révoquer et de supprimer la transaction.

```
IDENTIFIANT D'OBJET  confirmWaitTime ::= {id-it 14}
      ConfirmWaitTimeValue ::= GeneralizedTime
```

5.1.2 Corps de message PKI

```
PKIBody ::= CHOIX {
      ir      [0]  CertReqMessages,       -- demande d'initialisation
      ip      [1]  CertRepMessage,        -- réponse d'initialisation
      cr      [2]  CertReqMessages,       -- demande de certification
      cp      [3]  CertRepMessage,        -- réponse de certification
      p10cr   [4]  CertificationRequest,  -- demande de certification PKCS n°10
      popdecc  [5]  POPODecKeyChallContent -- défi pop
      popdecr  [6]  POPODecKeyRespContent, -- réponse pop
      kur     [7]  CertReqMessages,       -- demande de mise à jour de clé
      kup     [8]  CertRepMessage,        -- réponse de mise à jour de clé
      krr     [9]  CertReqMessages,       -- demande de récupération de clé
      krp     [10] KeyRecRepContent,      -- réponse de récupération de clé
      rr      [11] RevReqContent,         -- demande de révocation
      rp      [12] RevRepContent,         -- réponse de révocation
      ccr     [13] CertReqMessages,       -- demande de certification croisée
      ccp     [14] CertRepMessage,        -- réponse de certification croisée
      ckuann  [15] CAKeyUpdAnnContent,    -- annonce de mise à jour de clé de CA
      cann    [16] CertAnnContent,        -- annonce de certificat
      rann    [17] RevAnnContent,         -- annonce de révocation
      crlann  [18] CRLAnnContent,         -- annonce de CRL
      pkiconf [19] PKIConfirmContent,     -- confirmation
      nested  [20] NestedMessageContent,  -- message incorporé
      genm    [21] GenMsgContent,         -- message général
```

```
        genp     [22] GenRepContent,         -- réponse générale
        error    [23] ErrorMsgContent,       -- message d'erreur
        certConf [24] CertConfirmContent,    -- confirmation de certificat
        pollReq  [25] PollReqContent,        -- demande d'interrogation
        pollRep  [26] PollRepContent         -- réponse d'interrogation
        }
```

Les types spécifiques sont décrits au paragraphe 5.3.

5.1.3 Protection du message PKI

Certains messages de PKI vont être protégé en intégrité. (Noter que si un algorithme asymétrique est utilisé pour protéger un message et si le composant public pertinent a déjà été certifié, l'origine du message peut alors aussi être authentifiée. Par ailleurs, si le composant public n'est pas certifié, l'origine du message ne peut alors pas être automatiquement authentifiée, mais peut être authentifiée via des moyens hors bande.)

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

PKIProtection ::= CHAÎNE BINAIRE

L'entrée au calcul de PKIProtection est le codage en DER de la structure de données suivante :

```
ProtectedPart ::= SEQUENCE {
    en-tête   PKIHeader,
    corps     PKIBody
 }
```

Il PEUT y avoir des cas où la chaîne binaire PKIProtection est délibérément non utilisée pour protéger un message (c'est-à-dire, ce champ FACULTATIF est omis) parce que une autre protection, externe à PKIX, sera appliquée à sa place. Un tel choix est explicitement permis dans la présente spécification. Des exemples d'une telle protection externe incluent PKCS n° 7 [PKCS7] et l'encapsulation de multi parties de sécurité [RFC1847] du message de PKI (ou simplement de PKIBody en omettant l'étiquette CHOIX si les informations pertinentes d'en-tête de PKI sont portées en toute sécurité dans le mécanisme externe). Il est noté, cependant, que beaucoup de ces mécanismes externes exigent que l'entité d'extrémité possède déjà un certificat de clé publique, et/ou un nom distinctif unique, et/ou d'autres informations de cette sorte relatives à l'infrastructure. Donc, cela peut n'être pas approprié pour l'enregistrement initial, la récupération de clé, ou tout autre processus avec des caractéristiques "d'amorçage". Pour ces cas, il peut être nécessaire que le paramètre PKIProtection soit utilisé. À l'avenir, si/quand des mécanismes externes sont modifiés pour s'accommoder des scénarios d'amorçage, l'utilisation de PKIProtection pourrait devenir rare ou disparaître.

Selon les circonstances, les bits de PKIProtection peuvent contenir un code d'authentification de message (MAC, *Message Authentication Code*) ou une signature. Seuls les cas suivants peuvent se produire :

5.1.3.1 Informations secrètes partagées

Dans ce cas, l'envoyeur et le receveur partagent des informations secrètes (établies par des moyens hors bande ou à partir d'une opération antérieure de gestion de PKI). PKIProtection va contenir une valeur de MAC et l'algorithme de protection sera le suivant (voir aussi l'Appendice D.2) :

```
IDENTIFIANT D'OBJET  id-PasswordBasedMac ::= {1 2 840 113533 7 66 13}
    PBMParameter ::= SEQUENCE {
      sel               CHAÎNE D'OCTETS,
      owf                 AlgorithmIdentifier,
      compte d'itérations      ENTIER,
      mac                 AlgorithmIdentifier
    }
```

Dans l'algorithme de protection ci-dessus, la valeur de sel est ajoutée à l'entrée de secret partagé. La OWF *(one-way function, fonction unidirectionnelle)* est ensuite appliquée iterationCount fois, où le secret salé est l'entrée à la première itération et, pour chaque itération successive, l'entrée est réglée à être le résultat de l'itération précédente. Le résultat de l'itération finale (appelée "BASEKEY" pour faciliter la référence, d'une taille de "H") est ce qui est utilisé pour former la clé symétrique. Si l'algorithme de MAC exige une clé de K bis et si $K \leq H$, les K bits de plus fort poids de BASEKEY sont alors utilisés. Si $K > H$, alors tout BASEKEY est utilisé comme les H bits de poids fort de la clé, OWF("1" || BASEKEY)

est utilisé pour les H bits de poids fort suivants de la clé, OWF("2" || BASEKEY) est utilisé pour les H bits de poids fort suivants de la clé, et ainsi de suite, jusqu'à ce que tous les K bits aient été déduits. [Ici "N" est le codage d'octet ASCII du nombre N et "||" représente l'enchaînement.]

Note : il est RECOMMANDÉ que les champs de PBMParameter restent constants à travers les messages d'une même transaction (par exemple, ir/ip/certConf/pkiConf) afin de réduire les frais généraux associés au calcul du PasswordBasedMac *(MAC fondé sur le mot de passe)*.

5.1.3.2 Paires de clés DH

Lorsque envoyeur et receveur possèdent des certificats Diffie-Hellman avec des paramètres DH compatibles, afin de protéger le message, l'entité d'extrémité doit générer une clé symétrique sur la base de sa valeur de clé DH privée et de la clé publique DH du receveur du message de PKI. PKIProtection va contenir une valeur de MAC chiffrée avec cette clé symétrique déduite et le protectionAlg sera le suivant :

IDENTIFIANT D'OBJET id-DHBasedMac ::= {1 2 840 113533 7 66 30}

```
DHBMParameter ::= SEQUENCE {
   owf   AlgorithmIdentifier,      -- identifiant d'algorithme pour une fonction unidirectionnelle, SHA-1 recommandé)
   mac   AlgorithmIdentifier       -- AlgId du MAC (par exemple, DES-MAC, Triple-DES-MAC [PKCS11],
  }                                -- ou HMAC [RFC2104], [RFC2202])
```

Dans l'algorithme de protection ci-dessus, OWF est appliqué au résultat du calcul Diffie-Hellman. Le résultat de la fonction unidirectionnelle (appelé "BASEKEY" pour faciliter la référence, d'une taille de "H") est ce qui est utilisé pour former la clé symétrique. Si l'algorithme de MAC exige une clé de K bits et si K ≤ H, K bits de poids fort de BASEKEY sont utilisés. Si K > H, tout BASEKEY est alors utilisé pour les H bits de poids fort de la clé, OWF("1" || BASEKEY) est utilisé pour les H bits de poids fort suivants de la clé, OWF("2" || BASEKEY) est utilisé pour les H bits de poids fort suivants de la clé, et ainsi de suite, jusqu'à ce que tous les K bits aient été déduits. [Ici "N" est le codage ASCII des octets du nombre N et "||" représente l'enchaînement.]

5.1.3.3 Signature

Dans ce cas, l'envoyeur possède une paire de clés de signature et signe simplement le message de PKI. PKIProtection va contenir la valeur de signature et le protectionAlg sera un identifiant d'algorithme pour une signature numérique (par exemple, md5WithRSAEncryption ou dsaWithSha-1).

5.1.3.4 Protection multiple

Dans les cas où une entité d'extrémité envoie un message PKI protégé à une RA, la RA PEUT transmettre ce message à une CA, en y attachant sa propre protection (qui PEUT être un MAC ou une signature, selon les informations et les certificats partagés entre la RA et la CA). Cela est accompli en incorporant le message entier envoyé par l'entité d'extrémité au sein d'un nouveau message de PKI. La structure utilisée est la suivante.

NestedMessageContent ::= PKIMessages

(L'utilisation de PKIMessages, une SÉQUENCE DE PKIMessage, laisse la RA mettre en lots les demandes de plusieurs EE dans un seul nouveau message. Par simplicité, tous les messages du lot DOIVENT être du même type (par exemple, ir).) Si la RA souhaite modifier le ou les messages d'une certaine façon (par exemple, ajouter des valeurs de champ particulières ou de nouvelles extensions) elle PEUT alors créer les propres corps PKI qu'elle désire. Le message PKI d'origine provenant de l'EE PEUT être inclus dans le champ generalInfo de l'en-tête PKI (pour s'accommoder, par exemple, des cas dans lesquels la CA souhaite vérifier la POP ou d'autres informations sur le message d'origine de l'EE). Le type d'informations à utiliser dans cette situation est {id-it 15} (voir au paragraphe 5.3.19 la valeur de id-it) et la valeur d'informations est PKIMessages (les contenus DOIVENT être dans le même ordre que les demandes dans PKIBody).

5.2 Structures de données communes

Avant de spécifier les types spécifiques qui peuvent être placés dans un corps PKI, on définit des structures de données qui sont utilisées dans plus d'un cas.

5.2.1 Contenu de certificat demandé

Divers messages de gestion de PKI exigent que le générateur du message indique certains des champs dont la présence est exigée dans un certificat. La structure CertTemplate permet à une entité d'extrémité ou RA de spécifier tout ce qu'elle souhaite sur le certificat qu'elle exige. CertTemplate est identique à un certificat, mais avec tous les champs facultatifs.

Noter que même si le générateur spécifie complètement le contenu du certificat qu'il demande, une CA est libre de modifier les champs au sein du certificat réellement produit. Si le certificat modifié n'est pas acceptable au demandeur, il DOIT renvoyer un message certConf qui soit n'inclut pas ce certificat (via un CertHash) soit inclut ce certificat (via un CertHash) avec un statut de "rejeté". Voir au paragraphe 5.3.18 la définition et l'utilisation de CertHash et du message certConf.

Voir à l'Appendice C et la [RFC4211] la syntaxe de CertTemplate.

5.2.2 Valeurs chiffrées

Lorsque des valeurs chiffrées (restreintes, dans cette spécification, à des clés privées ou des certificats) sont envoyées dans des messages de PKI, la structure de données EncryptedValue est utilisée. Voir dans la [RFC4211] la syntaxe de EncryptedValue.

L'utilisation de cette structure de données exige que le créateur et le receveur prévu soient capables, respectivement, de chiffrer et déchiffrer. Normalement, cela va signifier que l'envoyeur et le receveur ont, ou sont capables de générer, une clé secrète partagée.

Si le receveur du message de PKI possède déjà une clé privée utilisable pour le déchiffrement, le champ encSymmKey PEUT alors contenir une clé de session chiffrée utilisant la clé publique du receveur.

5.2.3 Informations de codes d'état et d'échec pour les messages PKI

Tous les messages de réponse vont inclure des informations d'état. Les valeurs suivantes sont définies.

```
PKIStatus ::= ENTIER {
    accepté                          (0),
    accordé avec modifications       (1),
    rejet                            (2),
    attente                          (3),
    avertissement de révocation      (4),
    notification de révocation       (5),
    avertissement de mise à jour de clé (6)
 }
```

Celui qui répond peut utiliser la syntaxe suivante pour donner plus d'informations sur les cas d'échec.

```
PKIFailureInfo ::= CHAÎNE BINAIRE {
    mauvais algorithme               (0),
    échec de vérification de message (1),
    mauvaise demande                 (2),
    mauvaise heure                   (3),
    mauvais identifiant de certificat (4),
    mauvais format de données        (5),
    mauvaise autorité                (6),
    données incorrectes              (7),
    horodatage absent                (8),
    mauvais POP                      (9),
    certificat révoqué               (10),
    certificat confirmé              (11),
    mauvaise intégrité               (12),
    mauvais nom occasionnel de receveur (13),
    heure non disponible             (14),
    politique non acceptée           (15),
    extension non acceptée           (16),
    information supplémentaire non disponible (17),
    mauvais nom occasionnel d'envoyeur (18),
    mauvais gabarit de certificat    (19),
```

```
    signataire non de confiance          (20),
    identifiant de transaction utilisé   (21),
    version non acceptée                 (22),
    non autorisé                         (23),
    système indisponible                 (24),
    défaillance du système               (25),
    certificat dupliqué demandé          (26)
  }

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

5.2.4 Identification de certificat

Afin d'identifier des certificats particuliers, la structure de données CertId est utilisée. Voir la syntaxe de CertId dans la [RFC4211].

5.2.5 Clé publique de CA racine hors bande

Chaque CA racine doit être capable de publier sa clé publique actuelle via des moyens "hors bande" (OOB, *Out Of Band*). Bien que de tels mécanismes sortent du domaine d'application du présent document, on définit les structures de données qui peuvent prendre en charge de tels mécanismes.

Deux méthodes sont généralement disponibles : soit la CA publie directement son certificat auto signé, soit cette information est disponible via le répertoire (ou équivalent) et la CA publie un hachage de cette valeur pour permettre la vérification de son intégrité avant usage.

OOBCert ::= Certificat

Les champs au sein de ce certificat sont restreints à ce qui suit :

- o Le certificat DOIT être auto signé (c'est-à-dire, la signature doit être vérifiable en utilisant le champ SubjectPublicKeyInfo) ;
- o Les champs subject et issuer DOIVENT être identiques ;
- o Si le champ subject est NULL, les deux extensions subjectAltNames et issuerAltNames DOIVENT alors être présentes et avoir exactement la même valeur ;
- o Les valeurs de toutes les autres extensions doivent convenir à un certificat auto signé (par exemple, les identifiants de clé pour subject et issuer doivent être les mêmes).

```
OOBCertHash ::= SEQUENCE {
     hashAlg     [0] AlgorithmIdentifier     FACULTATIF,
     certId      [1] CertId                  FACULTATIF,
     hashVal      CHAÎNE BINAIRE
  }
```

L'intention de la valeur de hachage est que tous ceux qui ont reçu de façon sûre la valeur de hachage (via le moyen hors bande) peuvent vérifier un certificat auto signé pour cette CA.

5.2.6. Options d'archivage

Les demandeurs peuvent indiquer qu'ils souhaitent que PKI archive une valeur de clé privée en utilisant la structure PKIArchiveOptions. Voir dans la [RFC4211] la syntaxe de PKIArchiveOptions.

5.2.7 Information de publication

Les demandeurs peuvent indiquer qu'ils souhaitent que PKI publie un certificat en utilisant la structure PKIPublicationInfo. Voir dans la [RFC4211] la syntaxe de PKIPublicationInfo.

5.2.8 Structures de preuve de possession

Si la demande de certification est pour une paire de clés de signature (c'est-à-dire, une demande pour un certificat de vérification) la preuve de possession de la clé privé de signature est démontrée par l'utilisation de la structure POPOSigningKey. Voir à l'Appendice C et à la [RFC4211] la syntaxe de POPOSigningKey, mais noter que POPOSigningKeyInput a les stipulations de sémantique suivantes dans la présente spécification.

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

Par ailleurs, si la demande de certification est pour une paire de clés de chiffrement (c'est-à-dire, une demande d'un certificat de chiffrement) la preuve de possession de la clé privée de déchiffrement peut être apportée de trois façons.

5.2.8.1 Inclusion de la clé privée

Par l'inclusion de la clé privée (chiffrée) dans le CertRequest (dans le champ thisMessage de POPOPrivKey (voir à l'Appendice C) ou dans la structure de contrôle PKIArchiveOptions, si l'archivage de la clé privée est aussi désiré ou non).

5.2.8.2 Méthode indirecte

En faisant que la CA retourne non le certificat, mais un certificat chiffré (c'est-à-dire, le certificat chiffré sous une clé symétrique générée de façon aléatoire, et la clé symétrique chiffrée avec la clé publique pour laquelle la demande de certification est faite) -- c'est la méthode "indirecte" mentionnée précédemment au paragraphe 4.3.2. L'entité d'extrémité prouve sa connaissance de la clé de déchiffrement privée à la CA en fournissant le hachage de certificat (CertHash) correct pour ce certificat dans le message certConf. Cela démontre la POP parce que l'EE ne peut calculer le CertHash correct que si elle est capable de récupérer le certificat, et elle ne peut récupérer le certificat que si elle est capable de déchiffrer la clé symétrique en utilisant la clé privée requise. Il est clair, pour que ceci fonctionne que la CA NE DOIT PAS publier le certificat tant que le message certConf n'est pas arrivé (quand certHash doit être utilisé pour démontrer la POP). Voir les détails au paragraphe 5.3.18.

5.2.8.3 Protocole de mise au défi/réponse

En engageant l'entité d'extrémité dans un protocole de mise au défi-réponse (en utilisant les messages POPODecKeyChall et POPODecKeyResp ; voir ci-dessous) entre CertReqMessages et CertRepMessage -- c'est la méthode "directe" mentionnée au paragraphe 4.3.2. (Cette méthode va normalement être utilisée dans un environnement dans lequel une RA vérifie la POP et ensuite fait une demande de certification à la CA au nom de l'entité d'extrémité. Dans ce scénario, la CA fait confiance à la RA pour avoir fait la POP correctement avant de demander un certificat pour l'entité d'extrémité.) Le protocole complet ressemble alors à ce qui suit (noter que req' n'encapsule pas nécessairement req comme un message incorporé) :

```
 EE          RA          CA
---- req ---->
<--- chall ---
---- resp --->
                ---- req' --->
               <--- rep -----
               ---- conf --->
               <--- ack -----
<--- rep -----
---- conf --->
<--- ack -----
```

Ce protocole est évidemment plus long que le triple échange donné au choix (2) ci-dessus, mais permet à une autorité d'enregistrement locale d'être impliquée et il a comme propriété que le certificat lui-même n'est en fait pas créé tant que la preuve de possession n'est pas achevée. Dans certains environnements, un ordre différent des messages ci-dessus peut être requis, comme le suivant (ce peut être déterminé par une politique) :

```
EE          RA          CA
 ---- req ---->
```

```
<--- chall ---
---- resp --->
          ---- req' --->
          <--- rep -----
<--- rep -----
---- conf --->
          ---- conf --->
          <--- ack -----
<--- ack -----
```

Si la demande cert. est pour une paire de clés d'accord de clé (KAK) la POP peut alors utiliser une des trois façons décrites ci-dessus pour des paires de clés encapsulées, avec les changements suivants : (1) le texte entre parenthèses du point 2) est remplacé par "(c'est-à-dire, le certificat chiffré avec la clé symétrique déduite de la KAK privée de la CA et la clé publique pour laquelle la demande de certification a été faite)"; (2) le premier texte entre parenthèses du champ Challenge du "Défi" ci-dessous est remplacé par "(en utilisant PreferredSymmAlg (voir au paragraphe 5.3.19.4 et à l'Appendice E.5) et une clé symétrique déduite de la KAK privée de la CA et la clé publique pour laquelle la demande de certification a été faite)". Autrement, la POP peut utiliser la structure POPOSigningKey donnée dans la [RFC4211] (où le champ alg est DHBasedMAC et le champ signature est le MAC) comme une quatrième solution de remplacement pour démontrer la POP si la CA a déjà un certificat D-H qui est connu de l'EE.

Les messages de défi-réponse pour une preuve de possession d'une clé de déchiffrement privée sont spécifiées comme suit (voir [MvOV97], p.404 pour les détails). Noter que cet échange défi-réponse est associé au message de demande de certification précédant (et aux messages réponse de certification et confirmation) par l'identifiant de transaction utilisé dans l'en-tête PKI et par la protection (adjonction d'un MAC ou signature) appliqué au message de PKI.

```
POPODecKeyChallContent ::= SÉQUENCE DE Challenge
   Challenge ::= SEQUENCE {
      owf             AlgorithmIdentifier  FACULTATIF,
      witness         CHAÎNE D'OCTETS,
      challenge       CHAÎNE D'OCTETS
   }
```

Noter que la taille d'aléa doit être appropriée pour le chiffrement sous la clé publique du demandeur. Étant donné que "int" ne va normalement pas faire plus de 64 bits, cela laisse bien plus de 100 octets pour le champ "envoyeur" quand le module fait 1024 bits. Si, dans certains environnements, les noms sont si longs qu'ils ne puissent tenir (par exemple, des noms distinctifs très longs) toute portion qui tient devrait alors être utilisée (pour autant qu'elle inclue au moins le nom courant, et pour autant que le receveur soit capable de s'accommoder de l'abréviation).

```
POPODecKeyRespContent ::= SÉQUENCE DE ENTIER
```

5.2.8.4 Résumé des options POP

Le texte de ce paragraphe donne plusieurs options quant aux techniques de preuve de possession. En utilisant "SK" pour "clé de signature", "EK" pour "clé de chiffrement", et "KAK" pour "clé d'accord de clé", les techniques peuvent être résumées comme suit :

RAVerified ; *(autorité d'enregistrement vérifiée)*
SKPOP ; *(preuve de possession de la clé de signature)*
EKPOPThisMessage ; *(preuve de possession de la clé de chiffrement de ce message)*
KAKPOPThisMessage ; *(preuve de possession de la clé d'accord de clé de ce message)*
KAKPOPThisMessageDHMAC ; *(MAC D-H de la preuve de possession de la clé d'accord de clé de ce message)*
EKPOPEncryptedCert ; *(certificat chiffré de la preuve de possession de la clé de chiffrement)*
KAKPOPEncryptedCert ; *(certificat chiffré de la preuve de possession de la clé d'accord de clé)*
EKPOPChallengeResp ; *(réponse au défi de preuve de possession de clé de chiffrement)*
KAKPOPChallengeResp ; *(réponse au défi de preuve de possession de clé d'accord de clé)*.

Avec cette grille d'options, il est naturel de demander comment une entité d'extrémité peut savoir ce qui est accepté par la CA/RA (c'est-à-dire, quelles options on peut utiliser quand on demande des certificats). Les lignes directrices suivantes devraient préciser cette situation pour les mises en œuvre d'entités d'extrémité.

RAVerified. Ce n'est pas une décision de l'EE ; la RA utilise cela seulement si elle a vérifié la POP avant de transmettre la demande à la CA, de sorte qu'il n'est pas possible à l'EE de choisir cette technique.

SKPOP. Si l'EE a une paire de clés de signature, c'est la seule méthode de POP spécifiée à utiliser dans la demande pour un certificat correspondant.

EKPOPThisMessage et KAKPOPThisMessage. Abandonner ou non sa clé privée à la CA/RA est une décision de l'EE. Si l'EE décide de révéler sa clé, ce sont alors les seules méthodes de POP disponibles dans cette spécification pour le faire (et le type de paire de clés va déterminer laquelle de ces deux méthodes utiliser).

KAKPOPThisMessageDHMAC. L'EE peut seulement utiliser cette méthode si (1) la CA a un certificat D-H disponible à cette fin, et (2) l'EE a déjà une copie de ce certificat. Si ces deux conditions sont satisfaites, cette technique est alors clairement prise en charge et peut être utilisée par l'EE, si elle le désire.

EKPOPEncryptedCert, KAKPOPEncryptedCert, EKPOPChallengeResp, KAKPOPChallengeResp. L'EE prend une de ces options (dans le champ subsequentMessage) dans le message de demande, selon la préférence et le type de paire de clés. L'EE ne fait pas la preuve de possession à ce point, elle indique simplement quelle méthode elle veut utiliser. Donc, si la CA/RA répond avec une erreur "mauvaise POP", l'EE peut redemander en utilisant l'autre méthode de POP choisie dans subsequentMessage. Noter cependant que la présente spécification encourage l'utilisation du choix EncryptedCert et de plus dit que le défi-réponse va normalement être utilisé quand une RA est impliquée et fait la vérification de POP. Donc, l'EE devrait être capable de prendre une décision intelligente quant au choix des méthodes de preuve de possession dans le message de demande.

5.3 Structures de données spécifiques de l'opération

5.3.1 Demande d'initialisation

Un message de demande d'initialisation contient comme corps de PKI une structure de données CertReqMessages qui spécifie le ou les certificats demandés. Normalement, SubjectPublicKeyInfo, KeyId, et Validity sont les champs de gabarit qui peuvent être fournis pour chaque certificat demandé (voir les profils de l'Appendice D pour plus d'informations). Ce message est destiné à être utilisé pour les entités lors de la première initialisation dans la PKI. Voir à l'Appendice C et dans la [RFC4211] la syntaxe de CertReqMessages.

5.3.2 Réponse d'initialisation

Un message de réponse d'initialisation contient comme corps de PKI une structure de données CertRepMessage qui a pour chaque certificat demandé un champ PKIStatusInfo, un certificat sujet, et éventuellement une clé privée (normalement chiffrée avec une clé de session, qui est elle-même chiffrée avec la protocolEncrKey). Voir au paragraphe 5.3.4 la syntaxe de CertRepMessage. Noter que si la protection du message de PKI est "informations de secret partagé" (voir au paragraphe 5.1.3) tout certificat transporté dans le champ caPubs peut être directement de confiance comme certificat de CA racine par l'initiateur.

5.3.3 Demande de certification

Un message de demande de certification contient comme corps de PKI une structure de données CertReqMessages qui spécifie les certificats demandés. Ce message est destiné à être utilisé pour les entités de PKI existantes qui souhaitent obtenir des certificats supplémentaires. Voir à l'Appendice C et la [RFC4211] la syntaxe de CertReqMessages.

Autrement, le PKIBody PEUT être un CertificationRequest (cette structure est pleinement spécifiée par la structure ASN.1 CertificationRequest donnée dans la [RFC2986]). Cette structure peut être exigée pour les demandes de certificat pour les paires de clés de signature quand on désire une interopération avec les systèmes traditionnels, mais son utilisation est fortement déconseillée chaque fois qu'elle n'est pas absolument nécessaire.

5.3.4 Réponse de certification

Un message Réponse de certification contient comme corps de PKI une structure de données CertRepMessage, qui a une valeur d'état pour chaque certificat demandé, et facultativement une clé publique de CA, des informations d'échec, un certificat sujet, et une clé privée chiffrée.

```
CertRepMessage ::= SEQUENCE {
   caPubs       [1] SEQUENCE TAILLE (1..MAX) DE CMPCertificate      FACULTATIF,
   response          SÉQUENCE DE CertResponse
}
```

```
CertResponse ::= SEQUENCE {
   certReqId          ENTIER,
   status             PKIStatusInfo,
   certifiedKeyPair   CertifiedKeyPair    FACULTATIF,
   rspInfo       CHAÎNE D'OCTETS       FACULTATIF
-- analogue à la chaîne id-regInfo-utf8Pairs définie pour regInfo dans CertReqMsg [RFC4211]
 }

CertifiedKeyPair ::= SEQUENCE {
   certOrEncCert      CertOrEncCert,
   privateKey       [0] EncryptedValue      FACULTATIF, -- voir dans la [RFC4211] le commentaire sur le codage.
   publicationInfo [1] PKIPublicationInfo  FACULTATIF
}

CertOrEncCert ::= CHOIX {
   certificate        [0] CMPCertificate,
   encryptedCert      [1] EncryptedValue
}
```

Un seul des champs failInfo (dans PKIStatusInfo) et certificate (dans CertifiedKeyPair) peut être présent dans chaque CertResponse (selon l'état). Pour certaines valeurs d'état (par exemple, attente) aucun des champs facultatifs ne sera présent.

Avec un EncryptedCert et la clé de déchiffrement pertinente, le certificat peut être obtenu. L'objet de cela est de permettre à une CA de retourner la valeur d'un certificat, mais avec la contrainte que seul le receveur prévu peut obtenir le certificat réel. L'avantage de cette approche est qu'une CA peut répondre avec un certificat même en l'absence d'une preuve que le demandeur est bien l'entité d'extrémité qui peut utiliser la clé privée pertinente (noter que la preuve n'est pas obtenue tant que le message certConf n'est pas reçu par la CA). Donc, la CA n'aura pas à révoquer ce certificat au cas où quelque chose irait de travers avec la preuve de possession (mais elle PEUT le faire de toutes façons) selon sa politique).

5.3.5 Contenu de demande de mise à jour de clé

Pour les demandes de mise à jour de clé, on utilise la syntaxe de CertReqMessages. Normalement, SubjectPublicKeyInfo, KeyId, et Validity sont les champs du gabarit qui peuvent être fournis pour chaque clé à mettre à jour. Ce message est destiné à être utilisé pour demander des mises à jour aux certificats existants (non révoqués et non expirés) (donc, il est parfois appelé une opération de "mise à jour de certificat"). Une mise à jour est un remplacement de certificat contenant soit une nouvelle clé publique sujette, soit la clé publique sujette actuelle (bien que cette dernière pratique puisse n'être pas appropriée pour certains environnements). Voir à l'Appendice C et la [RFC4211] la syntaxe de CertReqMessages.

5.3.6 Contenu de réponse 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. Voir au paragraphe 5.3.4 la syntaxe de CertRepMessage.

5.3.7 Contenu de demande de récupération de clé

Pour les demandes de récupération de clé, la syntaxe utilisée est identique à celle de la demande d'initialisation CertReqMessages. Normalement, SubjectPublicKeyInfo et KeyId sont les champs de gabarit qui peuvent être utilisés pour fournir une clé publique de signature pour laquelle un certificat est exigé (Voir les profils de l'Appendice D pour plus d'informations).

Voir à l'Appendice C et la [RFC4211] la syntaxe de CertReqMessages. Noter que si un historique des clés est nécessaire, le demandeur doit fournir une commande de clé de chiffrement de protocole dans le message de demande.

5.3.8 Contenu de réponse de récupération de clé

Pour les réponses de récupération de clé, la syntaxe suivante est utilisée. Pour certaines valeurs d'état (par exemple, attente) aucun des champs facultatifs ne sera présent.

```
KeyRecRepContent ::= SEQUENCE {
   status          PKIStatusInfo,
```

```
   newSigCert  [0] Certificate                FACULTATIF,
   caCerts     [1] SEQUENCE TAILLE (1..MAX) DE Certificate          FACULTATIF,
   keyPairHist [2] SEQUENCE TAILLE (1..MAX) DE CertifiedKeyPair FACULTATIF.
}
```

5.3.9 Contenu de demande de révocation

Lorsque on demande la révocation d'un certificat (ou de plusieurs certificats), la structure de données suivante est utilisée. Le nom du demandeur est présent dans la structure PKIHeader.

```
RevReqContent ::= SÉQUENCE DE RevDetails
   RevDetails ::= SEQUENCE {
      certDetails      CertTemplate,
      crlEntryDetails  Extensions      FACULTATIF
}
```

5.3.10 Contenu de réponse de révocation

La réponse de révocation est la réponse au message ci-dessus. Si elle est produite, c'est envoyé au demandeur de la révocation. (Un message séparé d'annonce de révocation PEUT être envoyé au sujet du certificat pour lequel la révocation a été demandée.)

```
RevRepContent ::= SEQUENCE {
    status      SEQUENCE TAILLE (1..MAX) DE PKIStatusInfo,
    revCerts  [0] SEQUENCE TAILLE (1..MAX) DE CertId              FACULTATIF,
    crls      [1] SEQUENCE TAILLE (1..MAX) DE CertificateList   FACULTATIF
}
```

5.3.11 Contenu de demande de certification croisée

Les demandes de certification croisée utilisent la même syntaxe (CertReqMessages) que les demandes de certification normales, avec la restriction que la paire de clés DOIT avoir été générée par la CA demandeuse et la clé privée NE DOIT PAS être envoyée à la CA qui répond. Cette demande PEUT aussi être utilisée par des CA subordonnées pour obtenir leurs certificats signés par la CA parente. Voir l'Appendice C et la [RFC4211] pour la syntaxe de CertReqMessages.

5.3.12 Contenu de réponse de certification croisée

Les réponses de certification croisée utilisent la même syntaxe (CertRepMessage) que les réponses normales de certification, avec la restriction qu'aucune clé privée chiffrée ne peut être envoyée. Voir au paragraphe Section 5.3.4 la syntaxe de CertRepMessage.

5.3.13 Contenu d'annonce de mise à jour de clé de CA

Lorsque une CA met à jour sa propre paire de clés, la structure de données suivante PEUT être utilisée pour annoncer cet événement.

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

5.3.14 Annonce de certificat

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

```
CertAnnContent ::= Certificate
```

5.3.15 Annonce de révocation

Quand une CA a révoqué, ou est sur le point de révoquer, un certificat particulier, elle PEUT produire une annonce de cet événement (éventuellement à venir).

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

Une CA PEUT utiliser une telle annonce pour avertir (ou notifier) un sujet que son certificat est sur le point d'être (ou a été) révoqué. Cela va normalement être utilisé lorsque la demande de révocation ne vient pas du sujet concerné.

Le champ willBeRevokedAt contient l'heure à laquelle une nouvelle entrée sera ajouté à la ou aux CRL pertinentes.

5.3.16 Annonce de CRL

Quand une CA produit une nouvelle CRL (ou ensembles de CRL) la structure de données suivante PEUT être utilisée pour annoncer cet événement.

```
CRLAnnContent ::= SÉQUENCE DE CertificateList
```

5.3.17 Contenu de confirmation PKI

Cette structure de données est utilisée dans l'échange de protocole comme message final PKI. Son contenu est le même dans tous les cas -- en fait, il n'y a pas de contenu car l'en-tête PKI porte 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. À réception d'un PKIConfirm pour une réponse de certificat, le receveur PEUT le traiter comme un certConf dont tous les certificats sont acceptés.

5.3.18 Contenu de confirmation de certificat

Cette structure de données est utilisée par le client pour envoyer la confirmation à la CA/RA pour accepter ou rejeter les certificats.

```
CertConfirmContent ::= SÉQUENCE DE CertStatus
   CertStatus ::= SEQUENCE {
     certHash    CHAÎNE D'OCTETS,
     certReqId   ENTIER,
     statusInfo  PKIStatusInfo FACULTATIF
   }
```

Pour tout CertStatus, l'omission du champ statusInfo indique l'acceptation du certificat spécifié. Autrement, les détails explicites d'état (par rapport à l'acceptation ou au rejet PEUVENT être fournis dans le champ statusInfo, peut-être pour des besoins d'examen à la CA/RA.

Dans CertConfirmContent, l'omission d'une structure CertStatus correspondant à un certificat fourni dans le message de réponse précédent indique le rejet du certificat. Donc, un CertConfirmContent vide (une SEQUENCE de longueur zéro) PEUT être utilisé pour indiquer le rejet de tous les certificats fournis. Voir au paragraphe 5.2.8, item (2), une discussion du champ certHash par rapport à la preuve de possession.

5.3.19 Contenu du message général de PKI

```
InfoTypeAndValue ::= SEQUENCE {
       infoType            IDENTIFIANT D'OBJET,
       infoValue           TOUT DÉFINI PAR infoType  FACULTATIF
```

```
 }
-- où {id-it} = {id-pkix 4} = {1 3 6 1 5 5 7 4}
    GenMsgContent ::= SÉQUENCE DE InfoTypeAndValue
```

5.3.19.1 Certificat de chiffrement de protocole de CA

Ceci PEUT être utilisé par l'EE pour obtenir un certificat de la CA à utiliser pour protéger des 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 à cette fin.

5.3.19.2 Types de paires de clé de signature

Ceci PEUT être utilisé par l'EE pour obtenir la liste des algorithmes de signature (par exemple, RSA, DSA) dont la CA veut certifier les valeurs de clé publique sujette. Noter que pour les besoins de cet échange, rsaEncryption et rsaWithSHA1, par exemple, sont considérés comme équivalents ; la question qui reste posée est, "la CA veut elle certifier une clé publique RSA ?"

GenMsg: {id-it 2}, < absent >
GenRep: {id-it 2}, SEQUENCE TAILLE (1..MAX) DE AlgorithmIdentifier

5.3.19.3 Types de paires de clé de chiffrement/accord de clé

Ceci PEUT être utilisé par l'EE pour obtenir la liste des algorithmes de chiffrement/accord de clé dont la CA veut certifier les valeurs de clé publique sujette.

GenMsg : {id-it 3}, < absent >
GenRep : {id-it 3}, SEQUENCE TAILLE (1..MAX) DE AlgorithmIdentifier

5.3.19.4 Algorithme symétrique préféré

Ceci PEUT être utilisé par le client pour obtenir l'algorithme de chiffrement symétrique préféré de la CA pour toutes informations confidentielles qui doivent être échangées entre l'EE et la CA (par exemple, si l'EE veut envoyer sa clé de déchiffrement privée à la CA à des fins d'archivage).

GenMsg : {id-it 4}, < absent >
GenRep : {id-it 4}, AlgorithmIdentifier

5.3.19.5 Paire de clés de CA mise à jour

Ceci PEUT être utilisé par la CA pour annoncer un événement de mise à jour de clé de CA.

GenMsg : {id-it 5}, CAKeyUpdAnnContent

5.3.19.6 CRL

Ceci PEUT être utilisé par le client pour obtenir une copie de la plus récente CRL.

GenMsg : {id-it 6}, < absent >
GenRep : {id-it 6}, CertificateList

5.3.19.7 Identifiants d'objets non pris en charge

Ceci est utilisé par le serveur pour retourner une liste d'identifiants d'objets qu'il ne reconnaît ni ne prend en charge sur la liste soumise par le client.

GenRep : {id-it 7}, SEQUENCE TAILLE (1..MAX) DE IDENTIFIANT D'OBJET

5.3.19.8 Paramètres de paire de clés

Ceci PEUT être utilisé par l'EE pour demander les paramètres de domaine à utiliser pour générer la paire de clés pour certains algorithmes à clé publique. Il peut être utilisé, par exemple, pour demander les P, Q, et G appropriés pour générer la clé DH/DSA, ou pour demander un ensemble de courbes elliptiques bien connues.

GenMsg : {id-it 10}, IDENTIFIANT D'OBJET -- (identifiant d'objet d'algorithme)
GenRep : {id-it 11}, AlgorithmIdentifier | < absent >

Une valeur d'information absente dans GenRep indique que l'algorithme spécifié dans GenMsg n'est pas accepté.

Les EE DOIVENT s'assurer que les paramètres leur sont acceptables et que le message GenRep est authentifié (pour éviter les attaques de substitution).

5.3.19.9 Mots de passe de révocation

Ceci PEUT être utilisé par l'EE pour envoyer un mot de passe à une CA/RA aux fins d'authentification dans une demande de révocation ultérieure (dans le cas où la clé privée de signature appropriée n'est plus disponible pour authentifier la demande). Voir à l'Appendice B d'autres détails sur l'utilisation de ce mécanisme.

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

5.3.19.10 ImplicitConfirm

(Confirmation implicite) Voir au paragraphe 5.1.1.1 la définition et l'usage ; {id-it 13}.

5.3.19.11 ConfirmWaitTime

(Temps d'attente de confirmation) Voir au paragraphe 5.1.1.2 la définition et l'usage ; {id-it 14}.

5.3.19.12 Original PKIMessage

(Message PKI d'origine) Voir au paragraphe 5.1.1.3 la définition et l'usage ; {id-it 15}.

5.3.19.13 Étiquettes de langue prises en charge

Ceci PEUT être utilisé pour déterminer l'étiquette de langue appropriée à utiliser dans les messages suivants. L'envoyeur envoie sa liste des langages pris en charge (dans l'ordre des préférences) ; le receveur retourne celui qu'il souhaite utiliser. (Note : chaque UTF8String DOIT inclure une étiquette de langue.) Si aucune des étiquettes offertes n'est acceptée, une erreur DOIT être retournée.

GenMsg : {id-it 16}, SEQUENCE TAILLE (1..MAX) DE UTF8String
GenRep : {id-it 16}, SEQUENCE TAILLE (1) DE UTF8String

5.3.20 Contenu de réponse générale de PKI

GenRepContent ::= SÉQUENCE DE InfoTypeAndValue

Des exemples de GenReps qui PEUVENT être prises en charge incluent celles mentionnées dans les subdivisions du paragraphe 5.3.19.

5.3.21 Contenu de message d'erreur

Cette structure de données PEUT être utilisée par l'EE, la CA, ou la RA pour convoyer des informations d'erreur.

```
ErrorMsgContent ::= SEQUENCE {
    pKIStatusInfo        PKIStatusInfo,
    errorCode            ENTIER          FACULTATIF,
    errorDetails         PKIFreeText     FACULTATIF
 }
```

Ce message PEUT être généré à tout moment durant une transaction PKI. Si le client envoie cette demande, le serveur DOIT répondre avec une PKIConfirm, ou un autre ErrorMsg si une partie de l'en-tête n'est pas valide. Les deux côtés DOIVENT traiter ce message comme la fin de la transaction (si une transaction est en cours).

Si une protection est désirée sur le message, le client DOIT le protéger en utilisant la même technique (c'est-à-dire, signature ou MAC) que le message de début de la transaction. La CA DOIT toujours le signer avec une clé de signature.

5.3.22 Demande et réponse d'interrogation

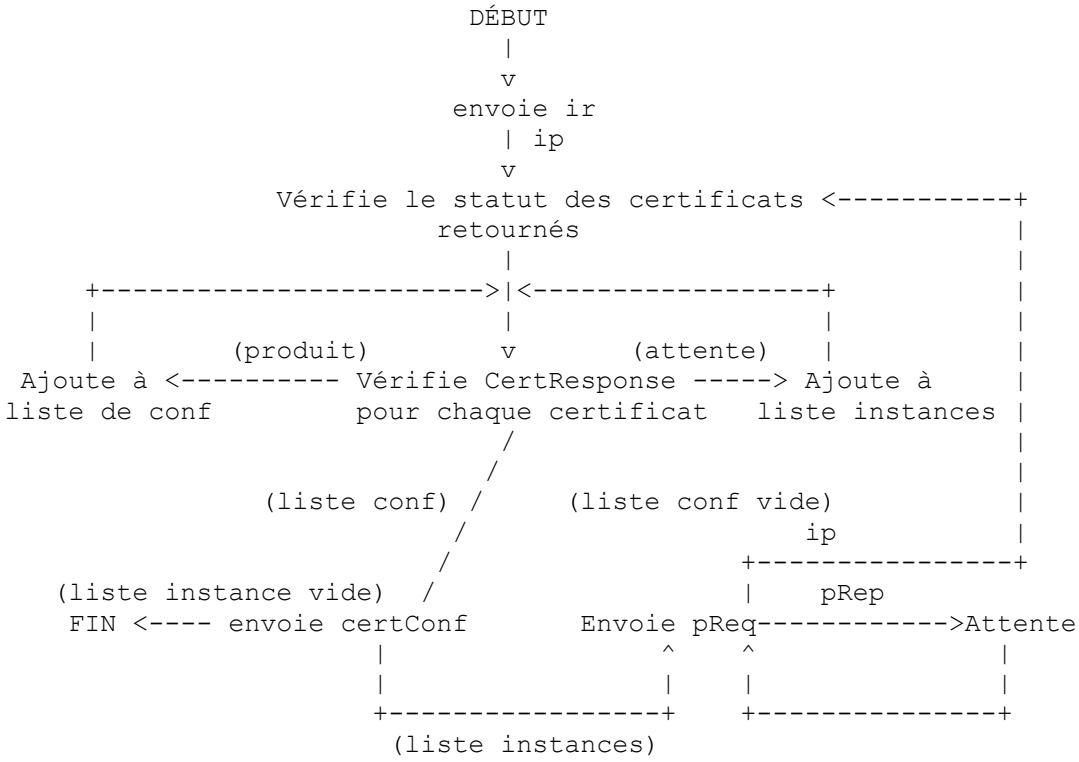
Cette paire de messages est destinée à traiter les scénarios dans lesquels le client a besoin d'interroger le serveur pour déterminer l'état d'une transaction ir, cr, ou kur en cours (c'est-à-dire, quand le PKIStatus "attente" a été reçu).

PollReqContent ::= SÉQUENCE DE SEQUENCE { certReqId ENTIER }

PollRepContent ::= SÉQUENCE DE SEQUENCE {
certReqId ENTIER,
checkAfter ENTIER, -- heure en secondes
reason PKIFreeText FACULTATIF }

Les points suivants décrivent quand les messages d'interrogation sont utilisés et comment ils le sont. On suppose que plusieurs messages certConf peuvent être envoyés durant les transactions. Il y en aura un envoyé en réponse à chaque ip, cp, ou kup qui contient un CertStatus pour un certificat produit.

1. En réponse à un message ip, cp, ou kup, un EE va envoyer un certConf pour tous les certificats produits et, suite à l'accusé de réception, une pollReq pour tous les certificats en instance.
2. En réponse à une pollReq, une CA/RA va retourner un ip, cp, ou kup si un ou plusieurs des certificats en instance est prêt ; autrement, elle va retourner un pollRep.
3. Si l'EE reçoit un pollRep, il va attendre au moins la valeur de checkAfter avant d'envoyer une autre pollReq.
4. Si un ip, cp, ou kup est reçu en réponse à une pollReq, elle va être traitée de la même façon que la réponse initiale.



Dans l'échange suivant, l'entité d'extrémité s'inscrit pour deux certificats dans une demande.

Étape	Entité d'extrémité		PKI
1	Formate ir		
2		-> ir ->	
3			Traite ir
4			Une intervention manuelle est requise pour les deux certificats.

```
5                       <- ip     <-
6     Traite ip
7     Formate pReq
8                       -> pReq    ->
9                                  Vérifie l'état des demandes de certificat
10                                 Certificats non prêts
11                                 Formate pRep
12                      <- pRep    <-
13    Attente
14    Formate pReq
15                      -> pReq    ->
16                                 Vérifie l'état des demandes de certificat
17                                 Un certificat est prêt
18                                 Formate ip
19                      <- ip      <-
20    Traite ip
21    Formate certConf
22                      -> certConf ->
23                                 Traite certConf
24                                 Formate ack
25                      <- pkiConf  <-
26    Formate pReq
27                      -> pReq    ->
28                                 Vérifie l'état des demandes de certificat
29                                 Le certificat est prêt
30                                 Formate ip
31                      <- ip      <-
31    Traite ip
32    Formate certConf
33                      -> certConf ->
34                                 Traite certConf
35                                 Formate ack
36                      <- pkiConf  <-
```

6. Fonctions obligatoires de gestion de PKI

La présente Section décrit certaines des fonctions de gestion de PKI mentionnées au paragraphe 3.1.

Elle traite des fonctions qui sont "obligatoires" en ce sens que toutes les mises en œuvre d'entité d'extrémité et de CA/RA DOIVENT être capables de fournir la fonctionnalité décrite. Cette partie est effectivement le profil de la fonction de gestion de PKI qui DOIT être pris en charge. Noter, cependant, que les fonctions de gestion décrites dans cette section ne doivent pas nécessairement être accomplies en utilisant les messages de PKI définis dans la Section 5 si d'autres moyens conviennent pour un certain environnement (voir à l'Appendice D les profils de messages de PKI qui DOIVENT être pris en charge).

6.1 Initialisation de CA racine

[Voir au paragraphe 3.1.1.2 la définition de " CA racine".]

Une CA racine nouvellement créée doit produire un "auto certificat", qui est une structure Certificate avec le profil défini pour le certificat "newWithNew" produit à la suite d'une mise à jour de clé de CA racine.

Pour rendre utile l'auto certificat de CA pour les entités d'extrémité qui n'acquièrent pas l'auto certificat via des moyens "hors bande", la CA doit aussi produire une "empreinte digitale" pour son certificat. Les entités d'extrémité qui acquièrent cette empreinte digitale en toute sécurité via des moyens "hors bande" peuvent alors vérifier l'auto certificat de la CA et donc, les autres attributs qui y sont contenus.

La structure de données utilisée pour porter l'empreinte digitale est le OOBCertHash.

6.2 Mise à jour de clé de CA racine

Les clés de CA (comme toutes les autres clés) ont une durée de vie finie et devront être mises à jour périodiquement. Les certificats NouveauAvecNouveau, NouveauAvecVieux, et VieuxAvecNouveau (voir au paragraphe 4.4.1) PEUVENT être produits par la CA pour aider les entités d'extrémité existantes qui détiennent le certificat auto signé en cours de la CA (VieuxAvecVieux) à transiter en toute sécurité au nouveau certificat auto signé de la CA (NouveauAvecNouveau) et pour aider les nouvelles entités d'extrémité qui vont détenir le NouveauAvecNouveau à acquérir en toute sécurité le VieuxAvecVieux pour la vérification des données existantes.

6.3 Initialisation de CA subordonnée

[Voir au paragraphe 3.1.1.2 la définition de "CA subordonnée".]

Du point de vue des protocoles de gestion de PKI, l'initialisation d'une CA subordonnée est semblable à l'initialisation d'une entité d'extrémité. La seule différence est que la CA subordonnée doit aussi produire une liste de révocation initiale.

6.4 Production de CRL

Avant toute production de certificats, une CA qui vient d'être établie (qui produit des CRL) doit produire des versions "vides" de chaque CRL qui seront produites périodiquement.

6.5 Demande d'informations de PKI

Lorsque une entité PKI (CA, RA, ou EE) souhaite acquérir des informations sur l'état actuel d'une CA, elle PEUT envoyer à cette CA une demande sur ces informations.

La CA DOIT répondre à la demande en fournissant (au moins) toutes les informations demandées. Si des informations ne peuvent pas être fournies, une erreur doit alors être envoyée au demandeur.

Si les messages de PKI sont utilisés pour demander et fournir ces informations de PKI, la demande DOIT alors être le message GenMsg *(message général)*, la réponse DOIT être le message GenRep *(réponse générale)*, et l'erreur DOIT être le message Error. Ces messages sont protégés en utilisant un MAC fondé sur des informations secrètes partagées (c'est-à-dire, PasswordBasedMAC) ou en utilisant tout autre moyen authentifié (si l'entité d'extrémité a un certificat existant).

6.6 Certification croisée

La CA demandeuse est la CA qui va devenir le sujet du certificat croisé ; la CA qui répond va devenir le producteur du certificat croisé.

La CA demandeuse CA doit être "en fonctionnement actif" avant d'initier l'opération de certification croisée.

6.6.1 Schéma unidirectionnel de demande-réponse

Le schéma de certification croisée est essentiellement une opération unidirectionnelle ; c'est-à-dire que quand elle réussit, cette opération résulte en la création d'un nouveau certificat croisé. Si l'exigence est que ces certificats croisés soient créés dans les "deux directions", alors chaque CA, à son tour, doit initier une opération de certification croisée (ou utiliser un autre schéma).

Ce schéma convient lorsque les deux CA en question peuvent déjà vérifier les signatures de l'autre (ils ont quelques points de confiance communs) ou lorsque il y a une vérification hors bande de l'origine de la demande de certification.

Description détaillée : la certification croisée est initiée à une CA connue de celui qui répond. L'administrateur de la CA pour celui qui répond identifie la CA avec qui il veut faire une certification croisée et l'équipement de la CA qui répond génère un code d'autorisation. L'administrateur de la CA qui répond passe ce code d'autorisation par des moyens hors bande à l'administrateur de la CA demandeuse. L'administrateur de la CA demandeuse entre le code d'autorisation dans la CA demandeuse afin d'initier l'échange en ligne.

Le code d'autorisation est utilisé aux fins d'authentification et de garantie de l'intégrité. C'est fait en générant une clé symétrique fondée sur le code d'autorisation et en utilisant la clé symétrique pour générer des codes d’authentification de message (MAC) sur tous les messages échangés. (L'authentification peut aussi être faite en utilisant des signatures au lieu

des MAC, si les CA sont capables de restituer et valider les clés publiques requises par certains moyens, comme la comparaison de hachage hors bande.)

La CA demandeuse initie l'échange en générant une demande de certification croisée (ccr) avec un nombre aléatoire frais (nombre aléatoire du demandeur). La CA demandeuse envoie alors le message ccr à la CA qui répond. Les champs dans ce message sont protégés de la modification par un MAC fondé sur le code d'autorisation.

À réception du message ccr, la CA répondeuse valide le message et le MAC, sauvegarde le nombre aléatoire du demandeur, et génère son propre nombre aléatoire (nombre aléatoire du répondeur). Ensuite, elle génère (et archive, si elle le désire) un nouveau certificat de demandeur qui contient la clé publique de CA du demandeur et est signé avec la clé privée de signature de la CA répondeuse. La CA répondeuse répond avec le message de réponse de certification croisée (ccp). Les champs de ce message sont protégés de la modification par un MAC fondé sur le code d'autorisation.

À réception du message ccp, la CA demandeuse valide le message (incluant les nombres aléatoires reçus) et le MAC. La CA demandeuse répond avec le message certConf. Les champs de ce message sont protégés de la modification par un MAC fondé sur le code d'autorisation. La CA demandeuse PEUT écrire le certificat de demandeur dans le répertoire pour l'aider à la construction ultérieure du chemin de certification.

À réception du message certConf, la CA répondeuse valide le message et le MAC, et renvoie un accusé de réception en utilisant le message PKIConfirm. Elle PEUT aussi publier le certificat de demandeur pour l'aider à la construction ultérieure du chemin de certification.

Note : le message ccr doit contenir une demande de certification "complète" ; c'est-à-dire, tous les champs excepté le numéro de série (incluant, par exemple, une extension BasicConstraints) doivent être spécifiés par la CA demandeuse. Le message ccp DEVRAIT contenir le certificat de vérification de la CA répondeuse ; si il est présent, la CA demandeuse doit alors vérifier ce certificat (par exemple, via le mécanisme hors bande).

(Un modèle plus simple, non interactif, de certification croisée peut aussi être envisagé, dans lequel la CA productrice acquiert la clé publique de la CA sujette à partir d'un répertoire, le vérifie via un mécanisme hors bande, et crée et publie le certificat croisé sans l'implication explicite de la CA sujette. Ce modèle peut être parfaitement légitime pour de nombreux environnements, mais comme il n'exige aucun échange de messages de protocole, sa description détaillée sort du domaine d'application de la présente spécification.)

6.7 Initialisation d'entité d'extrémité

Comme les CA, les entités d'extrémité doivent être initialisées. L'initialisation des entités d'extrémité exige au moins deux étapes :
- o acquisition des informations de PKI,
- o vérification hors bande d'une clé publique de CA racine.

(D'autres étapes possibles incluent la restitution des informations de condition de confiance et/ou une vérification hors bande des autres clés publiques de CA).

6.7.1 Acquisition des informations de PKI

Les informations EXIGÉES sont :
- o la clé publique de CA racine actuelle,
- o (si la CA qui certifie n'est pas une CA racine) le chemin de certification de la CA racine à la CA qui certifie ainsi que les listes de révocation appropriées,
- o les algorithmes et les paramètres d'algorithme que la CA qui certifie prend en charge pour chaque usage pertinent.

Des informations supplémentaires pourraient être requises (par exemple, les extensions acceptées ou des informations de politique de la CA) afin de produire une demande de certification qui réussisse. Cependant, par souci de simplicité, on ne rend pas obligatoire que l'entité d'extrémité acquière ces informations via les messages de PKI. Le résultat final est simplement que certaines demandes de certification peuvent échouer (par exemple, si l'entité d'extrémité veut générer sa propre clé de chiffrement, mais que la CA ne le permet pas).

Les informations requises PEUVENT être acquises comme décrit au paragraphe 6.5.

6.7.2 Vérification hors bande de la clé de CA racine

Une entité d'extrémité doit posséder en toute sécurité la clé publique de sa CA racine. Une méthode pour réaliser cela est de fournir à l'entité d'extrémité une empreinte digitale de l'auto certificat de CA via un moyen hors bande sûr. L'entité d'extrémité peut alors utiliser en toute sécurité l'auto certificat de CA. Voir les détails au paragraphe 6.1.

6.8 Demande de certificat

Une entité d'extrémité initialisée PEUT demander un certificat supplémentaire à tout moment (à toutes fins utiles). Cette demande sera faite en utilisant le message de demande de certification (cr). Si l'entité d'extrémité possède déjà une paire de clés de signature (avec un certificat de vérification correspondant) ce message cr va alors être normalement protégé par la signature numérique de l'entité. La CA retourne le nouveau certificat (si la demande est réussie) dans un CertRepMessage.

6.9 Mise à jour de clé

Lorsque une paire de clés est sur le point d'arriver à expiration, l'entité d'extrémité pertinente PEUT demander une mise à jour de clé ; c'est-à-dire qu'elle PEUT demander que la CA produise un nouveau certificat pour une nouvelle paire de clés (ou, dans certaines circonstances, un nouveau certificat pour la même paire de clés). La demande est faite en utilisant un message demande de mise à jour de clé (kur, *key update request*) (appelé, dans certains environnements, une opération "Mise à jour de certificat"). Si l'entité d'extrémité possède déjà une paire de clés de signature (avec le certificat de vérification correspondant) ce message va alors être normalement protégé par la signature numérique de l'entité. La CA retourne le nouveau certificat (si la demande réussit) dans un message de réponse de mise à jour de clé (kup, *key update response*) qui est syntaxiquement identique à un CertRepMessage.

7. Négociation de version

Cette section définit la négociation de version utilisée pour prendre en charge de plus anciens protocoles entre clients et serveurs.

Si un client connaît les versions de protocole prises en charge par le serveur (par exemple, à partir d'un échange antérieur de messages de PKI ou via des moyens hors bande) il DOIT alors envoyer un message de PKI avec le plus fort numéro de version accepté par lui et le serveur. Si un client ne sait pas quelles versions accepte le serveur, il DOIT alors envoyer un message de PKI en utilisant la plus forte version qu'il prend en charge.

Si un serveur reçoit un message avec une version qu'il prend en charge, la version du message de réponse DOIT être la même que la version reçue. Si un serveur reçoit un message avec une version supérieure ou inférieure à ce qu'il prend en charge, il DOIT alors renvoyer un message d'erreur avec le bit Version non prise en charge établi (dans le champ failureInfo *(informations d'échec)* de pKIStatusInfo *(informations d'état de PKI)*). Si la version reçue est supérieure à la plus forte version prise en charge, la version dans le message d'erreur DOIT alors être la plus forte version que le serveur prend en charge ; si la version reçue est inférieure à la plus basse version prise en charge, la version dans le message d'erreur DOIT être alors la plus basse version que le serveur prend en charge.

Si un client reçoit en retour un ErrorMsgContent *(erreur de contenu de message)* avec le bit Version non prise en charge établi et une version qu'il prend en charge, il PEUT alors réessayer la demande avec cette version.

7.1 Prise en charge des mises en œuvre de la RFC 2510

La RFC 2510 ne spécifiait pas le comportement des mises en œuvre qui reçoivent des versions qu'elle ne comprennent pas car il n'existait alors qu'une seule version. Avec l'introduction de la présente révision de la spécification, le comportement à l'égard des versions est recommandé.

7.1.1 Clients qui parlent à des serveurs de la RFC 2510

Si, après l'envoi d'un message cmp2000, un client reçoit un ErrorMsgContent avec une version de cmp1999, il DOIT alors interrompre la transaction en cours. Il PEUT ensuite réessayer la transaction en utilisant des messages de version cmp1999.

Si un client reçoit un message PKI de non erreur avec une version de cmp1999, il PEUT alors décider de continuer la transaction (si la transaction ne s'est pas terminée) en utilisant la sémantique de la RFC 2510. Si il ne choisit pas de faire ainsi et si la transaction n'est pas terminée, il DOIT alors interrompre la transaction et envoyer un ErrorMsgContent avec une version de cmp1999.

7.1.2 Serveurs qui reçoivent des messages PKI de version cmp1999

Si un serveur reçoit un message de version cmp1999, il PEUT revenir au comportement de la RFC 2510 et répondre avec des messages de version cmp1999. Si il ne choisit pas de faire ainsi, il DOIT alors renvoyer un ErrorMsgContent comme expliqué à la Section 7.

8. Considérations sur la sécurité

8.1 Preuve de possession avec clé de déchiffrement

Quelques considérations sur la cryptographie doivent être explicitement examinées. Dans les protocoles spécifiés ci-dessus, quand une entité d'extrémité est requise de prouver la possession d'une clé de déchiffrement, elle est effectivement mise au défi de déchiffrer quelque chose (son propre certificat). Ce schéma (et bien d'autres !) pourrait être vulnérable à une attaque si le possesseur de la clé de déchiffrement en question pouvait être conduit par tromperie à déchiffrer un défi arbitraire et à retourner le texte en clair à l'attaquant. Bien qu'un certain nombre d'autres défaillances de la sécurité de la présente spécification soient requises afin que cette attaque réussisse, il est concevable que des services futurs (par exemple, notariat, heure de confiance) pourraient être vulnérables à de telles attaques. Pour cette raison, on réitère la règle générale que les mises en œuvre devraient être très attentives lors du déchiffrement de "texte chiffré" arbitraire joint à la révélation du "texte source" récupéré car une telle pratique peut conduire à de sérieuses vulnérabilités de la sécurité.

8.2 Preuve de possession par exposition de la clé privée

Noter aussi qu'exposer une clé privée à la CA/RA est une technique de preuve de possession qui peut entraîner certains risques pour la sécurité (selon que si la CA/RA peut ou non être de confiance pour traiter de façon appropriée de tels matériaux). Il est conseillé que les mises en œuvre fassent attention lors du choix et de l'utilisation de ce mécanisme de POP, et que lorsque approprié, l'utilisateur de l'application déclare explicitement qu'il accepte en confiance que la CA/RA ait une copie de sa clé privée avant de procéder à la révélation de la clé privée.

8.3 Attaques contre l'échange de clé Diffie-Hellman

Une petite attaque de sous groupe durant un échange de clé Diffie-Hellman peut être effectuée comme suit. Une entité d'extrémité malveillante peut délibérément choisir des paramètres D-H qui lui permettent de déduire la clé privée de la CA (ou un nombre significatif des bits de celle-ci) durant l'archivage de la clé ou l'opération de récupération de la clé. Armé de cette connaissance, l'EE va alors être capable de restituer la clé privée de déchiffrement d'une autre entité d'extrémité qui ne la soupçonne pas, EE2, durant l'opération légitime d'archivage ou de récupération de clé de EE2 avec cette CA. Afin d'éviter la possibilité d'une telle attaque, deux attitudes sont disponibles : (1) la CA peut générer une paire de clés D-H fraîche à utiliser comme paire de clés de chiffrement de protocole pour chaque EE avec laquelle elle interagit ; (2) la CA peut entrer dans un protocole de validation de clé (non spécifié dans le présent document) avec chaque entité d'extrémité demandeuse pour s'assurer que la paire de clés de chiffrement de protocole de l'EE ne va pas faciliter cette attaque. L'option (1) est clairement plus simple (n'exigeant pas d'échange de protocole supplémentaire de la part de l'une ou l'autre partie) et est donc RECOMMANDÉE.

9. Considérations relatives à l'IANA

Les types généraux de message de PKI sont identifiés par des identifiants d'objet (OID, *object identifier*). Les OID pour les types généraux de message de PKI définis dans le présent document ont été alloués à partir d'un arc délégué par l'IANA au groupe de travail PKIX.

Les algorithmes cryptographiques auxquels le présent document fait référence sont identifiés par des identifiants d'objet (OID). Les OID pour les algorithmes de chiffrement ont été alloués à partir de plusieurs arcs appartenant à diverses organisations, incluant RSA Security, Entrust Technologies, l'IANA et l'IETF.

Si des algorithmes de chiffrement supplémentaires devaient être introduits, les promoteurs de ces algorithmes sont supposés allouer les OID nécessaires à partir de leurs propres arcs.

Aucune autre action de l'IANA n'est nécessaire pour le présent document ou ses mises à jour prévues.

Références normatives

[X509] Recommandation UIT-T X.509 , "Technologies de l'information - Interconnexion des systèmes ouverts - L'annuaire : cadres de clé publique et de certificat d'attribut", aussi norme ISO 9594-8:2001, mars 2000.

[MvOV97] Menezes, A., van Oorschot, P. and S. Vanstone, "Handbook of Applied Cryptography", CRC Press ISBN 0-8493-8523-7, 1996.

[RFC2104] H. Krawczyk, M. Bellare et R. Canetti, "HMAC : Hachage de clés pour l'authentification de message", février 1997.

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

[RFC2202] P. Cheng et R. Glenn, "Cas d'essai pour HMAC-MD5 et HMAC-SHA-1", septembre 1997. *(Information)*

[RFC2482] K. Whistler, G. Adams, "Étiquettes de langues dans un libellé Unicode", janvier 1999. *(Historique)*

[RFC3066] H. Alvestrand, "Étiquettes pour l'identification des langues", BCP 47, janvier 2001. *(Obsolète, voir la* RFC4646.)

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

[RFC4211] J. Schaad, "Format de message de demande de certificat d'infrastructure de clé publique X.509 pour l'Internet", septembre 2005. (*Remplace* RFC2511) *(P.S.)*

Références pour information

[ANSI-X9.42] American National Standards Institute, "Public Key Cryptography for The Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography", ANSI X9.42, février 2000.

[FIPS-180] National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-1, mai 1994.

[FIPS-186] National Institute of Standards and Technology, "Digital Signature Standard", FIPS PUB 186, mai 1994.

[PKCS7] RSA Laboratories, "The Public-Key Cryptography Standards - Cryptographic Message Syntax Standard. Version 1.5", PKCS 7, novembre 1993.

[PKCS11] RSA Laboratories, "The Public-Key Cryptography Standards - Cryptographic Token Interface Standard. Version 2.10", PKCS 11, décembre 1999.

[RFC1847] J. Galvin, S. Murphy, S. Crocker, N. Freed, "Sécurité multiparties pour MIME : multipartie/signée et multipartie/chiffrée", octobre 1995. *(P.S.)*

[RFC2559] S. Boeyen, T. Howes, P. Richard, "Protocoles de fonctionnement de l'infrastructure de clés publiques X.509 sur Internet - LDAPv2", avril 1999. (*Obsolète, voir* RFC3494) (*MàJ* RFC1778) *(Historique)*

[RFC2585] R. Housley et P. Hoffman, "Protocoles de fonctionnement de l'infrastructure de clé publique X.509 pour l'Internet : FTP et HTTP", mai 1999. *(P.S.)*

[RFC2986] M. Nystrom, B. Kaliski, "PKCS n° 10 : Spécification de la syntaxe de demande de certification, version 1.7", novembre 2000. *(Information)*

[RFC6712] T. Kause, M. Peylo, "Infrastructure de clé publique X.509 sur Internet -- Transfert HTTP pour le protocole de gestion de certificat (CMP)", septembre 2012. (*MàJ la* RFC4210) *(P.S.)*

Appendice A. Raisons de la présence des RA

Les raisons qui justifient la présence d'une RA peuvent se partager en celles qui sont dues à des facteurs techniques et celles qui sont de nature organisationnelle. Les raisons techniques incluent ce qui suit :

- Si on utilise des jetons matériels, toutes les entités d'extrémité n'auront pas l'équipement nécessaire pour les initialiser ; l'équipement de RA peut inclure la fonction nécessaire (ce peut aussi être une affaire de politique).
- Certaines entités d'extrémité peuvent n'avoir pas la capacité de publier les certificats ; là encore, la RA peut être convenablement placée pour cela.
- La RA sera capable de produire des demandes de révocation signées au nom des entités d'extrémité qui lui sont associées, tandis que l'entité d'extrémité peut n'être pas capable de faire cela (si la paire de clés est complètement perdue.

Certaines des raisons organisationnelles qui militent en faveur de la présence d'une RA sont :

- Il peut être plus rentable de concentrer les fonctionnalités dans l'équipement de RA que de fournir cette fonctionnalité à toutes les entités d'extrémité (en particulier si un équipement spécial d'initialisation de jeton doit être utilisé).
- Établir des RA au sein d'une organisation peut réduire le nombre de CA nécessaires, ce qui est parfois souhaitable.
- Les RA peuvent être mieux placées pour identifier les gens avec leur nom "électronique", en particulier si la CA est physiquement éloignée de l'entité d'extrémité.
- Pour de nombreuses applications, il y aura déjà en place une structure administrative de sorte que les candidats au rôle de RA seront faciles à trouver (ce qui peut n'être pas vrai de la CA).

Appendice B Utilisation de phrase de passe de révocation

La demande de révocation doit incorporer des mécanismes de sécurité convenables, incluant une authentification appropriée, afin de réduire la probabilité de réussite des attaques de déni de service. Une signature numérique sur la demande -- de mise en œuvre OBLIGATOIRE dans cette spécification si les demandes de révocation sont prises en charge -- peut fournir l'authentification requise, mais il y a des circonstances dans lesquelles un mécanisme de remplacement peut être désirable (par exemple, quand la clé privée n'est plus accessible et que l'entité souhaite demander une révocation avant la re-certification d'une autre paire de clés). Afin de s'accommoder de telles circonstances, un PasswordBasedMAC *(MAC fondé sur le mot de passe)* sur la demande est aussi de mise en œuvre OBLIGATOIRE dans la présente spécification (sous réserve de la politique locale de sécurité pour un environnement donné) si les demandes de révocation sont prises en charge et si les informations de secret partagé peuvent être établies entre le demandeur et le répondant avant qu'il soit besoin de la révocation.

Un mécanisme qui a été vu en usage dans certains environnements est la "phrase de passe de révocation", dans lequel une valeur d'une entropie suffisante (c'est-à-dire, une phrase de passe relativement longue plutôt qu'un bref mot de passe) est partagée entre (seulement) l'entité et la CA/RA à un certain moment avant la révocation ; cette valeur est ensuite utilisée pour authentifier la demande de révocation.

Dans cette spécification, la technique suivante pour établir des informations de secret partagé (c'est-à-dire, une phrase de passe de révocation) est de prise en charge FACULTATIVE. Son utilisation précise dans les messages de CMP est la suivante :

- L'OID et la valeur spécifiés au paragraphe 5.3.19.9 PEUVENT être envoyés dans un message GenMsg à tout moment, ou PEUVENT être envoyés dans le champ generalInfo de l'en-tête de PKI de tout message PKI à tout moment. (En particulier, la EncryptedValue peut être envoyée dans l'en-tête du message certConf qui confirme l'acceptation des certificats demandés dans un message de demande d'initialisation ou de certificat.) Cela porte une phrase de passe de révocation choisie par l'entité (c'est-à-dire, les octets déchiffrés du champ encValue) à la CA/RA pertinente ; de plus, le transfert est réalisé avec les caractéristiques appropriées de confidentialité (parce que la phrase de passe est chiffrée sous la clé de chiffrement de protocole de la CA/RA).

- Si une CA/RA reçoit la phrase de passe de révocation (OID et valeur spécifiés au paragraphe 5.3.19.9) dans un GenMsg, elle DOIT construire et envoyer un message GenRep qui comporte l'OID (avec la valeur absente) spécifié au paragraphe 5.3.19.9. Si la CA/RA reçoit la phrase de passe de révocation dans le champ generalInfo d'un en-tête PK de tout message de PKI, elle DOIT inclure l'OID (avec la valeur absente) dans le champ generalInfo de l'en-tête PKI du message de réponse PKI correspondant. Si la CA/RA est incapable de retourner le message de réponse approprié pour une raison quelconque, elle DOIT envoyer un message d'erreur avec un état de "rejet" et facultativement une raison failInfo établie.

- Le champ valueHint de EncryptedValue PEUT contenir un identifiant de clé (choisi par l'entité, ainsi que la phrase de passe elle-même) pour aider à une restitution ultérieure de la phrase de passe correcte (par exemple, quand la demande de révocation est construite par l'entité et reçue par la CA/RA).

- Le message de demande de révocation est protégé par un PasswordBasedMAC, avec la phrase de passe de révocation comme clé. Si c'est approprié, le champ senderKID dans l'en-tête PKI PEUT contenir la valeur précédemment transmise dans valueHint.

En utilisant la technique spécifiée ci-dessus, la phrase de passe de révocation peut être établie initialement et mise à jour à tout moment sans exiger de messages supplémentaires ou d'échanges hors bande. Par exemple, le message de demande de révocation lui-même (protégé et authentifié par un MAC qui utilise la phrase de passe de révocation comme clé) peut contenir, dans l'en-tête PKI, une nouvelle phrase de passe de révocation à utiliser pour authentifier de futures demandes de révocation pour tout autre certifiat de l'entité. Dans certains environnements ceci peut être préférable aux mécanismes qui révèlent la phrase de passe dans le message de demande de révocation, car cela peut permettre une attaque de déni de service dans laquelle la phrase de passe révélée est utilisée par un tiers non autorisé pour authentifier des demandes de révocation sur d'autres certificats de l'entité. Cependant, comme la phrase de passe n'est pas révélée dans le message de demande, il n'est pas exigé que la phrase de passe doive toujours être mise à jour quand une demande de révocation est faite (c'est-à-dire que la même phrase de passe PEUT être utilisée par une entité pour authentifier les demandes de révocation pour des certificats différents à différents moments).

De plus, la technique ci-dessus peut fournir une forte protection cryptographique sur le message entier de demande de révocation même quand on utilise pas de signature numérique. Les techniques qui font l'authentification de la demande de révocation en révélant simplement la phrase de passe de révocation ne fournissent normalement pas de protection cryptographique sur les champs du message de demande (de sorte qu'une demande de révocation d'un certificat peut être modifiée par un tiers non autorisé en une demande de révocation d'un autre certificat pour cette entité).

Appendice C. Précisions sur le comportement quant au message de demande

Dans le cas de mises à jour à la [RFC4211] qui causeraient des problèmes d'interprétation ou d'interopérabilité, la [RFC4211] DEVRAIT être le document normatif.

Les définitions suivantes sont tirées de la [RFC4211]. Elles sont incluse ici afin de codifier les éclaircissements comportementaux à ce message de demande ; autrement, toute la syntaxe et la sémantique sont identiques à celles de la [RFC4211].

```
CertRequest ::= SEQUENCE {
     certReqId       ENTIER,
     certTemplate  CertTemplate,
     controls        Controls FACULTATIF }
```

-- Si certTemplate est une SEQUENCE vide (c'est-à-dire, si tous les champs sont omis) les commandes PEUVENT contenir la commande id-regCtrl-altCertTemplate, qui spécifie un gabarit pour un certificat autre qu'un certificat de clé publique X.509 v3. À l'inverse, si certTemplate n'est pas vide (c'est-à-dire, si au moins un champ est présent) alors les commandes NE DOIVENT PAS contenir id-regCtrl-altCertTemplate. La nouvelle commande est définie comme suit :

```
IDENTIFIANT D'OBJET id-regCtrl-altCertTemplate ::= {id-regCtrl 7}
  AltCertTemplate ::= AttributeTypeAndValue

POPOSigningKey ::= SEQUENCE {
   poposkInput          [0] POPOSigningKeyInput FACULTATIF,
   algorithmIdentifier  AlgorithmIdentifier,
   signature            CHAÎNE BINAIRE }
```

-- *Pour les besoins de la présente spécification, le commentaire ASN.1 donné dans la [RFC4211] relève non seulement de certTemplate, mais aussi de la commande altCertTemplate. C'est-à-dire, "la signature (utilisant "algorithmIdentifier") est sur la valeur codée en DER de poposkInput (c'est-à-dire, la "valeur" OCTET du DER de POPOSigningKeyInput).

Note : Si CertReqMsg certReq certTemplate (ou le contrôle altCertTemplate) contient les valeurs de sujet et de clé publique, poposkInput DOIT alors être omis et la signature DOIT être calculée sur la valeur codée en DER de CertReqMsg certReq (ou la valeur codée en DER de AltCertTemplate). Si certTemplate/altCertTemplate ne contient

ni la valeur de sujet ni celle de clé publique (c'est-à-dire, si il contient seulement un des deux, ou aucun) poposkInput DOIT alors être présent et DOIT être signé. -- *

```
POPOPrivKey ::= CHOIX {
   thisMessage       [0] CHAÎNE BINAIRE,
-- *Le type de "thisMessage" est donné comme CHAÎNE BINAIRE dans la [RFC4211] ; il devrait être "EncryptedValue"
   (en accord avec le paragraphe 5.2.2, "Valeurs chiffrées", de cette spécification). Donc, le présent document fait
   l'éclaircissement comportemental de spécifier que le contenu de "thisMessage" DOIT être codé comme une
   EncryptedValue et ensuite enveloppé comme une CHAÎNE BINAIRE. Cela permet la portabilité nécessaire et la
   protection de la clé privée tout en maintenant la compatibilité des bits du réseau avec la [RFC4211]. -- *
   subsequentMessage [1] SubsequentMessage,
   dhMAC             [2] CHAÎNE BINAIRE }
```

Appendice D. Profils (EXIGÉS). de message de gestion de PKI

Cet Appendice contient les profils détaillés pour les messages de PKI qui DOIVENT être pris en charge par les mises en œuvre conformes (voir la Section 6).

On donne les profils pour les messages de PKI utilisés dans les opérations de gestion de PKI suivantes :
- o enregistrement/certification initial
- o schéma authentifié de base
- o demande de certificat
- o mise à jour de clé

D.1 Règles générales pour l'interprétation de ces profils

1. Lorsque des champs "FACULTATIF" ou "PAR DÉFAUT" ne sont pas mentionnés dans les profils individuels, ils DEVRAIENT être absents du message concerné (c'est-à-dire, un receveur peut légitimement rejeter un message contenant de tels champs comme étant syntaxiquement incorrects). Les champs obligatoires ne sont pas mentionnés si ils ont une valeur évidente (par exemple, dans cette version de la spécification, pvno est toujours 2).

2. Lorsque des structures se produisent dans plus d'nn message, leurs profils sont séparés comme approprié.

3 Les identifiants d'algorithme des structures de message PKI sont profilés séparément.

4. Un nom distinctif X.500 "spécial" est appelé "NULL-DN" ; cela signifie un nom distinctif qui contient une SÉQUENCE DE RelativeDistinguishedNames de longueur zéro (son codage en DER est '3000'H).

5. Lorsque un GeneralName est exigé pour un champ, mais qu'aucune valeur convenable n'est disponible (par exemple, une entité d'extrémité produit une demande avant de savoir son nom) le GeneralName doit alors être un NULL-DN X.500 (c'est-à-dire, le champ Nom du CHOIX va contenir un NULL-DN). Cette valeur spéciale peut être appelée un "NomGénéral-NULL".

6. Lorsque un profil omet de spécifier la valeur pour un nom général, la valeur de NomGénéral-NULL doit être présente dans le champ de message de PKI pertinent. Cela se produit avec le champ "envoyeur" de l'en-tête PKI pour certains messages.

7. Lorsque survient une ambiguïté due aux dénominations des champs, le profil les nomme en utilisant une motation à "point" (par exemple, "certTemplate.subject" signifie le champ sujet au sein d'un champ appelé certTemplate).

8. Lorsque une "SÉQUENCE DE types" fait partie d'un message, une motation de matrice de base zéro est utilisée pour décrire les champs au sein de la SÉQUENCE DE (par exemple, crm[0].certReq.certTemplate.subject se réfère à un sous champs du premier CertReqMsg contenu dans un message de demande).

9. Tous les échanges de messages de PKI dans les appendice de D.4 à D.6 exigent qu'un message certConf soit envoyé par l'entité initiatrice et qu'un PKIConfirm soit envoyé par l'entité qui répond. Le PKIConfirm n'est pas inclus dans certains des profils donnés car son corps est NULL et le contenu de son en-tête est clair d'après le contexte. Tout moyen authentifié peut être utilisé pour le protectionAlg (par exemple, MAC fondé sur le mot de passe si des informations de secret partagé sont connues, ou signature).

D.2 Profil d'utilisation d'algorithme

Le tableau qui suit contient les définitions des utilisations d'algorithmes au sein des protocoles de gestion PKI. Les colonnes du tableau sont :
Nom : un identifiant utilisé pour les profils de message
Utilisation : description de où et pourquoi l'algorithme est utilisé
Obligatoire : identifiant d'algorithme qui DOIT être pris en charge par les mises en œuvre conformes
Autres : alternatives à l'identifiant d'algorithme obligatoire

Nom	Utilisation	Obligatoire	Autres
MSG_SIG_ALG	Protection des messages PKI avec signature	DSA/SHA-1	RSA/MD5, ECDSA, ...
MSG_MAC_ALG	protection des messages PKI avec MAC	PasswordBasedMac	HMAC, X9.9...
SYM_PENC_ALG	chiffrement symétrique de la clé privée d'une entité d'extrémité où la clé est distribuée hors bande	3-DES (3-key-EDE, mode CBC)	AES,RC5, CAST-128...
PROT_ENC_ALG	algorithme asymétrique utilisé pour le chiffrement (clés symétriques pour le chiffrement de) de clés privées transportées dans les messages de PKI	D-H	RSA, ECDH, ...
PROT_SYM_ALG	algorithme de chiffrement symétrique utilisé pour le chiffrement des bits de clé privée (une clé de ce type est chiffrée en utilisant PROT_ENC_ALG)	3-DES (3-key-EDE, mode CBC)	AES,RC5, CAST-128...

Identifiants d'algorithme obligatoire et spécifications :

DSA/SHA-1 : AlgId : {1 2 840 10040 4 3};
Digital Signature Standard [FIPS-186]
Taille du module public : 1024 bits.

PasswordBasedMac : AlgId : {1 2 840 113533 7 66 13}, avec SHA-1 {1 3 14 3 2 26} comme le paramètre owf et HMAC-SHA1 {1 3 6 1 5 5 8 1 2} comme le paramètre mac ; (cette spécification), ainsi que Secure Hash Standard [FIPS-180] et la [RFC2104]
Taille de la clé HMAC : 160 bits (c'est-à-dire, "K" = "H" au paragraphe 5.1.3.1, "Informations de secret partagé")

3-DES : AlgId : {1 2 840 113549 3 7} ; (utilisé dans BSAFE de RSA et dans S/MIME).

D-H : AlgId : {1 2 840 10046 2 1} ; [ANSI-X9.42]
Taille du module public : 1024 bits.

```
DomainParameters ::= SEQUENCE {
      p      ENTIER,                    -- premier impair, p=jq +1
      g      ENTIER,                    -- générateur, g^q = 1 mod p
      q      ENTIER,                    -- facteur premier de p-1
      j      ENTIER FACULTATIF,         -- cofacteur, j>=2
      validationParms  ValidationParms FACULTATIF
  }
ValidationParms ::= SEQUENCE {
    germe       CHAÎNE BINAIRE,         -- germe pour la génération de nombre premier
    pGenCounter   ENTIER                -- vérification de paramètre
  }
```

D.3 Profil de preuve de possession

Champs de POP à utiliser (dans le champ signature du champ de pop de la structure ProofOfPossession) quand on prouve la possession d'une clé privée de signature qui correspond à une cle publique de vérification pour laquelle un certificat a été demandé.

Champ	Valeur	Commentaire
algorithmIdentifier	MSG_SIG_ALG	seule la protection de la signature est permise pour cette preuve
signature	présente	bits calculés en utilisant MSG_SIG_ALG

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

Toutes les CA/RA ne font pas la preuve de possession (de clé de signature, de clé de déchiffrement, ou de clé d'accord de clé) dans le protocole PKIX-CMP de demande de certification dans la bande (comment la POP est faite PEUT en fin de compte être une question de politique qui est rendue explicite pour toute CA dans son OID de politique publié et dans sa déclaration de pratique de certification). Cependant, la présente spécification déclare que les entités de CA/RA DOIVENT faire la POP (par un moyen quelconque) au titre du processus de certification. Toutes les entités d'extrémité DOIVENT être prêtes à fournir la POP (c'est-à-dire, les composants du protocole PKIX-CMP DOIVENT être pris en charge).

D.4 Enregistrement/certification initial (schéma d'authentification de base)

Une entité d'extrémité (non initialisée) demande un (premier) certificat à une CA. Quand la CA répond par un message contenant un certificat, l'entité d'extrémité réplique avec une confirmation de certificat. La CA renvoie une PKIConfirm, et ferme la transaction. Tous les messages sont authentifiés.

Ce schéma permet à l'entité d'extrémité de demander la certification d'une clé publique générée en local (normalement une clé de signature). L'entité d'extrémité PEUT aussi choisir de demander la génération et la certification centralisée d'une autre paire de clés (normalement, une paire de clés de chiffrement).

La certification ne peut être demandée que pour une clé publique générée en local (pour plus, il faut utiliser des messages PKI séparés).

L'entité d'extrémité DOIT prendre en charge la preuve de possession de la clé privée associée à la clé publique générée en local.

Préconditions :
1. L'entité d'extrémité peut authentifier la signature de la CA sur la base de moyens hors bande
2. L'entité d'extrémité et la CA partagent une clé symétrique à MAC,

Flux de messages :

n° d'étape	Entité d'extrémité		PKI
1	formate ir		
2		-> ir ->	
3			traite ir
4			formate ip
5		<- ip <-	
6	traite ip		
7	formate certConf		
8		-> certConf ->	
9			traite certConf
10			formate PKIConf
11		<- PKIConf <-	
12	traite PKIConf		

Pour ce profil, l'entité d'extrémité DOIT inclure tous les (c'est-à-dire, un ou deux) CertReqMsg dans un seul message de PKI, le CA PKI DOIT produire un seul message de réponse PKI qui contient la réponse complète (c'est-à-dire, incluant la seconde paire de clés FACULTATIVE, si elle etait demandée et si la génération de clé centralisée est prise en charge). Pour simplifier, on exige aussi que ce message DOIT être le message final (c'est-à-dire, sans utiliser la valeur d'état "attente").

L'entité d'extrémité a une interaction hors bande avec la CA/RA. Cette transaction établit le secret partagé, le numéro de référence et FACULTATIVEMENT le nom distinctif utilisé pour les deux noms d'envoyeur et de sujet dans le gabarit de certificat. Il est RECOMMANDÉ que le secret partagé fasse au moins 12 caractères.

Demande d'initialisation -- ir

Champ	Valeur	Commentaire
recipient	CA name	nom de la CA à qui on demande de produire un certificat.
protectionAlg	MSG_MAC_ALG	seule la protection par MAC est permise pour cette demande sur la base de la clé d'authentification initiale.
senderKID	referenceNum	numéro de référence que la CA a précédemment produit à l'entité d'extrémité (avec la clé à MAC).
transactionID	présent	valeur spécifique de la mise en œuvre, significative pour l'entité d'extrémité. [Si elle est déjà utilisée à la CA, un message rejet DOIT être produit par la CA]

senderNonce	présent	128 bits (pseudo-)aléatoires
freeText	toute valeur valide	
body	ir (CertReqMessages)	seuls un ou deux CertReqMsg sont permis ; si plus de certificats sont nécessaires, les demandes DOIVENT être dans des messages PKI séparés.
CertReqMsg	un ou deux présents	voir les détails ci-dessous.

Note : crm[0] signifie le premier (qui DOIT être présent), crm[1] signifie le second (qui est FACULTATIF, et est utilisé pour demander une clé générée centralement).

crm[0].certReq.certReqId	valeur fixe de zéro	c'est l'indice du gabarit au sein du message
crm[0].certReqcertTemplate	présent	DOIT inclure la valeur de clé publique sujette, autrement sans contrainte.
crm[0].pop... OPOSigningKey	facultativement présent	si la clé publique provenant de crm[0].certReq.certTemplate est une clé de signature -- une preuve de possession PEUT être requise dans cet échange (voir les détails à l'Appendice D.3).
crm[0].certReq.controls.archiveOptions	facultativement présent	l'entité d'extrémité PEUT demander que la clé privé générée en local soit archivée.
crm[0].certReq.controls.publicationInfo	facultativement présent	l'entité d'extrémité PEUT demander la publication du certificat résultant.
crm[1].certReqcertReqId	valeur fixe de un	indice du gabarit au sein du message
crm[1].certReqcertTemplate	présent	NE DOIT PAS inclure de bits réels de clé publique, autrement sans contrainte (par exemple, les noms peuvent ne pas être les mêmes que dans crm[0]). Noter que subjectPublicKeyInfo PEUT être présent et contenir un identifiant d'algorithme suivi par une chaîne binaire de longueur zéro pour la clé publique sujette si on souhaite informer la CA/RA des préférences d'algorithme et de paramètres concernant la paire de clésà générer.
crm[1].certReq.controls.protocolEncrKey	présent	[l'identifiant d'objet DOIT être PROT_ENC_ALG]. Si la génération de clé centralisée est prise en charge par cette CA, cette clé de chiffrement asymétrique à court terme (générée par l'entité d'extrémité) sera utilisée par la CA pour chiffrer (une clé symétrique utilisée pour chiffrer) une clé privée générée par la CA au nom de l'entité d'extrémité.
crm[1].certReq.controls.archiveOptions	facultativement présent	
crm[1].certReq.controls.publicationInfo	facultativement présent	
protection	présent	bits calculés en utilisant MSG_MAC_ALG

Réponse d'initialisation -- ip

Champ	**Valeur**	**Commentaire**
sender	nom de CA	nom de la CA qui a produit le message.
messageTime	présent	heure à laquelle la CA a produit le message.
protectionAlg	MSG_MAC_ALG	seule la protection par MAC est permise pour cette réponse.
senderKID	referenceNum	numéro de référence que la CA a produit précédemment à l'entité d'extrémité (avec la clé de MAC).
transactionID	présent	valeur provenant du message ir correspondant.
senderNonce	présent	128 bits (pseudo-)aléatoires.
recipNonce	présent	valeur provenant du nom occasionnel d'envoyeur dans le message ir correspondant
freeText	toute valeur valide	
body	ip (CertRepMessage)	contient exactement une réponse pour chaque demande.

La PKI (CA) répond à une demande (ou deux) comme approprié. crc[0] note la première (toujours présente) ; crc[1] note la seconde (seulement présente si le message ir contenait deux demandes et si la CA accepte la génération de clé centralisée).

crc[0].certReqId	valeur fixe de zéro	DOIT contenir la réponse à la première demande dans le message ir correspondant.
crc[0].status.status	présent	valeurs positives permises : "accepté", "accordéAvecModifs" ; valeurs négatives admises : "rejet".
crc[0].status. failInfo	présent	si et seulement si crc[0].status.status est "rejet".
crc[0].certifiedKeyPair	présent	si et seulement si crc[0].status.status est "accepté" ou "accordéAvecModifs"
certificate	présent	sauf si la clé publique de l'entité d'extrémité est une clé de chiffrement et que la POP est faite dans cet échange dans la bande.
encryptedCert	présent	si et seulement si la clé publique de l'entité d'extrémité est une clé de chiffrement et que la POP est faite dans cet échange dans la bande.
publicationInfo	facultativement présent	indique où le certificat a été publié (présent à la discrétion de la CA).
crc[1].certReqId status	valeur fixe de un	DOIT contenir la réponse à la seconde demande dans le message ir correspondant.

crc[1].status.	présent,	valeurs positives admises : "accepté", "accordéAvecModifs" ; valeurs négatives permises : "rejet".
crc[1].status. failInfo	présent	si et seulement si crc[0].status.status est "rejet".
crc[1]. certifiedKeyPair	présent	si et seulement si crc[0].status.status est "accepté" ou "accordéAvecModifs"
certificate	présent	
privateKey	présent	voir à l'Appendice C, les éclaircissements sur le comportement quant au message de demande
publicationInfo	facultativement présent	indique où le certificat a été publié (présent à la discrétion de la CA).
protection	présent	bits calculés en utilisant MSG_MAC_ALG.
extraCerts	facultativement présent	la CA PEUT fournir des certificats supplémentaires à l'entité d'extrémité.

Certificate confirm ; certConf

Champ	**Valeur**	**Commentaire**
sender	présent	le même que dans ir.
recipient	nom de CA	nom de la CA à qui on a demandé de produire un certificat.
transactionID	présent	valeur provenant des messages ir et ip correspondants.
senderNonce	présent	128 bits (pseudo-)aléatoires.
recipNonce	présent	valeur du senderNonce dans le message ip correspondant.
protectionAlg	MSG_MAC_ALG	seule la protection par MAC est permise pour ce message. Le MAC se fonde sur la clé initiale d'authentification partagée entre l'EE et la CA.
senderKID	referenceNum	numéro de référence que la CA a fourni précédemment à l'entité d'extrémité (avec la clé de MAC).
body	certConf	voir au paragraphe 5.3.18, "Contenu de confirmation PKI" le contenu des champs certConf . Note : deux structures CertStatus sont requises si les deux certificats de chiffrement et de signature ont été envoyés.
protection	présent	bits calculés en utilisant MSG_MAC_ALG

Confirmation ; PKIConf

Champ	**Valeur**	**Commentaire**
sender	présent	même que dans ip.
recipient	présent	nom d'envoyeur provenant de certConf.
transactionID	présent	valeur provenant du message certConf.
senderNonce	présent	128 bits (pseudo-)aléatoires.
recipNonce	présent	valeur du senderNonce provenant du message certConf.
protectionAlg	MSG_MAC_ALG	seule la protection par MAC est permisr pour ce message.
senderKID	referenceNum	
body	PKIConf	
protection	présent	bits calculés en utilisant MSG_MAC_ALG.

D.5 Demande de certificat

Une entité d'extrémité (initialisée) demande un certificat à une CA (pour une raison quelconque). Quand la CA répond avec un message contenant un certificat, l'entité d'extrémité réplique avec une confirmation de certificat. La CA répond avec une PKIConfirm, pour clore la transaction. Tous les messages sont authentifiés.

Le profil pour cet échange est identique à celui donné en Appendice D.4, avec les exceptions suivantes:

- o le nom de l'envoyeur DEVRAIT être présent ;
- o l'algorithme de protection de MSG_SIG_ALG DOIT être pris en charge (MSG_MAC_ALG PEUT aussi être pris en charge) dans les messages de demande, réponse, certConfirm, et PKIConfirm ;
- o senderKID et recipKID ne sont présents que si c'est exigé pour la vérification de message ;
- o le corps est cr ou cp ;
- o le corps peut contenir une ou deux structures CertReqMsg, mais l'une ou l'autre CertReqMsg peut être utilisée pour demander la certification d'une clé publique générée localement ou d'une clé publique génére centralement (c'est-à-dire, l'exigence de dépendance à la position de l'Appendice D.4 est supprimée) ;
- o les bits de protection sont calculés conformément au champ protectionAlg.

D.6 Demande de mise à jour de clé

Une entité d'extrémité (initialisée) demande un certificat à une CA (pour mettre à jour une paire de clés et/ou le certificat correspondant qu'elle possède déjà). Quand la CA répond par un message contenant un certificat, l'entité d'extrémité réplique par une confirmation de certificat. La CA répond avec un PKIConfirm, pour clore la transaction. Tous les messages sont authentifiés.

Le profil pour cet échange est identique à celui donné dans l'Appendice D.4, avec les exceptions suivantes :
1. le nom de l'envoyeur DEVRAIT être présent ;
2. l'algorithme de protection de MSG_SIG_ALG DOIT être pris en charge (MSG_MAC_ALG PEUT aussi être pris en charge) dans les messages de demande, réponse, certConfirm, et PKIConfirm ;
3. senderKID et recipKID ne sont présents que si c'est exigé pour la vérification du message ;
4. le corps est kur ou kup ;
5. le corps peut contenir une ou deux stuctures CertReqMsg, mais l'une ou l'autre CertReqMsg peut être utilisée pour demander la certification d'une clé publique générée localement ou d'une générée centralement (c'est-à-dire, l'exigence de dépendance à la position de l'Appendice D.4 est supprimée) ;
6. les bits de protection sont calculés conformément au champ protectionAlg ;
7. regCtrl OldCertId DEVRAIT être utilisé (sauf si il est clair pour l'envoyeur et le receveur -- par des moyens non spécifiés dans le présent document -- qu'il n'est pas nécessaire).

Appendice E. Profils (FACULTATIFS) de message de gestion de PKI

Cet Appendice contient les profils détaillés pour les messages PKI qui PEUVENT être pris en charge par les mises en œuvre (en plus des messages qui DOIVENT être pris en charge ; voir la Section 6 et l'Appendice D).

Les profils pour les messages de PKI utilisés dans les opérations de gestion de PKI suivantes fournissent :
- o la mise à jour de clé de CA ;
- o les informations de demande/réponse ;
- o la demande/réponse de certification croisée (unidirectionnelle) ;
- o l'initialisation dans la bande en utilisant un certificat d'identité externe.

Des versions ultérieures du présent document pourront étendre cela afin d'inclure des profils pour les opérations dont la liste figure ci-dessous (ainsi que d'autres opérations, si désiré).
- o demande de révocation,
- o publication de certificat,
- o publication de CRL.

E.1 Règles générales pour l'interprétation de ces profils

Identiques à celles de l'Appendice D.1.

E.2 Profil d'utilisation d'algorithme

Identique à celui de l'Appendice D.2.

E.3 Certificats auto signés

Profile comment une structure Certificate peut être "auto signée". Ces structures sont utilisées pour la distribution des clés publiques de CA. Cela peut se produire de trois façons (voir au paragraphe 4.4 la description de l'utilisation de ces structures) :

Type	Fonction
newWithNew	un vrai certificat "auto signé" ; la clé publique contenue DOIT être utilisable pour vérifier la signature (bien que cela n'assure que l'intégrité et aucune authentification).
oldWithNew	clé publique de CA racine précédente signée avec la nouvelle clé privée.
newWithOld	nouvelle clé publique de CA racine signée avec la clé privée précédente.

De tels certificats (incluant les extensions pertinentes) doivent contenir des valeurs "sensées" pour tous les champs. Par exemple, lorsque présent, subjectAltName DOIT être identique à issuerAltName, et, lorsque présent, keyIdentifiers doit contenir les valeurs appropriées, et cetera.

E.4 Mise à jour de clé de CA racine

Une CA racine met à jour sa paire de clés. Elle produit ensuite un message d'annonce de mise à jour de clé de CA qui peut être mis à la disposition (via un moyen de transport) des entités d'extrémité pertinentes. Un message de confirmation N'EST PAS EXIGÉ de la part des entités d'extrémité.

Message ckuann :

Champ	Valeur	Commentaire
envoyeur	nom de CA	nom de la CA
corps	ckuann	(CAKeyUpdAnnContent)
oldWithNew	présent	voir l'Appendice E.3
newWithOld	présent	voir l'Appendice E.3
newWithNew	présent	voir l'Appendice E.3
extraCerts	présence facultative	peut être utilisé pour "publier" les certificats (par exemple, les certificats signés en utilisant la nouvelle clé privée).

E.5 Demande/réponse d'informations de PKI

L'entité d'extrémité envoie un message général à la PKI demandant des détails qui vont être nécessaires pour les opérations ultérieures de gestion de PKI. La RA/CA répond avec une réponse générale. Si une RA génère la réponse, elle va alors simplement transmettre le message équivalent qu'elle a précédemment reçu de la CA, avec l'ajout possible de certificats aux champs extraCerts du message de PKI. Un message de confirmation N'EST PAS EXIGÉ de l'entité d'extrémité.

Flux de messages :

N° d'étape	Entité d'extrémité			PKI
1	formate genm			
2		->	genm ->	
3				traite genm
4				produit genp
5		<-	genp <-	
6	traite genp			

genM :

Champ	Valeur	Commentaire
receveur	nom de CA	nom de la CA comme contenu dans les extensions issuerAltName ou les champs "issuer" au sein des certificats.
protectionAlg	MSG_MAC_ALG ou MSG_SIG_ALG	tout algorithme de protection authentifié.
SenderKID	présent si requis	doit être present si requis pour la vérification de la protection du message.
freeText	toute valeur valide	
corps	genr (GenReqContent)	
GenMsgContent	SEQUENCE vide	toutes les informations pertinentes demandées.
protection	présent	bits calculés en utilisant MSG_MAC_ALG ou MSG_SIG_ALG.

genP :

Champ	Valeur	Commentaire
envoyeur	nom de CA	nom de la CA qui a produit le message.
protectionAlg	MSG_MAC_ALG ou MSG_SIG_ALG	tout algorithme de protection authentifié.
senderKID	présent si exigé	doit être présent si c'est exigé pour la vérification de la protection du message.
corps	genp	(GenRepContent)
CAProtEncCert	présent	(identifiant d'objet d'un des PROT_ENC_ALG) avec la valeur pertinente pour être utilisé si l'entité d'extrémité a besoin de chiffrer les informations pour la CA (par exemple, une clé privée pour les besoins de récupération).
SignKeyPairTypes	présent, avec la valeur pertinente	ensemble des identifiants d'algorithme de signature que cette CA va certifier pour clés publiques sujettes.
EncKeyPairTypes	présent, avec la valeur pertinente	ensemble des identitifants d'algorithme de chiffrement/accord de clé que cette CA va certifier pour les clés publiques sujettes.

PreferredSymmAlg	présent	(un identifiant d'objet de PROT_SYM_ALG) avec la valeur pertinente ; l'algorithmr symétrique que cette CA s'attend à être utilisé dans les messages de PKI ultérieurs (pour chiffrement).
CAKeyUpdateInfo	présent, avec la valeur pertinente	la CA PEUT fournir des informations sur une paire de clés de CA racine pertinente en utilisant ce champ (noter que cela n'implique pas que la CA qui répond soit la CA racine en question).
CurrentCRL	présent, avec la valeur pertinente	la CA PEUT fournir une copie d'une CRL complète (c'est-à-dire, la plus complète possible).
protection	présent	bits calculés en utilisant MSG_MAC_ALG ou MSG_SIG_ALG.
extraCerts	présence facultative	peut être utilisé pour envoyer des certificats à l'entité d'extrémité. Une RA PEUT ajouter son certificat ici.

E.6 Demande/réponse de certification croisée (unidirectionnelle)

La création d'un seul certificat croisé (c'est-à-dire, pas deux à la fois). La CA demandeuse PEUT choisir qui est responsable de la publication du certificat croisé créé par la CA qui répond par l'utilisation de la commande PKIPublicationInfo.

Préconditions :

1. La CA qui répond peut vérifier l'origine de la demande (éventuellement en recourant à des moyens hors bande) avant de traiter la demande.
2. La CA demandeuse peut authentifier l'origine de la réponse (éventuellement en recourant à des moyens hors bande) avant de traiter la réponse.

L'utilisation de la confirmation de certificat et la confirmation correspondante du serveur est déterminée par le champ generalInfo dans l'en-tête de PKI (voir au paragraphe 5.1.1). Le profil qui suit ne rend pas obligatoire la prise en charge de l'une ou l'autre confirmation.

Flux de messages :

N° d'étape	CA demandeuse		CA répondeuse
1	formate ccr		
2		-> ccr ->	
3			traite ccr
4			produit ccp
5		<- ccp <-	
6	traite ccp		

ccr :

Champ	Valeur	Commentaire
sender	nom de CA demandeuse	nom de la CA qui produit le message
recipient	nom de CA qui répond	nom de la CA à qui on demande de produire un certificat.
messageTime	heure de production du message	heure courante à la CA demandeuse.
protectionAlg	MSG_SIG_ALG	seule la protection de signature es permise pour cette demande.
senderKID	présent si exigé	doit être présent si exigé pour la vérification de la protection de message.
recipKID	présent si exigé	doit être présent si exigé pour la vérification de la protection de message.
transactionID	présent	valeur spécifique de la mise en œuvre, significative pour la CA qui demande. [si déjà utilisée à la CA qui répond, un message de rejet DOIT être produit par la CA qui réspond].
senderNonce	présent	128 bits (pseudo-)aléatoires
freeText	toute valeur valide	
body	ccr (CertReqMessages) un seul CertReqMsg permis; si plusieurs certificats croisés sont exigée, ils DOIVENT alors être dans des messages PKI séparés.	
certTemplate	présent	les détails suivent
version	v1 ou v3	la v3 est fortement RECOMMANDÉE.
signingAlg	présent	La CA demandeuse doit savoir à l'avance quel algorithme de hachage elle souhaite pour la signature du certificat.
subject	présent	ne peut être NULL-DN que si une valeur d'extension de subjectAltNames est proposée.
validity	présent	DOIT être complètement spécifié (c'est-à-dire, les deux champs présents)
issuer	présent	ne peut être NULL-DN que si une valeur d'extension de issuerAltNames est proposée.
publicKey	présent	La clé à certifier (qui doit être pour un algorithme de signature).

extensions	présence facultative	La CA demandeuse doit proposer des valeurs pour toutes les extensions dont elle exige la présence dans le certificat croisé.
POPOSigningKey	présent	voir à l'Appendice D3 le profil de preuve de possession.
protection	présent	bits calculés en utilisant MSG_SIG_ALG.
extraCerts	présence facultative	PEUT contenir tout certificat supplémentaire que le demandeur souhaite inclure.

ccp :

Champ	**Valeur**	**Commentaire**
sender	nom de CA qui répond	le nom de la CA qui a produit le message.
recipient	nom de CA demndeuse	nom de la CA qui a demandé la production d'un certificat.
messageTime	heure de production du message	heure courant à la CA qui répond.
protectionAlg	MSG_SIG_ALG	seule la protection de signature est permise pour ce message.
senderKID	présent si exigé	doit être présent si exigé pour la vérification de la protection de message.
recipKID	présent si exigé	
transactionID	présent	-- valeur provenant du message ccr correspondant.
senderNonce	présent	-- 128 bits (pseudo-)aléatoires.
recipNonce	présent	-- senderNonce provenant du message ccr correspondant.
freeText	toute valeur valide	
body	ccp (CertRepMessage)	une seule CertResponse permise ; si plusieurs certificats croisés sont requis, ils DOIVENT être dans des messages PKI séparés.
response	présent	
status	présent	
PKIStatusInfo.status	présent	si PKIStatusInfo.status est "accepté" ou "accordéAvecModifs", certifiedKeyPair DOIT être présent et failInfo DOIT être absent.
failInfo	présent selon PKIStatusInfo.status	si PKIStatusInfo.status est "rejet", certifiedKeyPair DOIT être absent et failInfo DOIT être présent et contenir les réglages de bits appropriés.
certifiedKeyPair	présent selon PKIStatusInfo.status	
certificate	présent selon certifiedKeyPair	le contenu réel du certificat doit être examiné par la CA demandeuse avant publication
protection	présent	bits calculés en utilisant MSG_SIG_ALG.
extraCerts	présence facultative	PEUT contenir tout certificat supplémentaire que celui qui répond souhaite inclure.

E.7 Initialisation dans la bande en utilisant un certificat d'identité externe

Une entité d'extrémité (non initialisée) souhaite initialiser la PKI avec une CA, CA-1. Elle utilise, pour les besoins de l'authentification, un certificat d'identité pré-existant produit par une autre CA (externe), CA-X. Une relation de confiance doit avoir déjà été établie entre CA-1 et CA-X afin que CA-1 puisse valider le certificat d'identité de l'EE signé par CA-X. De plus, un mécanisme doit déjà avoir été établi au sein de l'environnement personnel de sécurité (PSE, *Personal Security Environment*) de l'EE qui va lui permettre d'authentifier et vérifier les messages de PKI signés par CA-1 (par exemple, le PSE peut contenir un certificat produit pour la clé publique de CA-1, signée par une autre CA mais que l'EE estime de confiance sur la base de techniques d'authentification hors bande).

L'EE envoie une demande d'initialisation pour commencer la transaction. Quand CA-1 répond avec un message contenant le nouveau certificat, l'entité d'extrémité répond avec une confirmation de certificat. CA-1 répond avec un PKIConfirm pour clore la transaction. Tous les messages sont signés (les messages de l'EE sont signés en utilisant la clé privée qui correspond à la clé publique dans son certificat d'identité externe ; les messages de CA-1 sont signés en utilisant la clé privée qui correspond à la clé publique dans un certificat qui peut être suivi jusqu'à une ancre de confiance dans le PSE de l'EE).

Le profil pour cet échange est identique à celui donné dans l'Appendice D.4, avec les exceptions suivantes :

- l'EE et la CA-1 ne partagent pas une clé symétrique à MAC (c'est-à-dire, il n'y a pas d'information de secret partagé hors bande entre ces entités) ;
- le nom d'envoyeur dans ir DOIT être présent (et identique au nom sujet présent dans le certificat externe d'identité) ;
- protectionAlg de MSG_SIG_ALG DOIT être utilisé dans tous les messages ;
- le certificat externe d'identité DOIT être porté dans le champ extraCerts de l'ir ;
- senderKID et recipKID ne sont pas utilisés ;
- le corps est ir ou ip ;
- les bits de protection sont calculés conformément au champ protectionAlg.

Appendice F. Définitions ASN.1 compilables

PKIXCMP {iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0) id-mod-cmp2000(16)}

ÉTIQUETTES EXPLICITES DE DÉFINITIONS ::=

DÉBUT

-- EXPORTE TOUT --

IMPORTE

Certificate, CertificateList, Extensions, AlgorithmIdentifier,
UTF8String -- si exigé ; autrement, commenter
À PARTIR DE PKIX1Explicit88 {iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0) id-pkix1-explicit-88(1)}

GeneralName, KeyIdentifier
À PARTIR DE PKIX1Implicit88 {iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0) id-pkix1-implicit-88(2)}

CertTemplate, PKIPublicationInfo, EncryptedValue, CertId, CertReqMessages
À PARTIR DE PKIXCRMF-2005 {iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0) id-mod-crmf2005(36)}

-- voir aussi les précisions de comportement à CRMF codifiées dans l'Appendice C de cette spécification.

CertificationRequest
À PARTIR DE PKCS-10 {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-10(10) modules(1) pkcs-10(1)}

-- (spécifié dans la RFC 2986 avec la syntaxe ASN.1 1993 et des étiquettes IMPLICITES). Autrement, les mises en œuves peuvent directement inclure la syntaxe de la[RFC2986] dans ce module ; le reste du module contient les OID et constructions définis en local.

```
CMPCertificate ::= CHOIX {
     x509v3PKCert        Certificate
   }
```

-- Cette syntaxe, bien que compatible au bit près avec la définition de la norme X.509 de "Certificate", permet la possibilité de futurs types de certificat (comme les certificats d'attribut de X.509, les certificats WTLS de WAP, ou d'autres sortes de certificats) au sein de ce protocole de gestion de certificats, si le besoin se faisait sentir de prendre en charge une telle généralité. Les mises en œuvre qui ne prévoient pas le besoin de prendre un jour en charge d'autres types de certificats PEUVENT, si elles le souhaitent, ignorer la structure ci-dessus, et "ôter les commentaires" sur la suivante avant de compiler ce module ASN.1. (Noter que l'interopérabilité avec les mises en œuvre qui ne le font pas ne sera pas affectée par ce changement.)

```
-- CMPCertificate ::= Certificate

PKIMessage ::= SEQUENCE {
   header           PKIHeader,
   body             PKIBody,
   protection   [0] PKIProtection FACULTATIF,
   extraCerts   [1] SEQUENCE TAILLE (1..MAX) DE CMPCertificate FACULTATIF
}

PKIMessages ::= SEQUENCE TAILLE (1..MAX) DE PKIMessage

PKIHeader ::= SEQUENCE {
   pvno                ENTIER     { cmp1999(1), cmp2000(2) },
   sender              GeneralName,       -- identifie l'envoyeur
   recipient           GeneralName,       -- identifie le receveur prévu
   messageTime     [0] GeneralizedTime   FACULTATIF,
```

```
-- heure de production de ce message (utilisé quand l'envoyeur pense que le transport conviendra ; c'est-à-dire, que l'heure
     aura toujours un sens à la réceptiont)
   protectionAlg  [1] AlgorithmIdentifier   FACULTATIF,        -- algorithme utilisé pour calculer les bits de protection.
   senderKID      [2] KeyIdentifier  FACULTATIF,
   recipKID       [3] KeyIdentifier   FACULTATIF,          -- pour identifier les clés spécifiques utilisées pour la protection
   transactionID  [4] CHAÎNE D'OCTETS    FACULTATIF,
-- identifie la transaction; c'est-à-dire, ce sera le même dans les messages correspondants de demande, réponse, certConf, et
     PKIConf.
   senderNonce   [5] CHAÎNE D'OCTETS  FACULTATIF,
   recipNonce     [6] CHAÎNE D'OCTETS  FACULTATIF,
-- noms occasionnels utilisés pour fournir la protection contre la répétition, senderNonce est inséré par le créateur de ce
     message; recipNonce est un nom occasionnel précédemment inséré dans un message en rapport par le receveur auquel
     ce message est destiné
   freeText       [7] PKIFreeText            FACULTATIF,
-- ce peut être utilisé pour indiquer des instructions spécifiques du contexte (ce champ est destiné à l'homme).
   generalInfo    [8] SEQUENCE TAILLE (1..MAX) DE InfoTypeAndValue     FACULTATIF
-- ce peut être utilisé pour porter des informations spécifiques du contexte (ce champ n'est pas principalement destiné à la
     consommation humaine)
 }

PKIFreeText ::= SEQUENCE TAILLE (1..MAX) DE UTF8String
-- texte codé comme chaîne UTF-8 [RFC3629] (note : chaque UTF8String PEUT inclure une étiquette de langue
     [RFC3066] pour indiquer le langage du texte contenu ; voir les détails dans la [RFC2482]).

PKIBody ::= CHOIX {                          -- éléments de corps spécifiques du message.
   ir          [0]  CertReqMessages,         -- demande d'iitialisation.
   ip          [1]  CertRepMessage,          -- réponse d'initialisation.
   cr          [2]  CertReqMessages,         -- demande de certification.
   cp          [3]  CertRepMessage,          -- réponse de certification.
   p10cr       [4]  CertificationRequest,    -- importé de la [RFC2986]
   popdecc     [5]  POPODecKeyChallContent,  -- défi de preuve de possession.
   popdecr     [6]  POPODecKeyRespContent,   --réponse de preuve de possession.
   kur         [7]  CertReqMessages,         -- demande de mise à jour de clé.
   kup         [8]  CertRepMessage,          --réponse de mise à jour de clé.
   krr         [9]  CertReqMessages,         -- demande de récupération de clé.
   krp        [10] KeyRecRepContent,         -- réponse de récupération de clé.
   rr         [11] RevReqContent,            -- demande de révocation.
   rp         [12] RevRepContent,            -- réponse de révocation.
   ccr        [13] CertReqMessages,          -- demande de certification croisée.
   ccp        [14] CertRepMessage,           -- réponse de certification croisée.
   ckuann     [15] CAKeyUpdAnnContent,       -- annonce de mise à jour de clé de CA.
   cann       [16] CertAnnContent,           -- annonce de certificat.
   rann       [17] RevAnnContent,            -- annonce de révocation.
   crlann     [18] CRLAnnContent,            -- annonce de CRL.
   pkiconf    [19] PKIConfirmContent,        -- confirmation
   nested     [20] NestedMessageContent,     -- message incorporé.
   genm       [21] GenMsgContent,            -- message général.
   genp       [22] GenRepContent,            -- réponse générale.
   error      [23] ErrorMsgContent,          -- message d'erreur.
   certConf   [24] CertConfirmContent,       -- confirmation de certificat.
   pollReq    [25] PollReqContent,           -- demande d'interrogation.
   pollRep    [26] PollRepContent            -- réponse d'interrogation.
}

PKIProtection ::= CHAÎNE BINAIRE

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

IDENTIFIANT D'OBJET   id-PasswordBasedMac ::= {1 2 840 113533 7 66 13}
PBMParameter ::= SEQUENCE {
```

```
   salt            CHAÎNE D'OCTETS,
-- note :  les mises en œuvre PEUVENT souhaiter limiter les tailles acceptables de cette chaîne à des valeurs appropriées à
     leur environnement afin de réduire le risque d'attaque de déni de service.
   owf             AlgorithmIdentifier,            -- AlgId pour fonction unidirectionnelle (owf) (SHA-1 recommandé).
   iterationCount  ENTIER,                         -- nombre de fois que l'OWF est appliquée.
-- note : les mises en œuvre PEUVENT souhaiter limiter les tailles acceptables de cet entier aux valeurs appropriées à leur
     environnement afin de réduire le risque d'attaque de déni de service.
   mac             AlgorithmIdentifier
-- le MAC AlgId (par exemple, DES-MAC, Triple-DES-MAC [PKCS11], ou HMAC [RFC2104], [RFC2202])
    }

IDENTIFIANT D'OBJET id-DHBasedMac ::= {1 2 840 113533 7 66 30}
DHBMParameter ::= SEQUENCE {
   owf             AlgorithmIdentifier,            -- AlgId pour fonction unidirectionnelle (owf) (SHA-1 recommandé).
   mac             AlgorithmIdentifier    -- le MAC AlgId (par exemple, DES-MAC, Triple-DES-MAC [PKCS11],
 }                                           ou HMAC [RFC2104], [RFC2202])

NestedMessageContent ::= PKIMessages
PKIStatus ::= ENTIER {
   accepted            (0),                -- on a exactement ce qu'on a demandé.
   grantedWithMods     (1),
- on a quelque chose qui ressemble a ce qu'on a demandé ; le demandeur est chargé de s'assurer des différences.
   rejection           (2),                 -- on ne l'a pas, des informations sont ailleurs dans le message.
   waiting             (3),
--la partie corps de la demande n'a pas encore été traitée ; on s'attend à en savoir plus plus tard : le traitement approprié de
cette réponse d'état PEUT utiliser les messages PKI de demande/réponse d'interrogation spécifiés au paragraphe 5.3.22 ;
autrement, une interrogation dans la couche transport sous-jacente PEUT avoir son utilité à cet égard)
   revocationWarning   (4),                 -- ce message contient l'avertissement qu'une révocation est imminente.
   revocationNotification (5),              -- notification qu'une révocation s'est produite.
   keyUpdateWarning    (6)                  -- mise à jour déjà faite pour le oldCertId spécifié dans CertReqMsg.
}

PKIFailureInfo ::= CHAÎNE BINAIRE {
--comme on peut échouer de plus d'une façon, d'autres codes pourront être ajoutés à l'avenir si/quand nécessaire.
   badAlg              (0),      -- identifiant d'algorithme non reconnu ou non pris en charge.
   badMessageCheck     (1), -- échec de vérification d'intégrité (par exemple, la signature ne correspond pas).
   badRequest          (2),      -- transaction non permise ou non prise en charge.
   badTime             (3),  -- l'heure du message n'est pas assez proche de l'heure système, comme défini par la politique locale
   badCertId           (4),          -- aucun certificat ne correspond aux critères fournis.
   badDataFormat       (5),          -- les données soumises ont un mauvais format.
   wrongAuthority      (6),          -- l'autorité indiquée dans la demande est différente de celle qui a créé le jeton de réponse.
   incorrectData       (7),           -- les données du demandeur sont incorrectes (pour les services de notariat).
   missingTimeStamp    (8),           -- quand l'horodatage manque alors qu'ils devrait y être (d'après la politique).
   badPOP              (9),           -- la preuve de possession a échoué.
   certRevoked         (10),          -- le certificat a déjà été révoqué.
   certConfirmed       (11),          -- le certificat a déjà été confirmé.
   wrongIntegrity      (12),          -- intégrité invalide, fondée sur le mot de passe au lieu d'une signature ou vice versa.
   badRecipientNonce   (13),          -- nom occasionnel de receveur invalide, soit manquant, soit mauvaise valeur.
   timeNotAvailable    (14),          -- la source horaire de TSA n'est pas disponible.
   unacceptedPolicy    (15),          -- la politique de TSA demandée n'est pas acceptée par la TSA.
   unacceptedExtension (16),          -- l'extension demandée n'est pas acceptée par la TSA.
   addInfoNotAvailable (17),          -- les informations supplémentaires demandées ne sont pas comprises ou indisponibles.
   badSenderNonce      (18),          -- nom occasionnel d'envoyeur invalide, soit manquant soit mauvaise taille.
   badCertTemplate     (19),          -- gabarit de certificat invalide ou informations obligatoires manquantes.
   signerNotTrusted    (20),          -- le signataire du message est inconnu ou pas de confiance.
   transactionIdInUse  (21),          -- l'identifiant de transaction est déjà utilisé.
   unsupportedVersion  (22),          -- la version du message n'est pas acceptée.
   notAuthorized       (23),          -- l'envoyeur n'est pas autorisé à faire la demande ou à effectuer l'action précédente.
   systemUnavail       (24),          -- la demande ne peut pas être traitée à cause de l'indisponibilité du système.
   systemFailure       (25),          -- la demande ne peut pas être traitée à cause de la défaillance du système.
   duplicateCertReq    (26)           -- le certificat ne peut pas être produit parce qu'un double du certificat existe déjà.
}
```

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

OOBCert ::= CMPCertificate

OOBCertHash ::= SEQUENCE {
   hashAlg     [0] AlgorithmIdentifier     FACULTATIF,
   certId      [1] CertId                  FACULTATIF,
   hashVal       CHAÎNE BINAIRE       -- hashVal est calculé sur le codage en DER du certificat auto signé avec
 }                                          l'identifiant certID.

POPODecKeyChallContent ::= SÉQUENCE DE Challenge
-- Un défi par demande de certification de clé de chiffrement (dans le même ordre que ces demandes apparaissent dans les
     CertReqMessages).

Challenge ::= SEQUENCE {
   owf                 AlgorithmIdentifier  FACULTATIF,
-- DOIT être présent dans le premier défi; PEUT être omis dans tout défi suivant dans POPODecKeyChallContent (si il est
     omis, la owf utilisée dans le défi précédant immédiatement est à utiliser).
   witness             CHAÎNE D'OCTETS,
-- résultat de l'application de la fonction unidirectionnelle (owf) à un ENTIER aléatoire, A. [Noter qu'un ENTIER différent
     DOIT être utilisé pour chaque défi.]
   challenge           CHAÎNE D'OCTETS
-- chiffrement (sous une clé publique pour laquelle la demande de certificat est faite) de Rand, où Rand est spécifié comme
     Rand ::= SEQUENCE {
   --     int      ENTIER,       - l'ENTIER A généré au hasard (ci-dessus)
   --     sender   GeneralName         - le nom de l'envoyeur (comme inclus dans l'en-tête PKI).
   --  }
}

POPODecKeyRespContent ::= SEQUENCE D'ENTIER
-- Un ENTIER par demande de certification de clé de chiffrement (dans le même ordre que celui où les demandes
     apparaissent dans CertReqMessages). L'ENTIER A restitué (ci-dessus) est retourné à l'envoyeur du défi correspondant.

CertRepMessage ::= SEQUENCE {
   caPubs       [1] SEQUENCE TAILLE (1..MAX) DE CMPCertificate   FACULTATIF,
   response         SÉQUENCE DE CertResponse
}

CertResponse ::= SEQUENCE {
   certReqId           ENTIER,
-- pour confronter cette réponse à la demande correspondante (une valeur de -1 est à utiliser si certReqId n'est pas spécifié
     dans la demande correspondante).
   status              PKIStatusInfo,
   certifiedKeyPair    CertifiedKeyPair    FACULTATIF,
   rspInfo             CHAÎNE D'OCTETS       FACULTATIF
-- analogue à la chaîne id-regInfo-utf8Pairs définie pour regInfo dans CertReqMsg [RFC4211].
    }

CertifiedKeyPair ::= SEQUENCE {
   certOrEncCert       CertOrEncCert,
   privateKey      [0] EncryptedValue      FACULTATIF,        -- voir dans la [RFC4211] le commentaire sur le codage.
   publicationInfo [1] PKIPublicationInfo  FACULTATIF
 }

CertOrEncCert ::= CHOIX {
   certificate     [0] CMPCertificate,
   encryptedCert   [1] EncryptedValue
}
```

```
KeyRecRepContent ::= SEQUENCE {
   status                PKIStatusInfo,
   newSigCert      [0] CMPCertificate FACULTATIF,
   caCerts           [1] SEQUENCE TAILLE (1..MAX) DE CMPCertificate FACULTATIF,
   keyPairHist     [2] SEQUENCE TAILLE (1..MAX) DE CertifiedKeyPair FACULTATIF
 }

RevReqContent ::= SÉQUENCE DE RevDetails

RevDetails ::= SEQUENCE {
   certDetails        CertTemplate,
-- permet au demandeur de spécifier ce qu'il veut sur le certificat pour lequel la révocation est demandée (par exemple, dans
     les cas où le numéro de série n'est pas disponible).
   crlEntryDetails     Extensions       FACULTATIF        -- crlEntryExtensions demandées
    }

RevRepContent ::= SEQUENCE {
   status       SEQUENCE TAILLE (1..MAX) DE PKIStatusInfo,         -- dans l'ordre d'envoi dans RevReqContent.
    revCerts [0] SEQUENCE TAILLE (1..MAX) DE CertId FACULTATIF,
-- Identifiants pour lesquels la révocation a été demandee  (même ordre que status).
   crls  [1] SEQUENCE TAILLE (1..MAX) DE CertificateList FACULTATIF-- les CRL résultantes
 }                                                                                    (peuvent être plusieurs)

CAKeyUpdAnnContent ::= SEQUENCE {
   oldWithNew   CMPCertificate,                    -- vieille clé publique signée avec nouvelle clé privée
   newWithOld   CMPCertificate,                    -- nouvelle clé publique signée avec vieille clé priée
   newWithNew   CMPCertificate                     -- nouvelle clé publique signée avec nouvelle clé privée.
}

CertAnnContent ::= CMPCertificate

RevAnnContent ::= SEQUENCE {
   status              PKIStatus,
   certId              CertId,
   willBeRevokedAt     GeneralizedTime,
   badSinceDate        GeneralizedTime,
   crlDetails      Extensions  FACULTATIF  -- détails de CRL supplémentaire (par exemple, numéro, aiason, situation, etc.)
}

CRLAnnContent ::= SÉQUENCE DE CertificateList

CertConfirmContent ::= SÉQUENCE DE CertStatus

CertStatus ::= SEQUENCE {
  certHash    CHAÎNE D'OCTETS,
      -- hachage du certificat, avec le même algorithme qu'utilisé pour créer et vérifier la signature de certificat.
  certReqId   ENTIER,        -- pour confronter cette confirmation avec la demande/réponse correspondante.
      statusInfo  PKIStatusInfo  FACULTATIF
 }

PKIConfirmContent ::= NULL

InfoTypeAndValue ::= SEQUENCE {
   infoType              IDENTIFIANT D'OBJET,
   infoValue             TOUT DÉFINI PAR infoType  FACULTATIF
 }
-- Un exemple de contenu de InfoTypeAndValue inclut, sans s'y limiter, ce qui suit (commentaire dans ce module ASN.1 à
utiliser comme approprié dans un certain environenment) :
    --
    --   id-it-caProtEncCert    IDENTIFIANT D'OBJET ::= {id-it 1}
    --      CAProtEncCertValue      ::= CMPCertificate
    --   id-it-signKeyPairTypes IDENTIFIANT D'OBJET ::= {id-it 2}
    --      SignKeyPairTypesValue   ::= SÉQUENCE DE AlgorithmIdentifier
```

```
    --  id-it-encKeyPairTypes  IDENTIFIANT D'OBJET ::= {id-it 3}
    --     EncKeyPairTypesValue    ::= SÉQUENCE DE AlgorithmIdentifier
    --  id-it-preferredSymmAlg IDENTIFIANT D'OBJET ::= {id-it 4}
    --     PreferredSymmAlgValue   ::= AlgorithmIdentifier
    --  id-it-caKeyUpdateInfo  IDENTIFIANT D'OBJET ::= {id-it 5}
    --     CAKeyUpdateInfoValue    ::= CAKeyUpdAnnContent
    --  id-it-currentCRL       IDENTIFIANT D'OBJET ::= {id-it 6}
    --     CurrentCRLValue         ::= CertificateList
    --  id-it-unsupportedOIDs  IDENTIFIANT D'OBJET ::= {id-it 7}
    --     UnsupportedOIDsValue    ::= SÉQUENCE DE IDENTIFIANT D'OBJET
    --  id-it-keyPairParamReq  IDENTIFIANT D'OBJET ::= {id-it 10}
    --     KeyPairParamReqValue    ::= IDENTIFIANT D'OBJET
    --  id-it-keyPairParamRep  IDENTIFIANT D'OBJET ::= {id-it 11}
    --     KeyPairParamRepValue    ::= AlgorithmIdentifer
    --  id-it-revPassphrase    IDENTIFIANT D'OBJET ::= {id-it 12}
    --     RevPassphraseValue      ::= EncryptedValue
    --  id-it-implicitConfirm  IDENTIFIANT D'OBJET ::= {id-it 13}
    --     ImplicitConfirmValue    ::= NULL
    --  id-it-confirmWaitTime  IDENTIFIANT D'OBJET ::= {id-it 14}
    --     ConfirmWaitTimeValue    ::= GeneralizedTime
    --  id-it-origPKIMessage   IDENTIFIANT D'OBJET ::= {id-it 15}
    --     OrigPKIMessageValue     ::= PKIMessages
    --  id-it-suppLangTags     IDENTIFIANT D'OBJET ::= {id-it 16}
    --     SuppLangTagsValue       ::= SÉQUENCE DE UTF8String
    --
    -- où
    --
     --  id-pkix IDENTIFIANT D'OBJET ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5)
                                                mechanisms(5) pkix(7)}
    -- et
    --  id-it   IDENTIFIANT D'OBJET ::= {id-pkix 4}
    --
    --
-- Cette construction PEUT aussi être utilisée pour définir de nouveaux messages de demande et réponse de protocole de
    gestion de certificat PKIX, ou des messages d'objet général (par exemple, annonce) pour des besoins futurs ou des
    environnements spécifiques.

GenMsgContent ::= SÉQUENCE DE InfoTypeAndValue
-- Peut être envoyé par une EE, RA, ou CA (selon le contenu du message). Le paramètre FACULTATIF infoValue de
    InfoTypeAndValue va normalement être omis pour certains des exemples ci-dessus. Le receveur est libre d'ignorer tout
    identifiant d'objet contenu qu'il ne reconnaît pas. Envoyé de l'EE à la CA, l'ensemble vide indique que la CA peut
    envoyer toutes les informations qu'elle veut.

GenRepContent ::= SÉQUENCE DE InfoTypeAndValue
-- le receveur PEUT ignore tout OID contenu qu'il ne reconnaît pas.

ErrorMsgContent ::= SEQUENCE {
   pKIStatusInfo         PKIStatusInfo,
   errorCode             ENTIER          FACULTATIF,       -- codes d'erreur spécifiques de la mise en œuvre.
   errorDetails          PKIFreeText       FACULTATIF       -- déétals d'ereur spécifiques de la mise en œuvre.
}

PollReqContent ::= SÉQUENCE DE SEQUENCE {
   certReqId             ENTIER
 }

PollRepContent ::= SÉQUENCE DE SEQUENCE {
   certReqId             ENTIER,
   checkAfter            ENTIER,  -- heure en secondes
   reason               PKIFreeText FACULTATIF
}

FIN -- du module de CMP
```

Appendice G. Remerciements

Les auteurs remercient de leurs contributions les divers membres du groupe de travail PKIX de l'IETF et de la liste de diffusion ICSA CA-talk (liste dédiée uniquement à la discussion des efforts d'interopérabilité des CMP). Beaucoup de ces contributions ont précisé et amélioré de façon significative l'utilité de la présente spécification. Tomi Kause remercie Vesa Suontama et Toni Tammisalo de leur relecture et leurs commentaires.

Adresse des auteurs

Carlisle Adams
University of Ottawa
800 King Edward Avenue
P.O.Box 450, Station A
Ottawa, Ontario K1N 6N5
CA
téléphone : (613) 562-5800 ext. 2345
mél : cadams@site.uottawa.ca

Stephen Farrell
Trinity College Dublin
Distributed Systems Group
Computer Science Department
Dublin
IE
téléphone : +353-1-608-2945
mél : stephen.farrell@cs.tcd.ie

Tomi Kause
SSH Communications Security Corp
Valimotie 17
Helsinki 00380
FI
téléphone : +358 20 500 7415
mél : toka@ssh.com

Tero Mononen
SafeNet, Inc.
Fredrikinkatu 47
Helsinki 00100
FI
téléphone : +358 20 500 7814
mél : tmononen@safenet-inc.com

Déclaration complète de droits de reproduction

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