Groupe de travail Réseau
Request for Comments : 5272
RFC rendue obsolète : 2797
Catégorie : Sur la voie de la normalisation

J. Schaad, Soaring Hawk Consulting
M. Myers, TraceRoute Security, Inc.
juin 2008
Traduction Claude Brière de L'Isle

Gestion de certificat sur CMS (CMC)

Statut de ce mémoire
Le présent document spécifie un protocole Internet sur la voie de la normalisation pour la communauté de l'Internet, et appelle à des discussions et des suggestions pour son amélioration. Prière de se reporter à l'édition actuelle du STD 1 "Normes des protocoles officiels de l'Internet" pour connaître l'état de normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.
(La présente traduction incorpore les errata 2063 et 2731)

Résumé
Le présent document définit la syntaxe de base pour CMC, un protocole de gestion de certificats qui utilise la syntaxe de message cryptographique (CMS, *Cryptographic Message Syntax*). Ce protocole vise deux besoins immédiats au sein de la communauté de l'infrastructure de clé publique de l'Internet (PKI, *Public Key Infrastructure*) :
1. le besoin d'une interface aux produits et services de certification de clé publique fondés sur la CMS et la norme n° 10 de chiffrement à clé publique (PKCS n° 10, *Public Key Cryptography Standard #10*) et
2. le besoin d'un protocole de liste de PKI pour les clés seulement de chiffrement par suite de la conception de l'algorithme ou du matériel.

La CMC exige aussi l'utilisation du document de transport et du document d'usage des exigences avec le présent document pour une définition complète.

Table des matières

1. Introduction

Le présent document définit la syntaxe de base pour le protocole de gestion de certificats sur CMS (CMC, *Certificate Management on CMS*) qui utilise la syntaxe de message cryptographique (CMS, *Cryptographic Message Syntax*). Ce protocole vise deux besoins immédiats de la communauté Internet PKI :

1. Le besoin d'une interface pour les produits et services de certification à clé publique fondés sur la CMS et PKCS n° 10, et
2. Le besoin d'un protocole d'adhésion à PKI pour le chiffrement des seules clés dues à la conception de l'algorithme ou du matériel.

Un petit nombre de services supplémentaires sont définis pour compléter le service central de demande de certification.

1.1 Exigences du protocole

Le protocole doit être fondé autant que possible sur les spécifications existantes de la CMS, de PKCS n° 10 [RFC2314] et du format de message de demande de certificat (CRMF, *Certificate Request Message Format*) [RFC4211].

Le protocole doit prendre en charge les pratiques actuelles de l'industrie d'une demande de certification PKCS n° 10 suivie par une réponse PKCS n° 7 "certs-only" comme un sous ensemble du protocole.

Le protocole doit facilement prendre en charge les protocoles d'adhésion multi clés exigés par S/MIME et d'autres groupes.

Le protocole doit fournir un moyen de faire toutes les opérations d'adhésion en un seul aller-retour. Quand ce n'est pas possible, le nombre d'allers-retours doit être minimisé.

Le protocole doit être conçu de telle façon que toute la génération de clés puisse se produire chez le client.

Les algorithmes obligatoires utilisés par S/MIME doivent être pris en charge. Tous les autres algorithmes cités par les documents centraux de S/MIME devraient être pris en charge.

Le protocole doit contenir les méthodes de preuve de possession (POP, *Proof-of-Possession*). Des dispositions facultatives seront prises pour une POP à multiples allers-retours si nécessaire.

Le protocole doit prendre en charge les réponses différées et en instance aux demandes d'adhésion pour les cas où des procédures externes sont requises pour produire un certificat.

Le protocole doit prendre en charge des chaînes arbitraires d'autorités d'enregistrement (RA, *Registration Authority*) comme intermédiaires entre les demandeurs de certification et les autorités de certification (CA, *Certification Authority*).

1.2 Terminologie des 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].

1.3 Changements depuis la RFC 2797

On a fait une remise à niveau majeure de la disposition du document. Cela inclut deux étapes différentes. D'abors, on a retiré certains paragraphes du document et on les a déplacés dans deux autres documents. Les informations sur la façon de transporter nos messages se trouvent maintenant dans la [RFC5273]. Les informations sur les commandes et la façon dont les sections de ce document doivent être mises en œuvre ainsi que quels algorithmes sont requis se trouvent dans la [RFC5274].

Un certain nombre de nouvelles commandes ont été ajoutées dans cette version :
Informations étendues d'état de CMC au paragraphe 6.1.1
Ancres de confiance publiées au paragraphe 6.15
Données authentifiées au paragraphe 6.16
Traitement par lot de demandes et de réponses au paragraphe 6.17
Informations de publication au paragraphe 6.18
Modifier la demande de certification au paragraphe 6.5.1
Commande traitée au paragraphe 6.19
Preuve d'identité au paragraphe 6.2.2
Témoin d'identité de liaison POP V2 au paragraphe 6.3.1.1

2. Vue d'ensemble du protocole

Une transaction d'adhésion à PKI dans cette spécification est généralement composée d'un seul aller-retour de messages. Dans le plus simple cas, une demande d'adhésion à PKI, qu'on appellera ici une demande PKI, est envoyée du client au serveur et une réponse d'adhésion à PKI, qu'on appellera ici une réponse PKI, est ensuite retournée du serveur au client. Dans des cas plus compliqués, comme la production d'un certificat avec retard, plus d'un aller-retour est nécessaire.

La présente spécification définit deux types de demande PKI et deux types de réponse PKI.

Les demandes PKI sont formée en utilisant la structure PKCS n° 10 ou le CRMF. Les deux demandes PKI sont :
Simple demande PKI : le simple PKCS n° 10 (dans le cas où aucun autre service n'est nécessaire) et
Demande PKI complète : une ou plusieurs structures PKCS n° 10, CRMF ou autres messages de demande enveloppés dans une encapsulation CMS au titre d'une PKIData.

Les réponses PKI sont fondées sur SignedData ou AuthenticatedData [RFC3852]. Les deux réponses PKI sont :
Simple réponse PKI : une SignedData "certs-only" (dans le cas où aucun autre service n'est nécessaire) ou
Réponse PKI complète : un type de contenu PKIResponse enveloppé dans des SignedData.

Aucun service particulier n'est fourni pour le renouvellement (c'est-à-dire, un nouveau certificat avec la même clé) ou le changement de clé (c'est-à-dire, un nouveau certificat avec une nouvelle clé) des certificats de client. Les demandes de renouvellement et de changement de clé ressemblent plutôt à la même chose qu'une demande de certification, sauf que la preuve d'identité est fournie par les certificats existants provenant d'une CA de confiance. (C'est généralement la même CA, mais pourrait être une CA différente dans la même organisation où le nom est partagé.)

Aucun service particulier n'est fourni pour distinguer entre une demande de changement de clé et une nouvelle demande de certification (généralement pour un nouvel objet). Une commande pour un certificat non publié serait normalement incluse dans une demande de changement de clé, et serait omise dans une nouvelle demande de certification. Les CA ou d'autres agents de publication sont aussi supposés avoir des politiques pour retirer les certificats de la publication sur la base de nouveaux certificats ajoutés ou à l'expiration ou la révocation d'un certificat.

Une disposition existe, pour les RA qui participent au protocole, de prendre les demandes PKI, les envelopper dans une seconde couche de demande PKI avec des exigences ou déclarations supplémentaires du RA et de passer enuite ces nouvelles demandes PKI étendues à la CA.

La présente spécification ne fait aucune hypothèse sur le mécanisme de transport sous-jacent. L'utilisation de la CMS n'implique pas un transport fondé sur la messagerie électronique. Plusieurs méthodes de transport différentes possibles sont définies dans la [RFC5273].

Les services facultatifs disponibles dans la présente spécification sont la gestion de transaction, la détection de la répétition (par des noms occasionnels) la production de certificat différée, les demandes de révocation de certificat et la restitution de certificat/liste de révocation de certificats (CRL).

2.1 Terminologie

Plusieurs termes, abréviations, et acronymes différents sont utilisés dans ce document. Ils sont définis ici, sans ordre particulier, pour en faciliter l'usage :

EE (*End-Entity*) : entité d'extrémité, se réfère à l'entité qui possède une paire de clés et pour laquelle un certificat est produit.

RA (*Registration Authority*) : autorité d'enregistrement ou autorité d'enregistrement locale (LRA, *Local RA*) se réfère à une entité qui agit comme intermédiaire entre la EE et la CA. Plusieurs RA peuvent exister entre l'entité d'extrémité et l'autorité de certification. Les RA peuvent effectuer des services supplémentaires comme la génération ou l'archivage de clés. Le présent document utilise le terme de RA pour les RA et les LRA.

CA (*Certification Authority*) : autorité de certification, se réfère à l'entité qui produit les certificats.

Client : se réfère à une entité qui crée une demande PKI. Dans ce document, les RA et les EE peuvent être des clients.

Serveur : se réfère aux entités qui traitent les demandes PKI et créent les réponses PKI. Dans ce document, les CA et les RA peuvent être des serveurs.

PKCS n° 10 : se réfère à la norme de cryptographie à clé publique n° 10 [RFC2314], qui définit la syntaxe de la demande de certification.

CRMF (*Certificate Request Message Format*) : format de message de demande de certification, se réfère à la [RFC4211]. La CMC utilise cette syntaxe de demande de certification définie dans le présent document au titre du protocole.

CMS (*Cryptographic Message Syntax*) : syntaxe de message cryptographique, se réfère à la [RFC3852]. Le présent document fournit des services cryptographiques de base incluant le chiffrement et la signature avec et sans gestion de clét.

Demande/réponse PKI : se réfère aux demandes/réponses décrites dans ce document. Les demandes PKI incluent des demandes de certification, des demandes de révocation, etc. Les réponses PKI incluent des messages certs-only, des messages d'échec, etc.

Preuve d'identité : se réfère au client qui prouve qu'il est qui il dit qu'il est au serveur.

Adhésion ou demande de certification : se réfère au processus d'un client qui demande un certificat. Une demande de certification est un sous ensemble des demandes PKI.

POP (*Proof-of-Possession*) : preuve de possession, se réfère à une valeur qui peut être utilisée pour prouver que la clé privée correspondant à une clé publique est en sa possession et peut être utilisée par une entité d'extrémité. Les différents types de POP sont :

Signature : fournit la POP requise par une opération de signature sur des données.

Direct : fournit l'opération POP requise par un mécanisme de défi/réponse chiffré.

Indirect : fournit l'opération POP requise en retournant le certificat produit dans un état chiffré. (Cette méthode n'est pas utilisée par CMC.)

Publish : fournit l'opération POP requise en produisant la clé privée au producteur du certificat. (Cette méthode n'est pas utilisée actuellement par CMC. Elle pourrait être utilisée par les extensions Génération de clé ou Mandataire de clé.)

Attested : fournit l'opération POP requise en permettant à une entité de confiance d'affirmer que la POP a bien été prouvée par une des méthodes ci-dessus.

OID (*Object IDentifier*) : identifiant d'objet est un type de primitive dans la notation numéro un de syntaxe abstraite (ASN.1, *Abstract Syntax Notation One*).

2.2 Demandes et réponses du protocole

La Figure 1 montre les simples demandes et réponses PKI. Le contenu de la simple demande et réponse PKI est détaillé aux paragraphes 3.1 et 4.1.

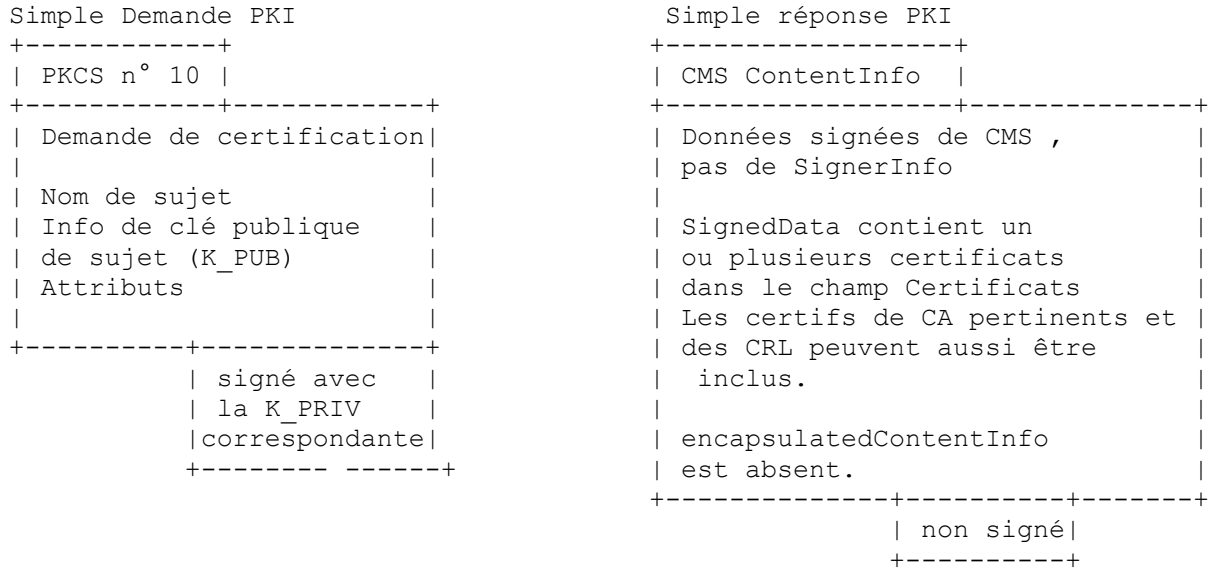

Figure 1 : Simples demandes et réponses de PKI

La Figure 2 montre les demandes et réponses PKI complètes. Le contenu de la demande et réponse PKI complète est détaillé aux paragraphes 3.2 et 4.2.

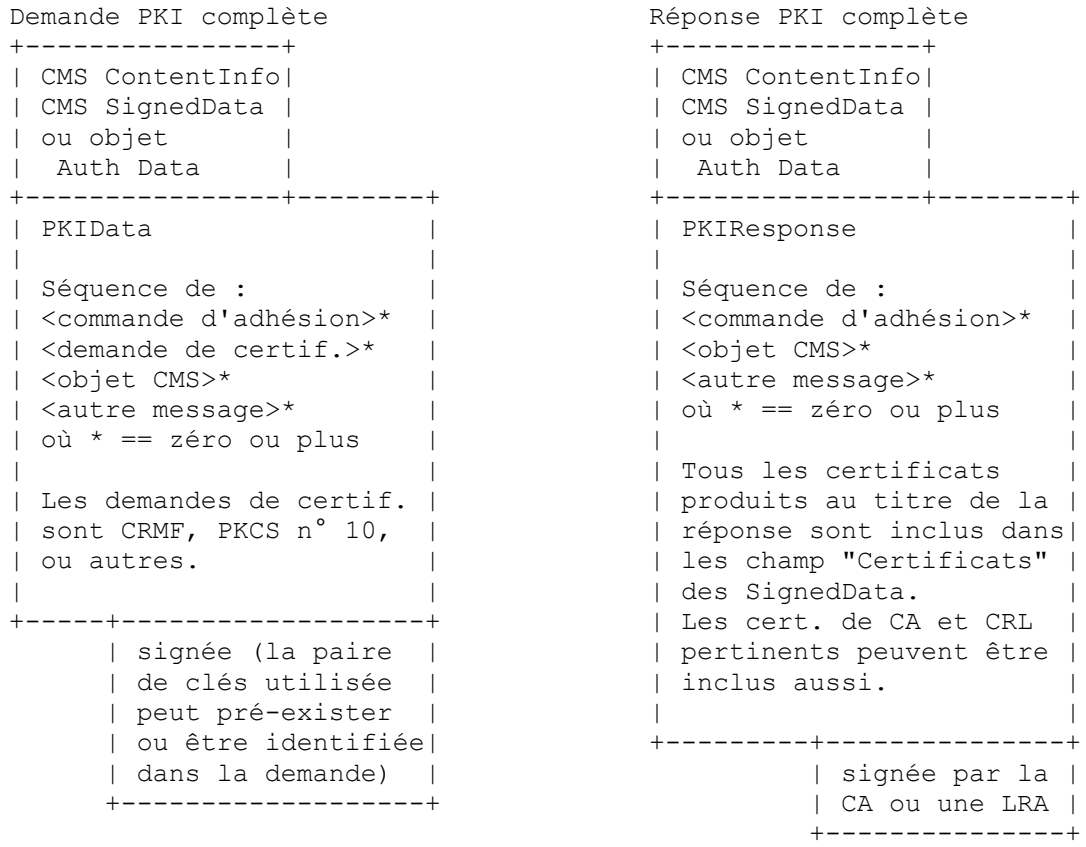

Figure 2 : Demandes et réponses PKI complètes

3. Demandes PKI

Deux types de demandes PKI existent. Cette section donne les détails pour les deux types.

3.1 Simple demande PKI

Une simple demande PKI utilise la syntaxe PKCS n° 10 CertificationRequest [RFC2314].

Quand un serveur traite une simple demande PKI, la réponse PKI retournée est :
Une simple réponse PKI en cas de succès.
Une réponse PKI complète en cas d'échec. Le serveur PEUT choisir de ne pas retourner de réponse PKI dans ce cas.

La simple demande PKI NE DOIT PAS être utilisée si une preuve d'identité doit être incluse.

La simple demande PKI ne peut pas être utilisée si la clé privée n'est pas capable de produire un type de signature (c'est-à-dire, des clés Diffie-Hellman (DH) peuvent utiliser les algorithmes de signature de la [RFC2875] pour la production de la signature).

La simple demande PKI ne peut pas être utilisée pour un des services évolués spécifiés dans ce document.

Le client PEUT incorporer une ou plusieurs extensions X.509v3 dans toute demande de certification fondée sur PKCS n° 10 comme attribut ExtensionReq. L'attribut ExtensionReq est défini comme :

```
ExtensionReq ::= SEQUENCE TAILLE (1..MAX) DE Extension
```

où Extension est importé de la [RFC3280] et ExtensionReq est identifié par :

IDENTIFIANT D'OBJET id-ExtensionReq ::= {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) 14}

Les serveurs DOIVENT être capables de traiter toutes les extensions définies, mais non interdites, dans la [RFC3280]. Les serveurs ne sont pas obligés d'être capables de traiter les autres extensions X.509v3 transmises en utilisant ce protocole, ni ne sont obligés d'être capables de traiter les extensions privées. Les serveurs ne sont pas obligés de mettre toutes les extensions demandées par un client dans un certificat. Il est permis aux serveurs de modifier les extensions demandées par un client. Les serveurs NE DOIVENT PAS altérer une extension au point de rendre invalide l'intention originale d'une extension demandée par un client. (Par exemple, changer l'usage de clé de keyAgreement en digitalSignature.) Si une demande de certification est refusée à cause de l'incapacité à traiter une extension demandée et si une réponse PKI est retournée, le serveur DOIT retourner une réponse PKI avec une valeur de CMCFailInfo de unsupportedExt.

3.2 Demande de PKI complète

La demande PKI complète fournit le plus de fonctionnalités et de souplesse.

La demande PKI complète est encapsulée dans des SignedData ou dans des AuthenticatedData avec un type de contenu encapsulé de id-cct-PKIData (paragraphe 3.2.1).

Quand un serveur traite une demande PKI complète, une réponse PKI DOIT être retournée. La réponse PKI retournée est :
Simple réponse PKI si l'adhésion a réussi et que seuls des certificats sont retournés. (Une commande CMCStatusInfoV2 avec succès est impliquée.)
Réponse PKI complète si l'adhésion a réussi et si des informations sont retournées en plus des certificats, si l'adhésion est en instance, ou si elle a échoué.

Si SignedData est utilisé, la signature peut être générée en utilisant le matériel de clé privée d'une signature incorporée de demande de certification (c'est-à-dire, incluse dans les champs TaggedRequest tcr ou crm) ou d'une clé de signature précédemment certifiée. Si la clé privée d'une demande de certification de signature est utilisée, alors :

a. La demande de certification contenant la clé publique correspondante DOIT inclure une extension Identifiant de clé de sujet.
b. La forme subjectKeyIdentifier du signerIdentifier dans SignerInfo DOIT être utilisée.
c. La valeur de la forme subjectKeyIdentifier des SignerInfo DOIT être l'identifiant de clé de sujet spécifiée dans la demande de certification correspondante. (La forme subjectKeyIdentifier des SignerInfo est utilisée ici parce que aucun certificat n'a

encore été produit pour la clé de signature.) Si la clé de demande est utilisée pour signer, il DOIT y avoir seulement une SignerInfo dans les SignedData.

Si AuthenticatedData est utilisé, alors :
a. L'option Information de mot de passe du receveur des RecipientInfo DOIT être utilisé.
b. Une clé générée au hasard est utilisée pour calculer la valeur du code d'authentification de message (MAC) sur le contenu encapsulé.
c. L'entrée pout l'algorithme de déduction de clé est l'enchaînement de l'identifiant (codé en UTF8) et du secret partagé.

Quand on crée une demande PKI pour renouveler ou changer les clés d'un certificat :
a. Les commandes Identification et Preuve d'identité sont absentes. Les mêmes informations sont fournies en utilisant un certificat existant provenant d'une CA quand on signe la demande PKI. Dans ce cas, la CA qui a produit le certificat d'origine et la CA à qui la demande est faite vont généralement être la même, mais pourraient avoir un opérateur commun.
b. Les CA et RA peuvent imposer des restrictions supplémentaires sur le certificat de signature utilisé. Elles peuvent exiger que le certificat de signature le plus récemment produit pour un client soit utilisé.
c. Certaines CA peuvent empêcher les opérations de renouvellement (c'est-à-dire, la réutilisation des mêmes clés). Dans ce cas, la CA DOIT retourner une réponse PKI avec noKeyReuse comme code d'échec de CMCFailInfo.

3.2.1 Type de contenu PKIData

Le type de contenu PKIData est utilisé pour la demande PKI complète. Un type de contenu PKIData est identifié par :

```
id-cct-PKIData ::= {id-pkix id-cct(12) 2 }
```

La structure ASN.1 correspondant au type de contenu PKIData est :

```
PKIData ::= SEQUENCE {
   controlSequence    SEQUENCE TAILLE(0..MAX) DE TaggedAttribute,
   reqSequence        SEQUENCE TAILLE(0..MAX) DE TaggedRequest,
   cmsSequence        SEQUENCE TAILLE(0..MAX) DE TaggedContentInfo,
   otherMsgSequence SEQUENCE TAILLE(0..MAX) DE OtherMsg
}
```

Les champs dans PKIData ont la signification suivante :

controlSequence est une séquence de commandes. Les commandes définies dans ce document se trouvent à la in Section 6. Des commandes peuvent être définies par d'autres parties. Des détails sur la structure TaggedAttribute se trouvent au paragraphe 3.2.1.1.

reqSequence est une séquence de demandes de certification. Les demandes de certification peuvent être une CertificationRequest (PKCS n° 10), un CertReqMsg (CRMF), ou une demande PKI définie en externe. Les détails complets se trouvent au paragraphe 3.2.1.2. Si une demande de certification définie en externe est présente, mais si le serveur ne comprend pas la demande de certification (ou ne veut pas la traiter) un CMCStatus de noSupport DOIT être retourné pour l'élément de demande de certification et aucune autre demande de certification n'est traitée.

cmsSequence est une séquence d'objets de message de la [RFC3852]. Voir les détails au paragraphe 3.2.1.3.

otherMsgSequence est une séquence d'objets de données arbitraires. Les objets de données placés ici sont référencés par une ou plusieurs commandes. Cela permet que les commandes utilisent de grandes quantités de données sans que les données soient incorporées dans la commande. Voir les détails au paragraphe 3.2.1.4.

Toutes les demandes de certification codées dans une seule PKIData DEVRAIENT être pour la même identité. Les RA qui traitent par lot (voir au paragraphe 6.17) sont supposées placer les demandes PKI reçues dans la cmsSequence d'une PKIData.

Le traitement des PKIData par un receveur est comme suit :

1. Toutes les commandes devraient être examinées et traitées d'une manière appropriée. Le traitement approprié est d'achever le traitement en cours, d'ignorer la commande, ou de la placer sur une liste de choses à faire pour traitement ultérieur. Les commandes peuvent être traitées dans n'importe quel ordre ; l'ordre dans la séquence n'est pas significatif.

2. Les éléments dans la reqSequence ne sont pas référencés par une commande. Ces éléments, qui sont des demandes de certification, doivent aussi être traitées. Comme avec les commandes, le traitement approprié peut être immédiat ou l'ajout à la liste des choses à faire ultérieurement.

3. Finalement, la liste des choses à faire est traitée. Souvent, la liste des choses à faire va être ordonnée en groupant des tâches spécifiques.

Aucun traitement n'est exigé pour les membres de cmsSequence ou otherMsgSequence de PKIData si ils sont présents et ne sont pas référencés par une commande. Dans ce cas, les membres de cmsSequence et otherMsgSequence sont ignorés.

3.2.1.1 Syntaxe des commandes

Les actions à effectuer pour une demande/réponse PKI sont fondées sur les commandes incluses. Chaque commande consiste en un identifiant d'objet et une valeur fondée sur l'identifiant d'objet.

La syntaxe d'une commande est :

```
TaggedAttribute ::= SEQUENCE {
   bodyPartID       BodyPartID,
   attrType         IDENTIFIANT D'OBJET,
   attrValues       ENSEMBLE DE AttributeValue
}

AttributeValue ::= ANY
```

Les champs dans TaggedAttribute ont la signification suivante :

bodyPartID est un entier unique qui identifie cette commande.

attrType est l'OID qui identifie la commande.

attrValues sont les valeurs de données utilisées pour traiter la commande. La structure des données dépend de la commande.

Le serveur final DOIT faire échouer tout le trautement d'une PKIData entière si une commande incluse n'est pas reconnue, si cette commande n'est pas déjà marquée traitée par une commande Commande traitée (voir au paragraphe 6.19) et si aucune autre erreur n'est générée. La réponse PKI DOIT inclure une valeur de CMCFailInfo avec la valeur badRequest et la bodyList DOIT contenir le bodyPartID de la ou les commandes invalides ou non reconnues. Un serveur est le serveur final si et seulement si il ne passe pas la demande PKI à un autre serveur. Un serveur n'est pas considéré comme étant le serveur final si il aurait passé la demande PKI à un autre serveur, mais a, à la place, retourné une erreur de traitement.

Les commandes définies dans le présent document se trouvent à la Section 6.

3.2.1.2 Formats de demande de certification

Les demandes de certification sont fondées sur PKCS n° 10, CRMF, ou d'autres formats de demandes. Le paragraphe 3.2.1.2.1 spécifie les exigences pour les clients et les serveurs qui traitent PKCS n° 10. Le paragraphe 3.2.1.2.2 spécifie les exigences pour les clients et serveurs qui traitent CRMF. Le paragraphe 3.2.1.2.3 spécifie les exigences pour les clients et les serveurs qui traitent les autres demandes.

```
TaggedRequest ::= CHOIX {
  tcr              [0] TaggedCertificationRequest,
  crm              [1] CertReqMsg,
  orm              [2] SEQUENCE {
    bodyPartID             BodyPartID,
    requestMessageType     IDENTIFIANT D'OBJET,
    requestMessageValue    ANY DÉFINI PAR requestMessageType
  }
}
```

Les champs dans TaggedRequest ont la signification suivante :

tcr est une demande de certification qui utilise la syntaxe PKCS n° 10. Les détails de PKCS n° 10 sont au paragraphe 3.2.1.2.1.

crm est une demande de certification qui utilise la syntaxe CRMF. Les détails de CRMF sont au paragraphe 3.2.1.2.2.

orm est une demande de certification défini en externe. Un exemple est une demande de certification d'attribut. Les champs de cette structure sont :

bodyPartID est le numéro d'identifiant pour cette demande de certification. Les détails sur les identifiants de partie de corps sont au paragraphe 3.2.2.

requestMessageType identifie le type Autre demande. Ces valeurs sont définies en dehors de ce document.

requestMessageValue sont les données associées au type Autre demande.

3.2.1.2.1 Syntaxe de la certification PKCS n° 10

Une demande de certification fondée sur PKCS n° 10 utilise la structure ASN.1 suivante :

```
TaggedCertificationRequest ::= SEQUENCE {
   bodyPartID            BodyPartID,
   certificationRequest  CertificationRequest
}
```

Les champs dans TaggedCertificationRequest ont la signification suivante :

bodyPartID est le numéro d'identifiant pour cette demande de certification. Les détails des identifiants de partie de corps sont au paragraphe 3.2.2.

certificationRequest contient la demande de certification fondée sur PKCS n° 10. Ses champs sont décrits dans la [RFC2314].

Quand ils produisent une demande de certification fondée sur PKCS n° 10, les clients DOIVENT produire la demande de certification avec un nom de sujet et une clé publique. Certains produits PKI sont traités en utilisant un répertoire central d'informations pour allouer des noms de sujet à réception d'une demande de certification. Pour s'accommoder de ce mode de fonctionnement, le champ Sujet dans une CertificationRequest PEUT être NULL, mais DOIT être présent. Les CA qui reçoivent une CertificationRequest avec un champ Sujet NULL PEUVENT rejeter de telles demandes de certification. Si elle est rejetée et si une réponse PKI est retournée, la CA DOIT retourner une réponse PKI avec la valeur CMCFailInfo réglée à badRequest.

3.2.1.2.2 Syntaxe de la certification CRMF

Un message CRMF utilise la structure ASN.1 suivante (définie dans la [RFC4211] et incluse ici par facilité) :

```
CertReqMsg ::= SEQUENCE {
  certReq    CertRequest,
  popo       ProofOfPossession  FACULTATIF,       -- le contenu dépend du type de clé.
  RegInfo    SEQUENCE TAILLE(1..MAX) DE AttributeTypeAndValue FACULTATIF }

CertRequest ::= SEQUENCE {
  certReqId      ENTIER,                          -- Identifiant pour confronter demande et réponse.
  certTemplate   CertTemplate,                     -- Champs choisis de certificat à produire.
  controls       Controls FACULTATIF }            -- Attributs qui affectent la production.

CertTemplate ::= SEQUENCE {
  version        [0] Version                 FACULTATIF,
  serialNumber   [1] ENTIER                  FACULTATIF,
  signingAlg     [2] AlgorithmIdentifier     FACULTATIF,
  issuer         [3] Name                    FACULTATIF,
  validity       [4] OptionalValidity        FACULTATIF,
  subject        [5] Name                    FACULTATIF,
  publicKey      [6] SubjectPublicKeyInfo    FACULTATIF,
  issuerUID      [7] UniqueIdentifier        FACULTATIF,
```

```
subjectUID    [8] UniqueIdentifier        FACULTATIF,
extensions    [9] Extensions              FACULTATIF }
```

Les champs dans CertReqMsg sont expliqués dans la [RFC4211].

Le présent document impose les restrictions supplémentaires suivantes à la construction et au traitement des demandes de certification CRMF :

Quand une demande PKI complète inclut une demande de certification CRMF, les deux champs subject et publicKey dans CertTemplate DOIVENT être définis. Le champ subject peut être codé comme NULL, mais DOIT être présent.

Quand des commandes CRMF et CMC existent toutes deux avec une fonction équivalente, la commande CMC DEVRAIT être utilisée. La commande CMC DOIT outrepasser la commande CRMF.

Le champ regInfo NE DOIT PAS être utilisé sur une demande de certification. Une fonction équivalente est fournie dans la commande CMC regInfo (paragraphe 6.12).

La méthode indirecte de preuve de POP n'est pas prise en charge dans ce protocole. Une des autres méthodes (incluant la méthode directe décrite dans ce document) DOIT être utilisée. La valeur de encrCert dans SubsequentMessage NE DOIT PAS être utilisée.

Comme subject et publicKeyValues sont toujours présents, le POPOSigningKeyInput NE DOIT PAS être utilisé dans le calcul de la valeur de POPSigningKey.

Un serveur n'est pas obligé d'utiliser toutes les valeurs suggérées par le client dans la demande de certification CRMF. Les serveurs DOIVENT être capables de traiter toutes les extensions définies, mais non interdites dans la [RFC3280]. Les serveurs ne sont pas obligés d'être capables de traiter les autres extensions X.509v3 transmises en utilisant ce protocole, ni ne sont obligés d'être capables de traiter les extensions privées. Il est permis aux serveurs de modifier les extensions demandées par le client. Les serveurs NE DOIVENT PAS altérer une extension en invalidant l'intention originale d'une extension demandée par le client (par exemple, changer l'usage de clé de keyAgreement en digitalSignature.) Si une demande de certification est refusée à cause de l'incapacité à traiter une extension demandée, le serveur DOIT répondre avec une réponse PKI complète avec une valeur CMCFailInfo de unsupportedExt.

3.2.1.2.3 Autres demande de certification

Le présent document permet que d'autres formats de demande de certification soient définis et utilisés. Un exemple d'un autre format de demande de certification est celui des certificats d'attribut. Ces autres formats de demande de certification sont définis en spécifiant un OID pour l'identification et la structure pour contenir les données à passer.

3.2.1.3 Objets d'informations de contenu

Le champ cmsSequence des messages PKIData et PKIResponse contient zéro, un ou plusieurs objets d'informations de contenu étiquetés. La syntaxe de cette structure est :

```
TaggedContentInfo ::= SEQUENCE {
   bodyPartID     BodyPartID,
   contentInfo    ContentInfo
}
```

Les champs dans TaggedContentInfo ont la signification suivante :

bodyPartID est un entier unique qui identifie cet objet d'informations de contenu.

contentInfo est un objet ContentInfo (défini dans la [RFC3852]).

Les quatre types de contenu utilisés dans cmsSequence sont AuthenticatedData, Data, EnvelopedData, et SignedData. Tous ces types de contenu sont définis dans la [RFC3852].

3.2.1.3.1 Données authentifiées

Le type de contenu AuthenticatedData fournit une méthode pour faire la validation fondée sur le secret pré partagé de données envoyées entre deux parties. À la différence de SignedData, il ne spécifie pas quelle partie a réellement généré l'information.

AuthenticatedData fournit l'authentification de l'origine dans des circonstances où un secret partagé existe, mais la confiance fondée sur PKI n'a pas encore été établie. Aucune confiance fondée sur PKI ne peut avoir été établie parce qu'une ancre de confiance n'a pas été installée sur le client ou qu'aucun certificat n'existe pour une clé de signature.

Le type de contenu AuthenticatedData est utilisé dans ce document pour :
la commande id-cmc-authData (paragraphe 6.16), et
l'enveloppeur de niveau supérieur dans les environnements où une clé de chiffrement seul est certifiée.

Ce type de contenu peut inclure des PKIData et une PKIResponse comme types de contenu encapsulé. Ces types de contenu incorporés peuvent contenir des commandes supplémentaires qui doivent être traitées.

3.2.1.3.2 Data

Le type de contenu Data permet un transport général de données non structurées.

Le type de contenu Data est utilisé par ce document pour :
Contenir la valeur aléatoire chiffrée y pour la preuve de POP dans la commande POP chiffrée (voir le paragraphe 6.7).

3.2.1.3.3 Données enveloppées

Le type de contenu EnvelopedData assure l'enveloppement des données.

Le type de contenu EnvelopedData est la principale méthode de protection de la confidentialité pour les informations sensibles dans ce protocole. EnvelopedData peut fournir le chiffrement d'une demande PKI entière (voir la Section 5). EnvelopedData peut aussi être utilisé pour envelopper le matériel de clé privée pour l'archivage de clé. Si le déchiffrement sur une EnvelopedData échoue, une réponse PKI complète est retournée avec une valeur de CMCFailInfo de badMessageCheck et un bodyPartID de 0.

3.2.1.3.4 Données signées

Le type de contenu SignedData fournit l'authentification et la protection de l'intégrité.

Le type de contenu SignedData est utilisé par ce document pour :
L'enveloppe externe d'une demande PKI.
L'enveloppe externe d'une réponse PKI.

Au titre du traitement d'une demande/réponse PKI, la ou les signatures DOIVENT être vérifiées. Si la signature ne se vérifie pas et si la demande/réponse PKI contient autre chose qu'une commande CMC Information d'état, une réponse PKI complète contenant une commande CMC Informations d'état DOIT être retournée en utilisant une CMCFailInfo de valeur badMessageCheck et un identifiant de partie de corps de 0.

Pour la réponse PKI, SignedData permet au serveur de signer les données de retour, si il en existe, et de porter les certificats et CRL correspondant à la demande PKI. Si aucune donnée n'est retournée au delà des certificats et CRL, les champs EncapsulatedInfo et SignerInfo ne sont pas remplis.

3.2.1.4 Autres corps de message

Le champ otherMsgSequence de la demande/réponse PKI permet de porter des objets de données arbitraires au titre d'une demande/réponse PKI. C'est destiné à contenir un objet de données qui n'est pas déjà enveloppé dans un champ cmsSequence (paragraphe 3.2.1.3). L'objet de données est ignoré sauf si une commande fait référence à l'objet de données par son identifiant de partie de corps (bodyPartID).

```
OtherMsg ::= SEQUENCE {
   bodyPartID        BodyPartID,
   otherMsgType      IDENTIFIANT D'OBJET,
   otherMsgValue     ANY DÉFINI PAR otherMsgType }
```

Les champs dans OtherMsg ont la signification suivante :
bodyPartID est l'identifiant unique qui identifie cet objet de données.
otherMsgType est l'OID qui définit le type de corps de message.
otherMsgValue sont les données.

3.2.2 Identification de partie de corps

Chaque élément d'une PKIData ou PKIResponse a un identifiant de partie de corps associé. L'identifiant de partie de corps est un entier de 4 octets utilisant l'ASN.1 suivant :

```
bodyIdMax ENTIER ::= 4 294 967 295

BodyPartID ::= ENTIER(0..bodyIdMax)
```

Les identifiants de partie de corps sont codés dans le champ certReqIds pour les objets CertReqMsg (dans une demande étiquetée (TaggedRequest)) ou dans le champ bodyPartID des autres objets. L'identifiant de partie de corps DOIT être unique au sein d'une seule PKIData ou PKIResponse. Les identifiants de partie de corps peuvent être dupliqués dans les différentes couches (par exemple, une PKIData incorporées dans une autre).

La valeur de bodyPartID de 0 est réservée pour être utilisée comme référence à l'objet PKIData en cours.

Certaines commandes, comme Ajouter des extensions (paragraphe 6.5.2) utilisent l'identifiant de partie de corps dans le champ pkiDataReference pour se référer à une demande PKI dans les PKIData en cours. Certaines commandes, comme Informations étendues d'état de CMC (paragraphe 6.1.1) vont aussi utiliser des identifiants de partie de corps pour se référer aux éléments dans la précédente demande PKI/

Réponse. Cela permet qu'une erreur soit explicite au sujet de la commande ou demande PKI à laquelle l'erreur s'applique.

Une BodyPartList contient une liste des parties de corps d'une demande/réponse PKI (c'est-à-dire, la commande Regrouper les demandes du paragraphe 6.17). Le type ASN.1 BodyPartList est défini comme :

```
BodyPartList ::= SEQUENCE TAILLE(1..MAX) DE BodyPartID
```

Un BodyPartPath contient un chemin des identifiants de partie de corps qui bougent à travers leur situation (c'est-à-dire, la commande Modifier la demande de certification au paragraphe 6.5.1). Le type ASN.1 BodyPartPath est défini comme :

```
BodyPartPath ::= SEQUENCE TAILLE(1..MAX) DE BodyPartID
```

3.2.3 Attribut Données CMC non signées

Il y a parfois besoin d'inclure des données dans une demande PKI destinée à être supprimée par une RA durant le traitement. Un exemple en est l'inclusion d'une clé privée chiffrée, où un agent d'archivage de clé supprime la clé privée chiffrée avant de l'envoyer à la CA. Un effet collatéral de ce désir est que chaque RA qui encapsule ces informations a besoin de déplacer les données afin qu'elles ne soient pas couvertes par la signature de la RA. (Une demande de client PKI encapsulée par une RA ne peut pas avoir une commande signée supprimée par l'agent d'archivage de clé sans casser la signature de la RA.) L'attribut de CMC Données non signées traite ce problème.

L'attribut de CMC Données non signées contient des informations qui ne sont pas directement signées par un client. Quand une RA rencontre cet attribut dans le champ d'attribut signé ou non authentifié d'une demande qu'il agrège, l'attribut de CMC Données non signées est supprimé de la demande avant de placer la demande dans une cmsSequence et est placé dans les tattributs non signés ou non authentifiés de l'enveloppe signée ou authentifiée de la RA.

L'attribut de CMC Données non signées est identifié par :

IDENTIFIANT D'OBJET id-aa-cmc-unsignedData ::= {id-aa 34}

L'attribut de CMC Données non signées a la définition ASN.1 :

```
CMCUnsignedData ::= SEQUENCE {
  bodyPartPath   BodyPartPath,
```

```
  identifier      IDENTIFIANT D'OBJET,
  content         ANY DÉFINI PAR identifier
}
```

Les champs dans CMCUnsignedData ont la signification suivante :

bodyPartPath est le chemin qui pointe sur la commande associé à ces données. Quand une RA déplace la commande dans un attribut non signé ou non authentifié jusqu'à un niveau au titre de l'enveloppement des données dans un nouveau SignedData ou AuthenticatedData, l'identifiant de partie de corps de l'élément incorporé dans les PKIData est ajouté devant la séquence bodyPartPath.

identifier est l'OID qui définit les données associées.

content sont les données.

Il DOIT y avoir au plus un attribut de CMC Données non signées dans la séquence UnsignedAttribute d'une SignerInfo ou dans la séquence UnauthenticatedAttribute d'un AuthenticatedData.
UnsignedAttribute consiste en un ensemble de valeurs ; l'attribut peut avoir tout nombre de valeurs supérieures à zéro dans cet ensemble. Si l'attribut de CMC Données non signées est dans un SignerInfo ou AuthenticatedData, il DOIT apparaître avec la ou les mêmes valeurs dans tous les éléments SignerInfo et AuthenticatedData.

4. Réponses PKI

Deux types de réponses PKI existent. Cette section donne les détails des deux types.

4.1 Réponse PKI simple

Les clients DOIVENT être capables de traiter la simple réponse PKI. La simple réponse PKI consiste en un SignedData sans EncapsulatedContentInfo et sans SignerInfo. Les certificats demandés dans la réponse PKI sont retournés dans le champ Certificat des SignedData.

Les clients NE DOIVENT PAS supposer que les certificats sont dans un ordre particulier. Les serveurs DEVRAIENT inclure tous les certificats intermédiaires nécessaires pour former des chemins de certification complets à une ou plusieurs ancres de confiance, pas juste le ou les certificats nouvellement produits. Le serveur PEUT de plus retourner des CRL dans le champ CRL. Les serveurs PEUVENT inclure des certificats auto signés. Les clients NE DOIVENT PAS faire implicitement confiance aux certificats auto signés inclus simplement à cause de leur présence dans le groupe de certificats. Dans le cas où les clients reçoivent un nouveau certificat auto signé du serveur, ils DEVRAIENT fournir un mécanisme pour permettre à l'utilisateur d'utiliser le certificat comme une ancre de confiance. (La commande Publier les ancres de confiance (paragraphe 6.15) devrait être utilisée dans le cas où le serveur veut que le client accepte un ou plusieurs certificats comme ancres de confiance. Cela exige l'utilisation du message Réponse PKI complète.)

4.2 Réponse PKI complète

Les clients DOIVENT être capables de traiter une réponse PKI complète.

La réponse PKI complète consiste en SignedData ou AuthenticatedData encapsulant un type de contenu PKIResponse. Les certificats produits dans une réponse PKI sont retournés dans le champ Certificats des SignedData immédiatement encapsulantes.

Les clients NE DOIVENT PAS supposer un ordre quelconque des certificats. Les serveurs DEVRAIENT inclure tous les certificats intermédiaires nécessaires pour former des chaînes complètes avec une ou plusieurs ancres de confiance, et pas juste le ou les certificats nouvellement produits. Le serveur PEUT de plus retourner des CRL dans le sac à CRL. Les serveurs PEUVENT inclure des certificats auto signés. Les clients NE DOIVENT PAS faire implicitement confiance aux certificats auto signés inclus simplement à cause de leur présence dans le sac de certificats. Dans le cas où les clients reçoivent un nouveau certificat auto signé du serveur, ils PEUVENT fournir un mécanisme pour permettre à l'utilisateur d'utiliser explicitement le certificat comme une ancre de confiance. (La commande Publier les ancres de confiance (paragraphe 6.15) existe afin de permettre la distribution des certificats d'ancre de confiance. Si une ancre de confiance publie une nouvelle ancre de confiance, c'est un cas où la confiance automatique de la nouvelle ancre de confiance pourrait être permise.)

4.2.1 Type de contenu PKIResponse

Le type de contenu PKIResponse est utilisé pour la réponse PKI complète. Le type de contenu PKIResponse est identifié par :

IDENTIFIANT D'OBJET id-cct-PKIResponse ::= {id-pkix id-cct(12) 3 }

La structure ASN.1 correspondant au type de contenu PKIResponse est :

```
PKIResponse ::= SEQUENCE {
   controlSequence     SEQUENCE TAILLE(0..MAX) DE TaggedAttribute,
   cmsSequence         SEQUENCE TAILLE(0..MAX) DE TaggedContentInfo,
   otherMsgSequence  SEQUENCE TAILLE(0..MAX) DE OtherMsg
}

ReponseBody ::= PKIResponse
```

Note : dans la [RFC2797], ce type ASN.1 était appelé ResponseBody. Il a été converti en PKIResponse pour être plus clair et l'ancien nom est gardé comme synonyme.

Les champs dans PKIResponse ont la signification suivante :

controlSequence est une séquence de commandes. Les commandes définies dans ce document se trouvent à la Section 6. Les commandes peuvent être définies par d'autres parties. Les détails de la structure TaggedAttribute se trouvent au paragraphe 3.2.1.1.

cmsSequence est une séquence d'objet de message [RFC3852]. Voir les détails au paragraphe 3.2.1.3.

otherMsgSequence est une séquence d'objets de données arbitraires. Les objets de données placés ici sont référencés par une ou plusieurs commandes. Cela permet que les commandes utilisent de grandes quantités de données sans que celles-ci soient incorporées dans la commande. Voir les détails au paragraphe 3.2.1.4.

Le traitement de PKIResponse par un receveur est le suivant :

1. Toutes les commandes devraient être examinées et traitées de manière appropriée. Le traitement approprié est d'achever le traitement en une fois, d'ignorer la commande, ou de placer la commande sur une liste de choses à faire ultérieurement.
2. Le traitement supplémentaire des parties de non élément inclut de sauvegarder les certificats et CRL présents dans des couches d'enveloppement. Ce type de traitement se fonde sur le consommateur de l'élément et les générateurs ne devraient pas s'appuyer dessus.

Aucun traitement n'est exigé pour les membres de cmsSequence ou otherMsgSequence de la PKIResponse, si des éléments sont présents et ne sont pas référencés par une commande. Dans ce cas, les membres de cmsSequence et otherMsgSequence sont à ignorer.

5. Application du chiffrement à une demande/réponse PKI

Il y a des occasions où une demande ou réponse PKI doit être chiffrée afin d'empêcher la divulgation des informations de la demande/réponse PKI à des entités non autorisées. Cette section décrit les moyens pour chiffrer les demandes et réponses PKI complètes (les simples demandes PKI ne peuvent pas être chiffrées). Des portions de données des demandes et réponses PKI qui sont placées dans le champ cmsSequence peuvent être chiffrées séparément.

La confidentialité est fournie en enveloppant la demande/réponse PKI (des SignedData) dans des EnvelopedData. Le type de contenu incorporé dans les EnvelopedData est id-SignedData. Noter que c'est différent de S/MIME où il y a une couche MIME placée entre les données chiffrées et signées. Il est recommandé que si une couche EnvelopedData est appliquée à une demande/réponse PKI, une seconde couche de signature soit placée en dehors de la couche EnvelopedData. La figure qui suit montre comment pourrait être effectuée cette incorporation :

Normal	**Option 1**	**Option 2**
SignedData	EnvelopedData	SignedData
PKIData	SignedData	EnvelopedData
	PKIData	SignedData
		PKIData

Note : PKIResponse peut être substitué à PKIData dans cette figure.

Les options 1 et 2 empêchent la fuite de données sensibles en chiffrant la demande/réponse PKI complète. Une RA qui reçoit une demande PKI qu'elle ne peut pas déchiffrer PEUT rejeter la demande PKI sauf si elle peut traiter la demande PKI sans connaître son contenu (c'est-à-dire, tout ce qu'elle fait est d'amalgamer plusieurs demandes PKI et les transmettre à un serveur).

Après que la RA a retiré l'enveloppe et terminé le traitement, elle peut alors appliquer une nouvelle couche EnvelopedData pour protéger les demandes PKI lors de la transmission au prochain agent de traitement. La Section 7 contient les informations sur le traitement par la RA.

Les demandes/réponses PKI complètes peuvent être chiffrées ou transmises en clair. Les serveurs DOIVENT fournir la prise en charge des trois options.

Autrement, un canal sûr authentifié pourrait exister entre des parties qui exigent la confidentialité. Les clients et serveurs PEUVENT utiliser de tels canaux plutôt que la technique décrite ci-dessus pour fournir une communication privée sûre des demandes/réponses PKI simples et complètes.

6. Commandes

Les commandes sont portées au titre des demandes et réponses PKI complètes. Chaque commande est codée par un OID unique suivi par les données de la commande (voir la syntaxe au paragraphe 3.2.1.1). Le codage des données se fonde sur la commande. Les systèmes de traitement vont d'abord détecter l'OID (type d'attribut TaggedAttribute) et traiter la valeur de la commande correspondante (valeurs d'attribut TaggedAttribute) avant de traiter le corps du message.

Les OID sont tous définis sous l'arc suivant :

IDENTIFIANT D'OBJET id-pkix ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) }

IDENTIFIANT D'OBJET id-cmc ::= { id-pkix 7 }

Le tableau suivant fait la liste des noms, de l'OID, et de la structure syntaxique pour chaque commande décrite dans ce document.

Description de l'identifiant	**OID**	**Structure ASN.1**	**Paragraphe**
id-cmc-statusInfo	id-cmc 1	CMCStatusInfo	6.1.2
id-cmc-identification	id-cmc 2	UTF8String	6.2.3
id-cmc-identityProof	id-cmc 3	CHAÎNE D'OCTETS	6.2.2
id-cmc-dataReturn	id-cmc 4	CHAÎNE D'OCTETS	6.4
id-cmc-transactionId	id-cmc 5	ENTIER	6.6
id-cmc-senderNonce	id-cmc 6	CHAÎNE D'OCTETS	6.6
id-cmc-recipientNonce	id-cmc 7	CHAÎNE D'OCTETS	6.6
id-cmc-addExtensions	id-cmc 8	AddExtensions	6.5.2
id-cmc-encryptedPOP	id-cmc 9	EncryptedPOP	6.7
id-cmc-decryptedPOP	id-cmc 10	DecryptedPOP	6.7
id-cmc-lraPOPWitness	id-cmc 11	LraPOPWitness	6.8
id-cmc-getCert	id-cmc 15	GetCert	6.9
id-cmc-getCRL	id-cmc 16	GetCRL	6.10
id-cmc-revokeRequest	id-cmc 17	RevokeRequest	6.11
id-cmc-regInfo	id-cmc 18	CHAÎNE D'OCTETS	6.12
id-cmc-responseInfo	id-cmc 19	CHAÎNE D'OCTETS	6.12
id-cmc-queryPending	id-cmc 21	CHAÎNE D'OCTETS	6.13
id-cmc-popLinkRandom	id-cmc 22	CHAÎNE D'OCTETS	6.3.1

id-cmc-popLinkWitness	id-cmc 23	CHAÎNE D'OCTETS	6.3.1
id-cmc-popLinkWitnessV2	id-cmc 33	CHAÎNE D'OCTETS	6.3.1.1
id-cmc-confirmCertAcceptance	id-cmc 24	CMCCertId	6.14
id-cmc-statusInfoV2	id-cmc 25	CMCStatusInfoV2	6.1.1
id-cmc-trustedAnchors	id-cmc 26	PublishTrustAnchors	6.15
id-cmc-authData	id-cmc 27	AuthPublish	6.16
id-cmc-batchRequests	id-cmc 28	BodyPartList	6.17
id-cmc-batchResponses	id-cmc 29	BodyPartList	6.17
id-cmc-publishCert	id-cmc 30	CMCPublicationInfo	6.18
id-cmc-modCertTemplate	id-cmc 31	ModCertTemplate	6.5.1
id-cmc-controlProcessed	id-cmc 32	ControlsProcessed	6.19
id-cmc-identityProofV2	id-cmc 34	IdentityProofV2	6.2.1

Tableau 1 : Attributs des commandes CMC

6.1 Commandes d'informations d'état de CMC

Les commandes Informations d'état de CMC retournent des informations sur l'état d'une demande/réponse d'un client/serveur. Deux commandes sont décrites dans c paragraphe. La commande Informations étendues d'état de CMC est la commande préférée ; la commande Informations d'état de CMC est incluse pour la rétro compatibilité avec la RFC 2797.

Les serveurs PEUVENT émettre plusieurs commandes Informations d'état de CMC se référant à une seule partie de corps. Les clients DOIVENT être capables de traiter plusieurs commandes Informations d'état de CMC dans une réponse PKI. Les serveurs DOIVENT utiliser la commande Informations étendues d'état de CMC, mais PEUVENT de plus utiliser la commande Informations d'état de CMC. Les clients DOIVENT être capables de traiter la commande Informations étendues d'état de CMC.

6.1.1 Commande Informations étendues d'état de CMC

La commande Informations étendues d'état de CMC est identifiée par l'OID id-cmc-statusInfoV2 ::= { id-cmc 25 }

La commande Informations étendues d'état de CMC a la définition ASN.1 suivante :

```
CMCStatusInfoV2 ::= SEQUENCE {
  cMCStatus        CMCStatus,
  bodyList         SEQUENCE TAILLE (1..MAX) DE BodyPartReference,
  statusString     UTF8String      FACULTATIF,
  otherInfo        OtherStatusInfo FACULTATIF
}

OtherStatusInfo ::= CHOIX {
  failInfo           CMCFailInfo,
  pendInfo           PendInfo,
  extendedFailInfo   ExtendedFailInfo
}

PendInfo ::= SEQUENCE {
  pendToken        CHAÎNE D'OCTETS,
  pendTime         GeneralizedTime
}

ExtendedFailInfo ::= SEQUENCE {
  failInfoOID          IDENTIFIANT D'OBJET,
  failInfoValue        ANY DEFINI PAR failInfoOID
}

BodyPartReference ::= CHOIX {
  bodyPartID       BodyPartID,
  bodyPartPath     BodyPartPath
}
```

Les champs dans CMCStatusInfoV2 ont la signification suivante :

cMCStatus contient la valeur d'état retournée. Les détails sont au paragraphe 6.1.3.

bodyList identifie les commandes ou autres éléments auxquels la valeur d'état s'applique. Si une erreur est retournée pour une simple demande PKI, ce champ est le choix bodyPartID de la BodyPartReference avec le seul entier de valeur 1.

statusString contient des informations de description supplémentaires. Cette chaîne est lisible par l'homme.

otherInfo contient des informations supplémentaires sur le code d'état de CMC retourné dans le champ cMCStatus.

Les champs dans OtherStatusInfo ont la signification suivante :

failInfo est décrit au paragraphe 6.1.4. Il fournit un code d'erreur qui précise la défaillance qui s'est produite. Ce choix n'est présent que si cMCStatus contient la valeur "failed".

pendInfo contient des informations sur quand et comment le client devrait demander le résultat de cette demande. Il est présent quand le cMCStatus est soit en instance, soit partiel. pendInfo utilise la structure PendInfo, qui a les champs suivants :

pendToken est le jeton utilisé dans la commande Interrogation en instance (paragraphe 6.13).

pendTime contient l'heure suggérée à laquelle le serveur veut être interrogé sur l'état de la demande de certification.

extendedFailInfo inclut des informations d'erreur détaillées qui dépendent de l'application. Ce choix n'est présent que si cMCStatus contient la valeur "échec". Des précautions devraient être prises quand on définit de nouvelles valeurs car elles pourraient n'être pas correctement reconnues par tous les clients et serveurs. La valeur CMCFailInfo de internalCAError peut être supposée si l'erreur étendue n'est pas reconnue. Ce champ utilise le type ExtendedFailInfo. ExtendedFailInfo a les champs suivants :

failInfoOID contient un OID qui est associé à un ensemble de valeurs d'erreur étendues.

failInfoValue contient un code d'erreur étendue à partir de l'ensemble de codes d'erreurs étendues défini.

Si le champ cMCStatus est "succès", la commande Informations étendues d'état de CMC PEUT être omise sauf si elle est le seul élément de la réponse.

6.1.2 Commande Informations d'état de CMC

La commande Informations d'état de CMC est identifiée par l'OID id-cmc-statusInfo ::= { id-cmc 1 }

La commande Informations d'état de CMC a la définition ASN.1 suivante :

```
CMCStatusInfo ::= SEQUENCE {
    cMCStatus       CMCStatus,
    bodyList        BodyPartList,
    statusString    UTF8String FACULTATIF,
    otherInfo       CHOIX {
      failInfo      CMCFailInfo,
      pendInfo      PendInfo } FACULTATIF
}
```

Les champs de CMCStatusInfo ont la signification suivante :

cMCStatus contient la valeur d'état retournée. Les détails sont au paragraphe 6.1.3.

bodyList contient la liste des commandes ou autres éléments auxquels la valeur d'état s'applique. Si une erreur est retournée pour une simple demande PKI, ce champ contient un seul entier de valeur 1.

statusString contient des informations de description supplémentaires. Cette chaîne est lisible par l'homme.

otherInfo fournit des informations supplémentaires qui complètent le code d'état de CMC retourné dans le champ cMCStatus.

failInfo est décrit au paragraphe 6.1.4. Il fournit un code d'erreur qui détaille la défaillance qui s'est produite. Ce choix n'est présent que si cMCStatus est "échec".

pendInfo utilise la structure ASN.1 PendInfo du paragraphe 6.1.1. Il contient des informations sur quand et comment le client devrait demander les résultats de cette demande. Le champ pendInfo DOIT être rempli pour une valeur de cMCStatus de "en instance" ou "partiel". Les détails se trouvent au paragraphe 6.1.1 (Commande Informations étendues d'état de CMC) et au paragraphe 6.13 (Commande Interrogation en instance).

Si le champ cMCStatus est "succès", la commande Informations d'état de CMC PEUT être omise sauf si elle est le seul élément de la réponse. Si aucun état n'existe pour une demande PKI simple ou complète, la valeur de "succès" est supposée.

6.1.3 Valeurs de CMCStatus

CMCStatus est un champ dans les commandes Informations étendues d'état de CMC et Informations d'état de CMC. Ce champ contient un code qui représente le succès ou l'échec d'une opération spécifique. CMCStatus a la structure ASN.1 suivante :

```
CMCStatus ::= ENTIER {
    succès                (0)
    -- réservé            (1)
    échec                 (2)
    en instance           (3)
    non pris en charge    (4)
    confirmation exigée   (5)
    pop exigée            (6)
    partiel               (7)
}
```

Les valeurs de CMCStatus ont la signification suivante :

succès indique que la demande a été accordée ou que l'action a été réalisée.

échec indique que la demande n'a pas été accordée ou que l'action n'a pas été réalisée. Plus d'informations sont incluses ailleurs dans la réponse.

en instance indique que la demande PKI n'a pas encore été traitée. Le demandeur est responsable de réinterroger pour cette demande PKI complète. "en instance" peut seulement être retourné pour les opérations de demande de certification.

non pris en charge indique que l'opération demandée n'est pas prise en charge.

confirmation exigée indique qu'une commande Confirmer l'acceptation du certificat (paragraphe 6.14) doit être retournée avant que le certificat puisse être utilisé.

POP exigé indique qu'une opération POP directe est requise (paragraphe 6.3.1.3).

partiel indique qu'une réponse PKI partielle est retournée. Le demandeur est chargé de répéter l'interrogation pour les portions non satisfaites de la demande PKI complète.

6.1.4. CMCFailInfo

CMCFailInfo est un champ dans les commandes Informations étendues d'état de CMC et Informations d'état de CMC. CMCFailInfo porte des informations plus détaillées pour l'interprétation d'une condition d'échec. Le champ CMCFailInfo a la structure ASN.1 suivante :

```
CMCFailInfo ::= ENTIER {
    badAlg              (0)
    badMessageCheck     (1)
    badRequest          (2)
    badTime             (3)
```

```
    badCertId            (4)
    unsupportedExt       (5)
    mustArchiveKeys      (6)
    badIdentity          (7)
    popRequired          (8)
    popFailed            (9)
    noKeyReuse           (10)
    internalCAError      (11)
    tryLater             (12)
    authDataFail         (13)
}
```

Les valeurs de CMCFailInfo ont la signification suivante :
badAlg indique un algorithme non reconnu ou non pris en charge.
badMessageCheck indique l'échec de la vérification d'intégrité.
badRequest indique que la transaction n'est pas permise ou prise en charge.
badTime indique que le champ d'heure du message n'est pas suffisamment proche de l'heure du système.
badCertId indique qu'aucun certificat correspondant aux critères fournis n'a pu être identifié.
unsupportedExt indique qu'une extension X.509 demandée n'est pas prise en charge par la CA receveuse.
mustArchiveKeys indique que du matériel de clé privée doit être fourni.
badIdentity indique que le contrôle d'identification a échoué à la vérification.
popRequired indique que le serveur exige une preuve de POP avant de produire le certificat.
popFailed indique que le traitement de POP a échoué.
noKeyReuse indique que la politique du serveur ne permet pas la réutilisation de clés.
internalCAError indique que la CA a eu une défaillance interne inconnue.
tryLater indique que le serveur n'accepte pas de demandes pour l'instant et que le client devrait réessayer plus tard.
authDataFail indique qu'une défaillance s'est produite durant le traitement des données authentifiées.

Si des raisons d'échec supplémentaires sont nécessaires, elles DEVRAIENT utiliser l'élément ExtendedFailureInfo dans la commande Informations étendues d'état de CMC. Cependant, pour des environnements clos elles peuvent être définies en utilisant ce type. Ces codes DOIVENT être dans la gamme de 1000 à 1999.

6.2 Commandes d'identification et de preuve d'identité

Certaines CA et RA exigent qu'une preuve d'identité soit incluse dans une demande de certification. Il existe de nombreuses façons différentes de le faire avec différents degrés de sécurité et fiabilité. La plupart sont proches de la demande d'une banque de fournir le nom de jeune fille de votre mère comme forme de preuve d'identité. Le raisonnement derrière l'exigence d'une preuve d'identité peut être trouvé dans l'Appendice C de la [RFC4211].

La CMC fournit une méthode pour prouver l'identité d'un client sur la base d'un secret partagé entre client et serveur. Si les clients prennent en charge la demande PKI complète, ils DOIVENT mettre en œuvre cette méthode de preuve d'identité (paragraphe 6.2.2). Les serveurs DOIVENT fournir cette méthode, mais PEUVENT de plus prendre en charge des méthodes bilatérales de force similaire.

Le présent document fournit aussi une commande Identification (paragraphe 6.2.3). Cette commande est une méthode simple pour permettre à un client de déclarer qui il est au serveur. Généralement, un secret partagé ET un identifiant de ce secret partagé sont passés du serveur au client. L'identifiant est placé dans la commande Identification, et le secret partagé est utilisé pour calculer la commande Preuve d'identité.

6.2.1 Commande Preuve d'identité version 2

La commande Preuve d'identité version 2 est identifiée par l'OID id-cmc-identityProofV2 ::= { id-cmc 34 }

La commande Preuve d'identité version 2 a la définition ASN.1 suivante :

```
IdentifyProofV2 ::= SEQUENCE {
   hashAlgID      AlgorithmIdentifier,
   macAlgID       AlgorithmIdentifier,
   witness        CHAÎNE D'OCTETS
}
```

Les champs de IdentityProofV2 ont la signification suivante :

hashAlgID est l'identifiant et les paramètres pour l'algorithme de hachage utilisé pour convertir le secret partagé en une clé pour l'algorithme de MAC.

macAlgID est l'identifiant et les paramètres pour l'algorithme de code d'authentification de message utilisé pour calculer la valeur du champ “witness”.

witness est la preuve d'identité.

La méthode requise commence par un transfert hors bande d'un jeton (le secret partagé). Le secret partagé devrait être généré de façon aléatoire. La distribution de ce jeton sort du domaine d'application de ce document. Le client utilise alors ce jeton pour preuve d'identité comme suit :

1. Le champ reqSequence de PKIData (codé exactement comme il apparaît dans la demande PKI complète, incluant le type de séquence et la longueur) est la valeur à valider.
2. Un hachage du secret partagé comme chaîne UTF8 est calculé en utilisant hashAlgID.
3. Un MAC est ensuite calculé en utilisant la valeur produite à l'étape 1 comme message et la valeur de l'étape 2 comme clé.
4. Le résultat de l'étape 3 est alors codé comme valeur de témoin dans la commande Preuve d'identité Version 2.

Quand le serveur vérifie la commande Preuve d'identité Version 2, il calcule la valeur du MAC de la même façon et la compare à la valeur du témoin dans la demande PKI.

Si la vérification d'une commande Preuve d'identité Version 2 par un serveur échoue, la valeur CMCFailInfo DOIT être présente dans la réponse PKI complète et DOIT avoir une valeur de badIdentity.

La réutilisation du secret partagé sur les essais de demande de certification permet au client et au serveur de conserver la même vue des valeurs acceptables de preuve d'identité. Cependant, la réutilisation du secret partagé peut éventuellement ouvrir la voie à certains types d'attaques.

Les mises en œuvre DOIVENT être capables de prendre en charge des jetons d'au moins 16 caractères. Des lignes directrices sur la quantité d'entropie réellement obtenue d'une longueur de jeton données sur la base des jeux de caractères peut être trouvées à l'Appendice A de [PASSWORD].

6.2.2 Commande Preuve d'identité

La commande Preuve d'identité est identifiée par l'OID id-cmc-identityProof ::= { id-cmc 3 }

La commande Preuve d'identité a la définition ASN.1 suivante :

IdentifyProof ::= CHAÎNE D'OCTETS

Cette commande est traitée de la même façon que la commande Preuve d'identité version 2. Dans ce cas, l'algorithme de hachage est fixé à SHA-1 et l'algorithme de MAC est fixé à HMAC-SHA1.

6.2.3 Commande Identification

Facultativement, les serveurs PEUVENT exiger l'inclusion de la commande non protégée Identification avec une commande Preuve d'identification. La commande Identification est destinée à contenir une chaîne de texte qui aide le serveur à localiser le secret partagé nécessaire pour valider le contenu de la commande Preuve d'identité. Si la commande Identification est incluse dans la demande PKI complète, la déduction de la clé dans l'étape 2 (du paragraphe 6.2.1) est altérée de sorte que le hachage de l'enchaînement du secret partagé et de la valeur d'identité en UTF8 (sans le type et les octets de longueur) sont hachés plutôt que juste le secret partagé.

La commande Identification est identifiée par l'OID id-cmc-identification ::= { id-cmc 2 }

La commande Identification a la définition ASN.1 suivante :

Identification ::= UTF8String

6.2.4 Génération matérielle du jeton de secret partagé

Le secret partagé entre l'EE et le serveur est parfois calculé en utilisant un appareil matériel qui génère une série de jetons. L'EE peut donc prouver son identité en transférant ce jeton en clair avec une chaîne de nom. Le protocole ci-dessus peut être utilisé avec un appareil matériel de jeton de secret partagé avec les modifications suivantes :

1. La commande Identification DOIT être incluse et DOIT contenir le jeton généré par le matériel.
2. La valeur du secret partagé utilisée ci-dessus est la même que le jeton généré par le matériel.
3. Toutes les demandes de certification DOIVENT avoir un nom de sujet, et le nom de sujet DOIT contenir les champs requis pour identifier le détenteur de l'appareil de jeton de matériel.
4. La demande de certification entière DOIT être protégée d'une certaine façon pour empêcher l'espionnage. Bien que le jeton soit limité en temps, il ne peut pas être permis à un espion actif d'extraire le jeton et de le soumettre dans une demande de certification différente avec la même valeur de jeton.

6.3 Liaison de l'identité et des informations POP

Dans une demande PKI complète, les informations d'identité sur le client sont portées dans la signature des SignedData contenant toutes les demandes de certification. Les informations de preuve de possession pour les paires de clés, sont cependant portées séparément pour chaque demande de certification PKCS n° 10 ou CRMF. (Pour les clés capables de générer une signature numérique, la POP est fournie par la signature sur la demande PKCS n° 10 ou CRMF. Pour les clés de chiffrement seul, les commandes décrites au paragraphe 6.7 sont utilisées.) Afin d'empêcher des attaques de style substitution, le protocole doit garantir que la même entité a généré les informations de POP et de preuve d'identité.

Ce paragraphe décrit deux mécanismes pour lier l'identité et les informations de POP : des valeurs de témoin déduites cryptographiquement du secret partagé (paragraphe 6.3.1.3) et le nom distinctif (DN, *distinguished name*) du secret partagé/sujet correspondant (paragraphe 6.3.2). Les clients et serveurs DOIVENT prendre en charge la technique de valeur de témoin. Les clients et serveurs PEUVENT prendre en charge la correspondance du DN de secret partagé/sujet ou d'autres techniques bilatérales de force similaire. L'idée derrière les deux mécanismes est de forcer le client à signer des données dans chaque demande de certification qui peuvent être directement associées au secret partagé ; cela va déjouer les tentatives d'inclure des demandes de certification provenant d'entités différentes dans une seule demande PKI complète.

6.3.1 Liaison cryptographique

La première technique qui lie les informations d'identité et de POP force le client à inclure des éléments d'information dérivés cryptographiquement du secret partagé comme une extension signée au sein de chaque demande de certification (PKCS n° 10 ou CRMF).

6.3.1.1 Commande Témoin de liaison POP version 2

La commande Témoin de liaison POP version 2 est identifiée par l'OID id-cmc-popLinkWitnessV2 ::= { id-cmc 33 }

La commande Témoin de liaison POP version 2 a la définition ASN.1 suivante :

```
PopLinkWitnessV2 ::= SEQUENCE {
   keyGenAlgorithm   AlgorithmIdentifier,
   macAlgorithm      AlgorithmIdentifier,
   witness           CHAÎNE D'OCTETS
}
```

Les champs de PopLinkWitnessV2 ont la signification suivante :

keyGenAlgorithm contient l'algorithme utilisé pour générer la clé pour l'algorithme de MAC. Cela va généralement être un algorithme de hachage, mais pourrait être un algorithme plus complexe.

macAlgorithm contient l'algorithme utilisé pour créer la valeur du témoin.

witness contient la valeur de témoin calculée.

Cette technique est utile si des DN d'objet nuls sont utilisés (parce que, par exemple, le serveur peut générer le DN sujet pour le certificat sur la seule base du secret partagé). Le traitement commence quand le client reçoit hors bande le secret partagé provenant du serveur. Le client calcule alors les valeurs suivantes :

1. Le client génère une chaîne d'octets aléatoire, R, qui DEVRAIT faire au moins 512 bits.

2. La clé est calculée à partir du secret partagé en utilisant l'algorithme de keyGenAlgorithm.
3. Un MAC est ensuite calculé sur la valeur aléatoire produite à l'étape 1, en utilisant la clé calculée à l'étape 2.
4. La valeur aléatoire produite à l'étape 1 est codée comme valeur de la commande Liaison POP aléatoire. Cette commande DOIT être incluse dans la demande PKI complète.
5. La valeur de MAC produite à l'étape 3 est placée dans la commande Témoin de liaison POP ou le champ Témoin de la commande Témoin de liaison POP V2.
   * Pour CRMF, la commande Témoin de liaison POP/Témoin de liaison POP V2 est incluse dans le champ Commandes de la structure CertRequest.
   * Pour PKCS n° 10, la commande Témoin de liaison POP/Témoin de liaison POP V2 est incluse dans le champ Attributs de la structure CertificationRequestInfo.

À réception, les serveurs DOIVENT vérifier que chaque demande de certification contient une copie de la commande Témoin de liaison POP/Témoin de liaison POP V2 et que sa valeur a été, en utilisant la méthode ci-dessus, déduite du secret partagé et de la chaîne aléatoire incluse dans la commande Liaison POP aléatoire.

La commande Identification (voir au paragraphe 6.2.3) ou le DN sujet d'une demande de certification, peut être utilisée pour aider à identifier quel secret partagé a été utilisé.

6.3.1.2 Commande Témoin de liaison POP

La commande Témoin de liaison POP est identifiée par l'OID id-cmc-popLinkWitness ::= { id-cmc 23 }

La commande Témoin de liaison POP a la définition ASN.1 suivante :

PopLinkWitness ::= CHAÎNE D'OCTETS

Pour cette commande, SHA-1 est utilisé comme algorithme de génération de clé. HMAC-SHA1 est utilisé comme algorithme de MAC.

6.3.1.3 Commande Liaison POP aléatoire

La commande Liaison POP aléatoire est identifiée par l'OID id-cmc-popLinkRandom ::= { id-cmc 22 }

La commande Liaison POP aléatoire a la définition ASN.1 suivante :

PopLinkRandom ::= CHAÎNE D'OCTETS

6.3.2 Liaison secret partagé / DN sujet

La seconde technique pour lier les informations d'identité et de POP est de lier un nom distinctif de sujet particulier (DN sujet) aux secrets partagés qui sont distribués hors bande et d'exiger que les clients qui utilisent le secret partagé pour prouver leur identité incluent ce DN sujet exact dans chaque demande de certification. Il est prévu que beaucoup des connexions client-serveur qui utilisent la preuve d'identité fondée sur le secret partagé vont utiliser ce mécanisme. (Il est courant de ne pas omettre les informations de DN sujet de la demande de certification.)

Quand le secret partagé est généré et transféré hors bande pour initier le processus d'enregistrement (paragraphe 6.2) un DN sujet particulier est aussi associé au secret partagé et communiqué au client. (Le DN sujet généré DOIT être unique par entité en accord avec la politique de la CA ; un DN sujet nul ne peut pas être utilisé. Une pratique courante pourrait être de placer la valeur d'identification au titre du DN sujet.) Quand le client génère la demande PKI complète, il DOIT utiliser ces deux éléments d'information comme suit :

1. Le client DOIT inclure le DN sujet spécifique qu'il a reçu avec le secret partagé comme nom de sujet dans chaque demande de certification (PKCS n° 10 et/ou CRMF) dans la demande PKI complète. Les noms de sujet dans les demandes de certification NE DOIVENT PAS être nuls.

2. Le client DOIT inclure une commande Preuve d'identité (paragraphe 6.2.2) ou Preuve d'identité version 2 (paragraphe 6.2.1) dérivée du secret partagé, dans la demande PKI complète.

Le serveur qui reçoit ce message DOIT (a) valider la commande Preuve d'identité et ensuite, (b) vérifier que le DN sujet inclus dans chaque demande de certification correspond à celui associé au secret partagé. Si l'une ou l'autre de ces vérifications échoue, la demande de certification DOIT être rejetée.

6.3.3 Messages de renouvellement et de changement de clés

Quand on fait un renouvellement ou un changement de clé de demande de certification, lier les informations d'identité et de POP est simple. Le client copie le DN sujet pour un certificat signant actuel dans le champ Nom de sujet de chaque demande de certification faite. La POP pour chaque demande de certification va maintenant couvrir ces informations. La couche de signature la plus externe est créée en utilisant le certificat signant actuel, qui permet que l'identité originale soit associée à la demande de certification. Comme le nom dans le certificat signant actuel et les noms dans les demandes de certification correspondent, la liaison nécessaire a été réalisée.

6.4 Commande Retour des données

La commande Retour des données permet aux clients d'envoyer des données arbitraires (généralement un type d'informations d'état interne) au serveur et d'avoir le retour des données au titre de la réponse PKI complète. Les données placées dans une commande Retour des données sont considérées comme opaques pour le serveur. La même commande est utilisée pour les demandes et réponses PKI complètes. Si la commande Retour des données apparaît dans une demande PKI complète, le serveur DOIT les retourner au titre de la réponse PKI.

Au cas où les informations de la commande Retour des données devraient être confidentielles, il est prévu que le client va appliquer un type de chiffrement aux données contenues, mais les détails de cela sortent du domaine de cette spécification.

La commande Retour des données est identifié par l'OID id-cmc-dataReturn ::= { id-cmc 4 }

La commande Retour des données a la définition ASN.1 suivante :

DataReturn ::= CHAÎNE D'OCTETS

Un client pourrait utiliser cette commande pour placer un identifiant marquant la source exacte du matériel de clé privée. Cela pourrait être l'identifiant d'un appareil matériel contenant la clé privée.

6.5 Commandes de modification de certificat de RA

Ces commandes existent pour que les RA soient capables de modifier le contenu d'une demande de certification. Des modifications pourraient être nécessaires pour des raisons variées. Elles incluent l'ajout d'extensions de certificat ou la modification de noms de sujet et/ou de noms de sujet de remplacement.

Deux commandes existent à cette fin. La première commande, Modifier la demande de certification (paragraphe 6.5.1) permet à la RA de remplacer ou retirer tout champ du certificat. La seconde commande, Ajouter des extensions (paragraphe 6.5.2) permet seulement l'ajout d'extensions.

6.5.1 Commande Modifier la demande de certification

La commande Modifier la demande de certification est utilisée par les RA pour changer les champs dans un certificat demandé.

La commande Modifier la demande de certification est identifiée par l'OID id-cmc-modCertTemplate ::= { id-cmc 31 }

Modifier la demande de certification a la définition ASN.1 suivante :

```
ModCertTemplate ::= SEQUENCE {
   pkiDataReference     BodyPartPath,
   certReferences       BodyPartList,
   replace              BOOLÉEN PAR DÉFAUT VRAI,
   certTemplate         CertTemplate
}
```

Les champs dans ModCertTemplate ont la signification suivante :

pkiDataReference est le chemin de la demande PKI contenant la ou les demandes de certification à modifier.

certReferences se réfère à une ou plusieurs demandes de certification dans la demande PKI référencée par pkiDataReference à modifier. Chaque BodyPartID de la séquence certReferences DOIT être égale au bodyPartID d'une TaggedCertificationRequest (PKCS n° 10) ou au certReqId de la CertRequest au sein d'une CertReqMsg (CRMF). Par définition, les extensions de certificat incluses dans le champ certTemplate sont appliquée à chaque demande de certification référencée dans la séquence certReferences. Si une demande correspondant à bodyPartID ne peut pas être trouvée, une CMCFailInfo avec une valeur de badRequest est retournée qui fait référence à cette commande.

replace spécifie si la demande de certification cible est à modifier en remplaçant ou en supprimant des champs. Si la valeur est VRAI, les données de cette commande remplacent les données dans la demande de certification cible. Si la valeur est FAUX, les données dans la demande de certification cible sont supprimées. L'action est légèrement différente pour le champ Extensions de certTemplate ; chaque extension est traitée individuellement plutôt que comme une seule unité.

certTemplate est un objet Gabarit de certificat [RFC4211]. Si un champ est présent et si "replace" est VRAI, il remplace ce champ dans la demande de certification. Si le champ est présent et si "replace" est FAUX, le champ dans la demande de certification est supprimé. Si le champ est absent, aucune action n'est effectuée. Chaque extension est traitée comme un seul champ.

Les serveurs DOIVENT être capables de traiter toutes les extensions définies, mais non interdites, dans la [RFC3280]. Les serveurs ne sont pas obligés d'être capables de traiter chaque extension X.509v3 transmise en utilisant ce protocole, ni ne sont obligés d'être capables de traiter d'autres extensions privées. Les serveurs ne sont pas obligés de mettre toutes les extensions demandées par la RA dans un certificat. Il est permis aux serveurs de modifier les extensions demandées par la RA. Les serveurs NE DOIVENT PAS altérer une extension au point de renverser la signification d'une extension demandée par le client. Si une demande de certification est refusée à cause de l'incapacité à traiter une extension demandée et si une réponse PKI complète est retournée, le serveur DOIT retourner une valeur CMCFailInfo avec la valeur de unsupportedExt.

Si une demande de certification est la cible de plusieurs commandes Modifier la demande de certification, le comportement est :

- Si la commande A existe dans une couche qui contient la couche de la commande B, la commande A DOIT outrepasser la commande B. En d'autres termes, les commandes devraient être appliquées de la couche la plus interne à la couche la plus externe.
- Si la commande A et la commande B sont dans le même PKIData (c'est-à-dire, la même couche d'enveloppement) l'ordre d'application n'est pas déterminé.

Le même ordre d'application est utilisé si une demande de certification est la cible d'une commande Modifier la demande de certification et d'une commande Ajouter des extensions.

6.5.2 Commande Ajouter des extensions

La commande Ajouter des extensions a été déconseillée en faveur de la commande Modifier la demande de certification. Elle a été remplacée afin que des champs de la demande de certification autres que des extensions puissent être modifiés.

La commande Ajouter des extensions est utilisée par les RA pour spécifier des extensions supplémentaires à celles qui sont incluses dans les certificats.

La commande Ajouter des extensions est identifié par l'OID id-cmc-addExtensions ::= { id-cmc 8 }

La commande Ajouter des extensions a la définition ASN.1 suivante :

```
AddExtensions ::= SEQUENCE {
   pkiDataReference     BodyPartID,
   certReferences       SEQUENCE DE BodyPartID,
   extensions           SEQUENCE DE Extension
}
```

Les champs dans AddExtensions ont la signification suivante :

pkiDataReference contient l'identité de partie de corps de la demande de certification incorporée.

certReferences est une liste de références à une ou plusieurs des demandes de certification contenues dans une PKIData. Chaque identifiant de partie de corps de la séquence certReferences DOIT être égal au bodyPartID d'une TaggedCertificationRequest (PKCS n° 10) ou au certReqId de la CertRequest au sein d'un CertReqMsg (CRMF). Par définition, les extensions mentionnées sont à appliquer à chaque demande de certification référencée dans la séquence certReferences. Si une demande de certification correspondant à bodyPartID ne peut pas être trouvée, le CMCFailInfo avec une valeur de badRequest est retournée en faisant référence à cette commande.

extensions est une séquence d'extensions à appliquer aux demandes de certification référencées.

Les serveurs DOIVENT être capables de traiter toutes les extensions définies, mais non interdites, dans la [RFC3280]. Les serveurs ne sont pas obligés d'être capables de traiter chaque extension X.509v3 transmise en utilisant ce protocole, ni ne sont obligés d'être capables traiter d'autres extensions privées. Les serveurs ne sont pas obligés de mettre toutes les extensions demandées par la RA dans un certificat. Il est permis aux serveurs de modifier les extensions demandées par la RA. Les serveurs NE DOIVENT PAS altérer une extension demandée par un client au point d'en inverser la signification. Si une demande de certification est refusée à cause de l'incapacité de traiter une extension demandée et si une réponse est retournée, le serveur DOIT retourner une CMCFailInfo avec la valeur de unsupportedExt.

Si plusieurs commandes Ajouter des extensions existent dans une demande PKI complète, le comportement exact est laissé à la politique de la CA. Cependant, il est recommandé que la politique suivante soit utilisée. Ces règles vont être appliquées aux extensions individuelles au sein d'une commande Ajouter des extensions (par opposition à une approche de "tout ou rien").

1. Si le conflit est dans une seule PKIData, la demande de certification va être rejetée avec une valeur de CMCFailInfo de badRequest.

2. Si le conflit est entre différentes PKIData, la version la plus externe de l'extension va être utilisée (en permettant à la RA d'outrepasser l'extension demandée).

6.6 Commande Identifiant de transaction et commandes Nom occasionnel d'envoyeur et receveur

Les transactions sont identifiées et suivies avec un identifiant de transaction. Si il est utilisé, les clients génèrent les identifiants de transaction et conservent leur valeur jusqu'à ce que le serveur réponde avec une réponse PKI complète qui termine la transaction. Les serveurs incluent de même les identifiants de transaction reçus dans la réponse PKI complète.

La commande Identifiant de transaction est identifiée par l'OID id-cmc-transactionId ::= { id-cmc 5 }

La commande Identifiant de transaction a la définition ASN.1 suivante :

TransactionId ::= ENTIER

La commande Identifiant de transaction identifie une certaine transaction. Elle est utilisée par le client et le serveur pour gérer l'état d'une opération. Les clients PEUVENT inclure une commande Identifiant de transaction dans une demande. Si la demande originale contient une commande Identifiant de transaction, toutes les demandes et réponses suivantes DOIVENT inclure la même commande Identifiant de transaction.

La protection contre la répétition est prise en charge par l'utilisation des commandes Nom occasionnel d'envoyeur et de receveur. Si des noms occaionnels sont utilisés, dans le premier message d'une transaction, une commande Nom occasionnel de receveur n'est pas transmise ; une commande Nom occasionnel d'envoyeur est incluse par le générateur de la transaction et conservée pour référence ultérieure. Le receveur d'une commande Nom occasionnel d'envoyeur reflète cette valeur au générateur comme une commande Nom occasionnel de receveur et inclut sa propre commande Nom occasionnel d'envoyeur. À réception de cette réponse par le générateur de la transaction, celui-ci compare la valeur de la commande Nom occasionnel de receveur à la valeur qu'il a conservée. Si les valeurs correspondent, le message peut être accepté pour un traitement de sécurité ultérieur. La valeur reçue pour une commande Nom occasionnel d'envoyeur est aussi conservée pour être incluse dans le prochain message associé à la même transaction.

Les commandes Nom occasionnel d'envoyeur et Nom occasionel de receveur sont identifiées par les OID :
id-cmc-senderNonce ::= { id-cmc 6 }
id-cmc-recipientNonce ::= { id-cmc 7 }

La commande Nom occasionnel d'envoyeur a la définition ASN.1 suivante :

SenderNonce ::= CHAÎNE D'OCTETS

La commande Nom occasionnel de receveur a la définition ASN.1 suivante :

RecipientNonce ::= CHAÎNE D'OCTETS

Les clients PEUVENT inclure une commande Nom occasionnel d'envoyeur dans la demande PKI initiale. Si un message inclut une commande Nom occasionnel d'envoyeur, la réponse DOIT inclure la valeur transmise de la précédente commande Nom occasionnel d'envoyeur reçue comme commande Nom occasionnel de receveur et inclure une nouvelle valeur comme sa propre commande Nom occasionnel d'envoyeur.

6.7 Commandes POP chiffrées et déchiffrées

Les serveurs PEUVENT exiger que cette méthode POP ne soit utilisée que si une autre méthode POP n'est pas disponible. Les serveurs DEVRAIENT rejeter toutes les demandes de certification contenues dans une PKIData si une POP demandée manque pour tout élément au sein de PKIData.

De nombreux serveurs exigent la preuve que l'entité qui a généré la demande de certification possède réellement le composant privé correspondant de la paire de clés. Pour les clés qui peuvent être utilisées comme clés de signature, signer la demande de certification avec la clé privée sert de POP sur cette paire de clés. Avec les clés qui peuvent seulement être utilisées pour des opérations de chiffrement, la POP DOIT être effectuée en forçant le client à déchiffrer une valeur. Voir à la Section 5 de la [RFC4211] une discussion détaille de POP.

Par nécessité, les clés POP seulement de chiffrement ne peuvent pas être faites en un aller-retour, car il y a quatre étapes distinctes :

1. Le client dit au serveur le composant public d'une nouvelle paire de clés de chiffrement.

2. Le serveur envoie au client un défi de POP, chiffré avec la clé publique de chiffrement présentée.

3. Le client déchiffre le défi POP en utilisant la clé privée qui correspond à la clé publique présentée et renvoie le texte en clair au serveur.

4. Le serveur valide le défi POP déchiffré et continue le traitement de la demande de certification.

La CMC définit deux commandes différentes. La première traite du défi chiffré envoyé du serveur à l'utilisateur dans l'étape 2. La seconde traite du défi déchiffré envoyé du client au serveur dans l'étape 3.

La commande POP chiffrée est utilisée pour envoyer le défi chiffré du serveur au client au titre de la PKIResponse. (Noter qu'on suppose que le message envoyé à l'étape 1 ci-dessus est une demande PKI complète et que la réponse dans l'étape 2 est une réponse PKI complète incluant des CMCFailInfo spécifiant qu'une POP est explicitement exigée, et fournissant le défi de POP dans la commande encryptedPOP.)

La commande POP chiffrée est identifiée par l'OID id-cmc-encryptedPOP ::= { id-cmc 9 }

La commande POP chiffrée a la définition ASN.1 suivante :

```
EncryptedPOP ::= SEQUENCE {
    request          TaggedRequest,
    cms              ContentInfo,
    thePOPAlgID      AlgorithmIdentifier,
    witnessAlgID     AlgorithmIdentifier,
    witness          CHAÎNE D'OCTETS
}
```

La commande POP déchiffrée est identifié par l'OID id-cmc-decryptedPOP ::= { id-cmc 10 }

La commande POP déchiffrée a la définition ASN.1 suivante :

```
DecryptedPOP ::= SEQUENCE {
```

```
    bodyPartID       BodyPartID,
    thePOPAlgID      AlgorithmIdentifier,
    thePOP           CHAÎNE D'OCTETS
}
```

L'algorithme POP chiffrée fonctionne comme suit :

1. Le serveur génère au hasard la valeur de preuve de POP et l'associe à la demande.
2. Le serveur retourne la commande POP chiffrée avec les champs suivants établis :

   request est la demande de certification originale (elle est incluse ici afin que le client n'ait pas besoin de garder une copie de la demande).

   cms est une EnvelopedData, le type de contenu encapsulé étant id-data et le contenu étant la valeur de la preuve de POP ; cette valeur doit être assez longue pour qu'on ne puisse pas inverser la valeur à partir du hachage du témoin. Si la demande de certification contient une extension Identifiant de clé de sujet (SKI, *Subject Key Identifier*) l'identifiant du receveur DEVRAIT alors être le SKI. Si la forme issuerAndSerialNumber est utilisée, le IssuerName DOIT être codé comme NUL et le SerialNumber comme le bodyPartID de la demande de certification.

   thePOPAlgID identifie l'algorithme à utiliser pour calculer la valeur de retour de POP.

   witnessAlgID identifie l'algorithme de hachage utilisé sur la valeur de preuve de POP pour créer le champ "witness".

   witness est la valeur hachée de la valeur de la preuve de POP.

3. Le client déchiffre le champ cms pour obtenir la valeur de preuve de POP. Le client calcule H (valeur de preuve de POP) en utilisant le witnessAlgID et le compare à la valeur de witness. Si les valeurs ne coïncident pas ou si le déchiffrement ne réussit pas, le client DOIT interrompre le processus d'adhésion. Le client interrompt le processus en envoyant une demande contenant une commande Informations d'état de CMC avec la valeur de CMCFailInfo de popFailed.

4. Le client crée la commande POP déchiffrée au titre d'une nouvelle PKIData. Les champs dans DecryptedPOP sont :

   bodyPartID se réfère à la demande de certification dans la nouvelle demande PKI.

   thePOPAlgID est copié de encryptedPOP.

   thePOP contient la preuve de possession. Cette valeur est calculée par thePOPAlgID en utilisant la valeur de preuve de POP et la demande.

5. Le serveur recalcule alors la valeur de thePOP à partir de la valeur en antémémoire et de la demande et la compare à la valeur de thePOP. Si les valeurs ne correspondent pas, le serveur NE DOIT PAS produire le certificat. Le serveur PEUT produire un nouveau défi ou PEUT aussi faire échouer la demande.

Quand on définit les algorithmes pour thePOPAlgID et witnessAlgID, il faut veiller à s'assurer que le résultat de witnessAlgID n'est pas une valeur utile pour court-circuiter le calcul avec thePOPAlgID. La valeur de la preuve de POP est utilisée comme valeur secrète dans l'algorithme HMAC et la demande est utilisée comme les données. Si la valeur de preuve de POP est supérieure à 64 octets, seuls le 64 premiers octets de la valeur de preuve de POP sont utilisés comme secret.

Un problème potentiel avec l'algorithme ci-dessus est la quantité d'état qu'une CA a besoin de conserver afin de vérifier la valeur de POP retournée. On décrit ci après une des nombreuses façons possibles de traiter le problème en réduisant la quantité d'état conservé à la CA à une seule valeur (ou petit ensemble de valeurs).

1. Le serveur génère un germe aléatoire x, constant sur toutes les demandes. (La valeur de x va normalement être altérée sur une base régulière et conservée un bref instant ensuite.)

2. Pour une demande de certification R, le serveur calcule y = F(x,R). F peut être, par exemple, HMAC-SHA1(x,R). Tout ce qui est important pour l'absence d'état est qu'il soit calculable de façon cohérente avec seulement l'état constant connu x et la fonction F, les autres entrées venant de la structure de demande de certification. y ne devrait pas être prévisible sur la base de la connaissance de R, d'où l'utilisation d'une fonction unidirectionnelle comme HMAC-SHA1.

6.8 Commande Témoin POP de RA

Dans un scénario de demande de certification qui implique une RA, la CA peut permettre (ou exiger) que la RA effectue le protocole de POP avec l'entité qui a généré la demande de certification. Dans ce cas, la RA a besoin d'un moyen pour informer la CA qu'elle a fait la POP. La commande Témoin POP de RA traite ce problème.

La commande Témoin POP de RA est identifiée par l'OID id-cmc-lraPOPWitness ::= { id-cmc 11 }

La commande Témoin POP de RA a la définition ASN.1 suivante :

```
LraPopWitness ::= SEQUENCE {
   pkiDataBodyid   BodyPartID,
   bodyIds         SEQUENCE de BodyPartID
}
```

Les champs dans LraPOPWitness ont la signification suivante :

pkiDataBodyid contient l'identifiant de partie de corps des TaggedContentInfo incorporées contenant la demande PKI complète du client. pkiDataBodyid est réglé à 0 si la demande est dans les PKIData actuelles.

bodyIds est une liste des demandes de certification pour lesquelles la RA a effectué une authentification hors bande. La méthode d'authentification pourrait être l'archivage du matériel de clé privée, un défi-réponse, ou d'autres moyens.

Si un serveur de certification ne permet pas à une RA de faire la vérification de POP, il retourne une CMCFailInfo avec la valeur de popFailed. La CA NE DOIT PAS commencer un défi-réponse pour re-vérifier la POP elle-même.

6.9 Commande Obtenir un certificat

Tout ce qui est décrit dans ce paragraphe est de mise en œuvre facultative.

La commande Obtenir un certificat est utilisée pour restituer un certificat produit précédemment d'un répertoire de certificats. Une CA, une RA, ou un service indépendant peut fournir ce répertoire. Les clients supposés utiliser cette facilité sont ceux où un répertoire pleinement déployé serait infaisable ou indésirable.

La commande Obtenir un certificat est identifiée par l'OID id-cmc-getCert ::= { id-cmc 15 }

La commande Obtenir un certificat a la définition ASN.1 suivante :

```
GetCert ::= SEQUENCE {
    issuerName    GeneralName,
    serialNumber  ENTIER }
```

Les champs dans GetCert ont la signification suivante :

issuerName est le nom du producteur du certificat.

serialNumber identifie le certificat à restituer.

Le serveur qui répond à cette demande place le certificat demandé dans le champ "certificates" de SignedData. Si la commande Obtenir un certificat est la seule commande d'une demande PKI complète, la réponse devrait être une simple réponse PKI.

6.10. Commande Obtenir une CRL

Tout ce qui est décrit dans ce paragraphe est de mise en œuvre facultative.

La commande Obtenir une CRL est utilisée pour restituer des CRL d'un répertoire de CRL. Une CA, une RA, ou un service indépendant peuvent fournir ce répertoire. Les clients supposés utiliser cette facilité sont ceux où un répertoire pleinement déployé serait infaisable ou indésirable.

La commande Obtenir une CRL est identifiée par l'OID id-cmc-getCRL ::= { id-cmc 16 }

La commande Obtenir une CRL a la définition ASN.1 suivante :

```
GetCRL ::= SEQUENCE {
   issuerName    Name,
   cRLName       GeneralName       FACULTATIF,
   time          GeneralizedTime FACULTATIF,
   reasons       ReasonFlags       FACULTATIF }
```

Les champs dans GetCRL ont la signification suivante :

issuerName est le nom du producteur de la CRL.

cRLName peut être la valeur des CRLDistributionPoints dans le certificat de sujet ou valeur équivalente dans le cas où le certificat ne contient pas une telle valeur.

time est utilisé par le client pour spécifier entre potentiellement plusieurs productions de CRL celle dont la valeur de thisUpdate est inférieure mais la plus proche de l'heure spécifiée. En l'absence d'un composant "time", la CA retourne toujours la CRL la plus récente.

reasons est utilisé pour spécifier laquelle des CRL parmi celles partagées par raison de révocation. Les mises en œuvre devraient se souvenir que alors qu'une demande de révocation spécifique a un seul code CRLReason -- et par conséquent que les entrées dans la CRL vont avoir une seule valeur de code de CRLReason -- une seule CRL peut agréger des informations pour un ou plusieurs reasonFlags.

Un serveur qui répond à cette demande place la CRL demandée dans le champ crls des SignedData. Si la commande Obtenir une CRL est la seule commande dans une demande PKI complète, la réponse devrait être une simple réponse PKI.

6.11 Commande Demande de révocation

La commande Demande de révocation est utilisée pour demander qu'un certificat soit révoqué.

La commande Demande de révocation est identifié par l'OID id-cmc-revokeRequest ::= { id-cmc 17 }

La commande Demande de révocation a la définition ASN.1 suivante :

```
RevokeRequest ::= SEQUENCE {
   issuerName      Name,
   serialNumber    ENTIER,
   reason          CRLReason,
   invalidityDate  GeneralizedTime       FACULTATIF,
   sharedSecret    CHAÎNE D'OCTETS FACULTATIF,
   comment         UTF8string            FACULTATIF }
```

Les champs de RevokeRequest ont la signification suivante :

issuerName est le nom du producteur du certificat à révoquer.

serialNumber est le numéro de série du certificat à révoquer.

reason est le code suggéré de CRLReason pour laquelle le certificat est révoqué. La CA peut utiliser cette valeur à sa discrétion en construisant la CRL.

invalidityDate est la valeur suggérée pour l'extension de date d'invalidité de la CRL. La CA peut utiliser cette valeur à sa discrétion en construisant la CRL.

sharedSecret est une valeur secrète enregistrée par l'EE quand le certificat a été obtenu pour permettre la révocation d'un certificat en cas de perte de clé.

comment est un commentaire lisible par l'homme.

Pour qu'une demande de révocation soit fiable en cas de dispute, une forte preuve d'origine est exigée. Cependant, dans le cas où une EE a perdu l'usage de sa clé privée de signature, il est impossible à l'EE de produire une signature numérique (avant la certification d'une nouvelle paire de clés de signature). La commande Demande de révocation permet à l'EE d'envoyer à la CA un secret partagé qui peut être utilisé comme authentifiant de remplacement en cas de perte de l'usage de la clé privée de signature de l'EE. L'acceptabilité de cette pratique est une affaire de politique de sécurité locale.

Il est possible de signer la révocation pour le certificat perdu avec un certificat différent dans certaines circonstances. Un client peut signer une révocation pour une clé de chiffrement avec un certificat signant si les informations de nom correspondent. De même, un administrateur ou une RA peut se voir attribuer la capacité de révoquer le certificat d'un tiers. L'acceptation de la révocation par le serveur dépend dans ces cas de la politique locale.

Les clients DOIVENT fournir la capacité de produire une commande Demande de révocation signée numériquement. Les clients DEVRAIENT être capables de produire une commande Demande de révocation non signée contenant le secret partagé de l'EE (le message non signé consistant en des SignedData sans signature). Si un client fournit une auto révocation fondée sur le secret partagé, il DOIT être capable de produire une commande Demande de révocation contenant le secret partagé. Les serveurs DOIVENT être capables d'accepter les deux formes de demande de révocation.

La structure d'une demande de révocation non signée, fondée sur le secret partagé est une affaire de mise en œuvre locale. Le secret partagé n'a pas besoin d'être chiffré quand il est envoyé dans une commande Demande de révocation. Le secret partagé est à utilisation unique (c'est-à-dire, il est utilisé pour demander la révocation du certificat) et la connaissance publique du secret partagé après la révocation du certificat n'est pas un problème. Les clients doivent informer les utilisateurs que le même secret partagé NE DEVRAIT PAS être utilisé pour plusieurs certificats.

Une réponse PKI complète DOIT être retournée pour une demande de révocation.

6.12 Commandes Informations d'enregistrement et de réponse

La commande Informations d'enregistrement permet aux clients de passer des informations supplémentaires au titre d'une demande PKI complète.

La commande Informations d'enregistrement est identifiée par l'OID id-cmc-regInfo ::= { id-cmc 18 }

La commande Informations d'enregistrement a la définition ASN.1 suivante :

RegInfo ::= CHAÎNE D'OCTETS

Le contenu de ces données est fondé sur un accord bilatéral entre client et serveur.

La commande Informations de réponse permet à un serveur de retourner des informations supplémentaires au titre d'une réponse PKI complète.

La commande Informations de réponse est identifiée par l'OID id-cmc-responseInfo ::= { id-cmc 19 }

La commande Informations de réponse a la définition ASN.1 suivante :

ResponseInfo ::= CHAÎNE D'OCTETS

Le contenu de ces données est fondé sur un accord bilatéral entre client et serveur.

6.13 Commande Interrogation en instance

Dans certains environnements, les exigences de traitement pour une intervention manuelle ou autres vérifications d'identité peuvent retarder le retour du certificat. La commande Interrogation en instance permet au client d'interroger un serveur sur l'état d'une demande de certification en instance. Le serveur retourne un pendToken au titre des commandes Informations étendues d'état de CMC et Informations d'état de CMC (dans le champ otherInfo). Le client copie le pendToken dans la commande Interrogation en instance pour identifier au serveur la demande de certification correcte. Le serveur retourne une heure suggérée pour que le client interroge sur l'état d'une demande de certification en instance.

La commande Interrogation en instance est identifiée par l'OID id-cmc-queryPending ::= { id-cmc 21 }

La commande Interrogation en instance a la définition ASN.1 suivante :

QueryPending ::= CHAÎNE D'OCTETS

Si un serveur retourne une CMCStatusInfo en instance ou partielle (la transaction est encore en instance) les otherInfo PEUVENT être omises. Si les otherInfo ne sont pas omises, la valeur de "pendInfo" DOIT être la même que celle de la valeur originale des pendInfo.

6.14 Commande Confirmation de l'acceptation du certificat

Certaines CA exigent que les clients donnent une confirmation positive que les certificats praduits à l'EE sont acceptables. La commande Confirmation de l'acceptation du certificat est utilisée à cette fin. Si l'information d'état de CMC sur une réponse PKI est confirmRequired, alors le client DOIT retourner une commande Confirmation de l'acceptation du certificat contenue dans une demande PKI complète.

Les clients DEVRAIENT attendre la réponse PKI du serveur que la confirmation a été reçue avant d'utiliser le certificat pour tout objet.

La commande Confirmation de l'acceptation du certificat est identifié par l'OID id-cmc-confirmCertAcceptance ::= { id-cmc 24 }

La commande Confirmation de l'acceptation du certificat a la définition ASN.1 suivante :

CMCCertId ::= IssuerAndSerialNumber

CMCCertId contient le nom du producteur et le numéro de série du certificat qui est accepté.

Les serveurs DOIVENT retourner une réponse PKI complète pour une commande Confirmation de l'acceptation du certificat.

Noter que si la CA inclut cette commande, il va y avoir deux allers-retours complets de messages.
1. Le client envoie la demande de certification à la CA.
2. La CA retourne une réponse PKI complète avec le certificat et cette commande.
3. Le client envoie une demande PKI complète à la CA avec une commande Informations étendues d'état de CMC qui accepte et une commande Confirmation d'acceptation de certificat ou une commande Informations étendues d'état de CMC qui rejette le certificat.
4. La CA envoie une réponse PKI complète au client avec une commande Informations étendues d'état de CMC de succès.

6.15 Commande Publier les ancres de confiance

La commande Publier les ancres de confiance permet la distribution d'ensembles d'ancres de confiance d'une autorité centrale à une EE. La même commande est aussi utilisée pour mettre à jour l'ensemble d'ancres de confiance. Les ancres de confiance sont distribuées sour la forme de certificats. Elles sont supposées, mais ce n'est pas exigé, être des certificats auto signés. Les informations sont extraites de ces certificats pour établir les entrées de l'algorithme de validation de certificat du paragraphe 6.1.1 de la [RFC3280].

La commande Publier les ancres de confiance est identifiée par l'OID id-cmc-trustedAnchors ::= { id-cmc 26 }

La commande Publier les ancres de confiance a la définition ASN.1 suivante :

```
PublishTrustAnchors ::= SEQUENCE {
   seqNumber       ENTIER,
   hashAlgorithm  AlgorithmIdentifier,
   anchorHashes   SEQUENCE DE CHAÎNE D'OCTETS
}
```

Les champs dans PublishTrustAnchors ont la signification suivante :

seqNumber est un entier qui indique la localisation dans une séquence de mises à jour.

hashAlgorithm est l'identifiant et les paramètres de l'algorithme de hachage utilisé pour calculer les valeurs du champ anchorHashes. Toutes les mises en œuvre DOIVENT utiliser SHA-1 pour ce champ.

anchorHashes sont les hachages pour les certificats qui sont à traiter comme des ancres de confiance par le client. Les certificats réels sont transportés dans le sac de certificats de la structure SignedData contenante.

Bien qu'il soit recommandé que l'envoyeur place les certificats qui sont traités comme de confiance dans la réponse PKI, cela n'est pas exigé car les certificats devraient pouvoir être obtenus en utilisant les techniques de découverte normales.

Avant d'accepter les changements d'ancres de confiance, un client DOIT au moins faire ce qui suit : valider la signature sur la réponse PKI à une ancre de confiance actuelle, vérifier avec la politique pour s'assurer que le signataire a la permission d'utiliser la commande, valider que l'heure de publication authentifiée dans la signature est proche de l'heure actuelle, et valider que le numéro de séquence est supérieur à celui utilisé précédemment.

Dans le cas où plusieurs agents publient un ensemble d'ancres de confiance, il relève de la politique locale de déterminer comment les différentes ancres de confiance devraient être combinées. Les clients DEVRAIENT être capables de traiter indépendamment la mise à jour de plusieurs ancres de confiance.

Note : les clients qui traitent cette commande doivent faire extrêmement attention lors de la validation que l'opération soit permise. Un traitement incorrect de cette commande permet à un attaquant de changer l'ensemble d'ancres de confiance chez le client.

6.16 Commande Données authentifiées

La commande Données authentifiées permet à un serveur de fournir des données en retour au client de manière authentifiée. Cette commande utilise la stucture Données authentifiées pour permettre la validation des données. Cette commande est utilisée lorsque le client a un secret partagé et un identifiant de secret avec le serveur, mais lorsque une ancre de confiance n'a pas encore été téléchargée sur le client de sorte qu'un certificat signant pour le serveur ne peut pas être validé. Le cas spécifique où cette commande a été créée pour être utilisée est avec la commande Publier les ancres de confiance (paragraphe 6.15) mais elle peut être utilisé aussi dans d'autres cas.

La commande Données authentifiées est identifiée par l'OID id-cmc-authData ::= { id-cmc 27 }

La commande Données authentifiées a la définition ASN.1 suivante :

AuthPublish ::= BodyPartID

AuthPublish est un identifiant de partie de corps qui se réfère à un membre de l'élément cmsSequence pour la réponse PKI ou Données de PKI actuelle. L'élément cmsSequence est AuthenticatedData. Le contenu encapsulé est un id-cct-PKIData. Les commandes dans la controlSequence doivent être traitées si l'authentification réussit. (Un exemple est la commande Publier les ancres de confiance du paragraphe 6.15.)

Si l'opération d'authentification échoue, les CMCFailInfo authDataFail sont retournées.

6.17 Commandes Regrouper les demandes et réponses

Ces commandes permettent à une RA de collecter plusieurs demandes dans une seule demande PKI complète et de la transmettre à une CA. Le serveur va alors traiter les demandes et retourner le résultat dans une réponse PKI complète.

La commande Regrouper les demandes est identifiée par l'OID id-cmc-batchRequests ::= {id-cmc 28}

La commande Regrouper les réponses est identifiée par l'OID id-cmc-batchResponses ::= {id-cmc 29}

Les deux commandes Regrouper les demandes et Regrouper les réponses ont la définition ASN.1 suivante :

BodyPartList ::= SEQUENCE DE BodyPartID

Les données associées à ces commandes sont un ensemble d'identifiants de partie de corps. Chaque demande/réponse est placée comme une entrée individuelle dans la cmcSequence de la nouvelle PKIData/PKIResponse. Les identifiants de partie de corps de ces entrées sont alors placés dans la liste de parties de corps associée à la commande.

Quand un serveur traite une commande Regrouper les demandes, il PEUT retourner les réponses dans une ou plusieurs réponses PKI. Une valeur de CMCStatus de "partiel" est retournée sur toutes les réponses PKI sauf la dernière. Le CMCStatus va être un succès si la commande Regrouper les demandes a été traitée ; les réponses sont créées avec leur propre code de CMCStatus. Les erreurs sur les demandes individuelles ne sont pas propagées jusqu'au niveau supérieur.

Quand une réponse PKI avec une valeur de CMCStatus de "partiel" est retournée, la commande Interrogation en instance (paragraphe 6.13) est utilisée pour restituer des résultats supplémentaires. L'état retourné inclut une heure suggérée après laquelle le client devrait demander les résultats supplémentaires.

6.18 Commande Informations de publication

La commande Informations de publication permet de modifier la publication de certificats déjà produits, pour publication et pour suppression de publication. Un usage courant de cette commande est de retirer un certificat existant de la publication durant une opération de changement de clés. Cette commande devrait toujours être traitée après la production des nouveaux certificats et les demandes de révocation. Cette comande ne devrait pas être traitée si la production d'un certificat a échoué.

La commande Informations de publication est identifiée par l'OID id-cmc-publishCert ::= { id-cmc 30 }

La commande Informations de publication a la définition ASN.1 suivante :

```
CMCPublicationInfo ::= SEQUENCE {
    hashAlg       AlgorithmIdentifier,
    certHashes    SEQUENCE of CHAÎNE D'OCTETS,
    pubInfo       PKIPublicationInfo

PKIPublicationInfo ::= SEQUENCE {
     action      ENTIER {
                   dontPublish    (0),
                   pleasePublish (1) },
     pubInfos  SEQUENCE TAILLE (1..MAX) DE SinglePubInfo FACULTATIF }

-- pubInfos NE DOIT PAS être présent si action est "dontPublish" (si action est "pleasePublish" et si pubInfos est omis,
"dontCare" est supposé)

SinglePubInfo ::= SEQUENCE {
    pubMethod    ENTIER {
      dontCare    (0)
      x500        (1)
      web         (2)
      ldap        (3) }
  pubLocation  GeneralName FACULTATIF }
}
```

Les champs dans CMCPublicationInfo ont la signification suivante :

hashAlg est l'identifiant d'algorithme de l'algorithme de hachage utilisé pour calculer les valeurs dans certHashes.

certHashes sont les hachages des certificats pour lesquels la publication va changer.

pubInfo est l'information de où et comment les certificats devraient être publiés. Les champs dans pubInfo (tirés de la [RFC4211]) ont la signification suivante :

action indique l'action que le service devrait effectuer. Elle a deux valeurs :

dontPublish indique que la PKI ne devrait pas publier le certificat (cela peut indiquer que le demandeur a l'intention de publier lui-même le certificat). dontPublish a la connotation supplémentaire de supprimer de la publication le certificat si il est déjà publié.

pleasePublish indique que la PKI PEUT publier le certificat en utilisant tout moyen qu'il choisit sauf si pubInfos est présent. L'omission de la commande Informations de publication de CMC résulte en le même comportement.

pubInfos indique comment (par exemple, X500, Web, adresse IP) la PKI DEVRAIT publier le certificat.

Un seul certificat NE DEVRAIT PAS apparaître dans plus d'une commande Informations de publication. Le comportement est indéfini si il le fait.

6.19 Commande Commande traitée

La commande Commande traitée permet à une RA d'indiquer aux processeurs de commandes suivants qu'une commande spécifique a déjà été traitée. Cela permet à une RA au milieu d'un flux de traitement de traiter une commande définie soit dans un contexte local, soit dans un document suivant.

La commande Commande traitée est identifiée par l'OID id-cmc-controlProcessed ::= { id-cmc 32 }

La commande Commande traitée a la définition ASN.1 suivante :

```
ControlList ::= SEQUENCE {
  bodyList       SEQUENCE TAILLE (1..MAX) DE BodyPartReference
}
```

bodyList est une série d'identifiants de partie de corps qui forment un chemin pour chacune des commandes qui ont été traitées par la RA. Cette commande n'est nécessaire que pour les commandes qui ne font pas partie de cette norme et donc vont causer une condition d'erreur d'un serveur qui tente de traiter une commande non définie dans ce document. Aucun état d'erreur n'est nécessaire car une erreur cause le retour par la RA de la demande au client avec l'erreur plutôt que de passer la demande au prochain serveur dans la liste des traitements.

7. Autorités d'enregistrement

La présente spécification permet l'utilisation des RA. Une RA se tient entre la EE et la CA. Du point de vue de l'EE, la RA appaaraît comme une CA, et pour le serveur, la RA apparaît comme un client. Les RA reçoivent les demandes PKI, effectuent le traitement local et ensuite les transmettent aux CA. Certains des types de traitement local qu'une RA peut effectuer incluent :

- o de grouper plusieurs demandes PKI,
- o d'effectuer des preuves de POP par défi/réponse,
- o d'ajouter des extensions de certificat privées ou normalisées à toutes les demandes de certification,
- o d'archiver le matériel de clé privée,
- o d'acheminer les demandes aux différentes CA.

Quand une RA reçoit une demande PKI, elle a trois options : elle peut transmettre la demande PKI sans modification, elle peut ajouter une nouvelle couche d'enveloppe à la demande PKI, ou elle peut supprimer une ou plusieurs couches existantes et ajouter une nouvelle couche d'enveloppe .

Quand une RA ajoute une nouvelle couche d'enveloppe à une demande PKI, elle crée une nouvelle PKIData. La nouvelle couche contient toutes les commandes requises (par exemple, si la RA fait la preuve de POP pour une clé de chiffrement ou la commande Ajouter des extensions pour modifier une demande PKI) et la demande PKI du client. La demande PKI du client est placée dans le champ cmsSequence si c'est une demande PKI complète et dans le champ reqSequence si c'est une simple demande PKI. Si une RA groupe plusieurs demandes PKI de client, alors chaque demande PKI de client est placée dans la localisation appropriée dans l'objet PKIData de la RA avec toutes les commandes pertinentes.

Si plusieurs RA sont sur le chemin entre la EE et la CA, cela va conduire à plusieurs couches d'enveloppe sur la demande.

En traitant une demande PKI, une RA NE DOIT PAS altérer des demandes de certification (PKCS n° 10 ou CRMF) car toute altération invaliderait la signature sur la demande de certification et donc la POP pour la clé privée.

Un exemple de ce à quoi cela ressemblerait est illustré par la figure suivante :

SignedData (par la RA)
  PKIData
    controlSequence
      déclaration de commandes ajoutées par la RA
    reqSequence
      Zéro, une ou plusieurs simples demandes PKI des clients
    cmsSequence
      Zéro, une ou plusieurs demandes PKI complètes des clients
        SignedData (signée par le client)
          PKIData

Dans certaines circonstances, une RA est obligée de supprimer des couches d'enveloppe. Les paragraphes qui suivent décrivent le traitement requis si des couches de chiffrement et des couches de signature doivent être supprimées.

7.1 Suppression du chiffrement

Deux cas exigent qu'une RA supprime ou change le chiffrement d'une demande PKI. Dans le premier cas, le chiffrement a été appliqué pour protéger la demande PKI entière contre des entités non autorisées. Si la CA n'a pas d'entrée d'informations de receveur dans la couche de chiffrement, la RA DOIT supprimer la couche de chiffrement. La RA PEUT ajouter une nouvelle couche de chiffrement avec ou sans ajouter de nouvelle couche de signature.

Le second changement de chiffrement qui peut être exigé est de changer le chiffrement à l'intérieur d'une couche de signature. Dans ce cas, la RA DOIT supprimer toutes les couches de signature contenant le chiffrement. Toutes les déclarations de commandes DOIVENT être fusionnées en accord avec les règles de la politique locale lorsque chaque couche de signature est supprimée et les commandes fusionnées résultantes DOIVENT être placées dans une nouvelle couche de signature fournie par la RA. Si la couche de signature fournie par la EE a aussi besoin d'être supprimée, la RA peut aussi supprimer cette couche.

7.2 Suppression d'une couche de signature

Il existe seulement deux instances où une RA devrait supprimer une couche de signature sur une demande PKI complète : si une couche de chiffrement doit être modifiée dans la demande, ou si une CA ne va pas accepter de délégation secondaire (c'est-à-dire, plusieurs signatures de RA). Dans toutes les autres situations, les RA NE DEVRAIENT PAS supprimer une couche de signature d'une demande PKI.

Si une RA supprime une couche de signature d'une demande PKI, toutes les déclarations de commandes DOIVENT être fusionnées en accord avec les règles de la politique locale. Les déclarations de commandes fusionnées résultantes DOIVENT être placées dans une nouvelle couche de signature fournie par la RA.

8. Considérations sur la sécurité

Des mécanismes pour déjouer les attaques en répétition peuvent être exigés en particulier des mises en œuvre de ce protocole selon l'environnement de fonctionnement. Dans les cas où la CA conserve des informations d'état significatives, les attaques en répétition peuvent être détectées sans l'inclusion des mécanismes facultatifs de nom occasionnel. Les mises en œuvre de ce protocole doivent considérer avec attention les conditions d'environnement avant de choisir de mettre ou non en œuvre les commandes senderNonce et recipientNonce décrites au paragraphe 6.6. Les développeurs de client PKI à états contraints sont fortement encouragés à incorporer l'utilisation de ces commandes.

Une attention extrême doit être portée à l'archivage d'une clé de signature. Le détenteur de la clé archivée peut avoir la capacité d'utiliser la clé pour générer des signatures falsifiées. Il y a cependant des raisons qui font qu'une clé de signature devrait être archivée. Une clé signante de CA archivée peut être récupérée dans le cas d'une défaillance empêchant de continuer de produire des CRL à la suite d'un désastre.

Des précautions doivent précéder l'archivage de clés. Une fois qu'une clé est donnée à une entité d'archivage, celle-ci pourrait utiliser les clés d'une façon impropre. L'attention des utilisateurs devrait être attirée particulièrement sur la nécessité de faire une vérification appropriée du certificat utilisé pour chiffrer le matériel de la clé privée.

Les clients et serveurs doivent faire des vérifications sur les paramètres de chiffrement avant de produire des certificats pour s'assurer que des paramètres faibles ne sont pas utilisés. Une description de l'attaque du petit sous groupe est fournie dans la [RFC2631]. Les méthodes pour éviter l'attaque du petit sous groupe peuvent être trouvées dans la [RFC2785]. Les mises en œuvre de CMC devraient avoir conscience de cette attaque lors des validations de paramètres.

Quand on utilise un secret partagé pour l'authentification, le secret partagé devrait être généré en utilisant de bonnes techniques de nombres aléatoires [RFC4086]. Le choix du secret par l'utilisateur permet de monter des attaques de dictionnaire.

Un soin extrême doit être apporté au traitement de la commande Publier les ancres de confiance. Un traitement incorrect peut conduire à la pratique du "claquement" où un attaquant change l'ensemble d'ancres de confiance pour affaiblir la sécurité.

Une méthode de contrôle de l'utilisation de la commande Publier les ancres de confiance est la suivante. Le client a besoin de s'associer à chaque ancre de confiance acceptée par le client à la source de l'ancre de confiance. De plus, le client devrait associer à chaque ancre de confiance les types de messages pour lesquels l'ancre de confiance est valide (c'est-à-dire, l'ancre de confiance est elle utilisée pour valider des messages S/MIME, TLS, ou des messages d'adhésion à la CMC ?).

Quand un nouveau message est reçu avec une commande Publier les ancres de confiance, le client ne devrait accepter l'ensemble de nouvelles ancres de confiance pour des applications spécifiques que si la signature est validée, si le signataire du message a l'approbation de la politique requise pour mettre à jour les ancres de confiance, et si la politique locale va aussi permettre la mise à jour des ancres de confiance.

La structure de CMS AuthenticatedData assure l'intégrité du message, elle n'assure pas l'authentification du message dans tous les cas. Quand on utilise des MAC dans ce document, les restrictions suivantes doivent être observées. Tous les messages devraient être pour une seule entité. Si deux entités sont placées dans un seul message, les entités peuvent générer de nouveaux messages qui ont un MAC valide et pourraient être supposés provenir de l'envoyeur du message original. Toutes les entités qui ont accès au secret partagé peuvent générer des messages qui vont réussir à la validation de MAC. Cela signifie qu'il faut faire attention à garder cette valeur secrète. Chaque fois que possible, la structure SignedData devrait être utilisée de préférence à la structure AuthenticatedData.

9. Considérations relatives à l'IANA

Le présent document définit un certain nombre d'objets de commandes. Ils sont identifiés par des identifiants d'objet (OID). Les objets sont définis à partir d'un arc délégué par l'IANA au groupe de travail PKIX. Aucune autre action de l'IANA n'est nécessaire pour ce document ou ses mises à jour prévues.

10. Remerciements

Les auteurs et le groupe de travail PKIX remercient de leur participation Xiaoyi Liu et Jeff Weinstein qui ont aidé les auteurs des versions originelles de ce document.

Les auteurs remercient Brian LaMacchia de son travail de développement et de rédaction de beaucoup des concepts présentés dans ce document. Les auteurs remercient aussi Alex Deacon et Barb Fox de leurs contributions.

11. Références

11.1 Références normatives

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

[RFC2314] B. Kaliski, "PKCS n° 10 : Syntaxe de demande de certification, version 1.5", mars 1998. (*Obs., voir* RFC2986) *(Info.)* Noter que cette version de PKCS n° 10 est utilisée pour la compatibilité avec l'utilisation de la syntaxe ASN.1 de 1988. Un travail est en cours dans le groupe de travail PKIX pour mettre à jour avec la syntaxe ASN.1 de 2003.

[RFC2875] H. Prafullchandra, J. Schaad, "Algorithmes de preuve de possession Diffie-Hellman", juillet 2000. *(P.S.) (Remplacée par* RFC655)

[RFC3280] R. Housley, W. Polk, W. Ford et D. Solo, "Profil de certificat d'infrastructure de clé publique X.509 et de liste de révocation de certificat (CRL) pour l'Internet", avril 2002. *(Obsolète, voir* RFC5280)

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

[RFC4211] J. Schaad, "Format de message de demande de certificat d'infrastructure de clé publique X.509 pour l'Internet", septembre 2005. (*P.S. ; remplace* RFC2511 ; *MàJ par* RFC9045*)*

11.2 Références pour information

[PASSWORD] Burr, W., Dodson, D., and Polk, "Electronic Authentication Guideline", NIST SP 800-63, avril 2006.

[RFC2631] E. Rescorla, "Méthode d'accord de clé Diffie-Hellman", juin 1999. *(P.S.)*

[RFC2785] R. Zuccherato, "Méthodes pour éviter les attaques de "petit sous-groupe" sur la méthode d'accord de clés Diffie-Hellman pour S/MIME", mars 2000. *(Information)*

[RFC2797] M. Myers, X. Liu, J. Schaad et J. Weinstein, "Messages de gestion de certificat sur CMS", avril 2000. (*Obsolète, voir* 5272, *P.S*)

[RFC4086] D. Eastlake 3rd, J. Schiller, S. Crocker, "Exigences d'aléa pour la sécurité", juin 2005. (*Remplace* RFC1750) (BCP0106)

[RFC5273] J. Schaad, M. Myers, "Gestion de certificat sur CMS (CMC) : protocoles de transport", juin 2008. *(P.S.)*

[RFC5274] J. Schaad, M. Myers, "Messages de gestion de certificat sur CMS (CMC) : exigences de conformité", juin 2008. *(P.S.)*

Appendice A. Module ASN.1

```
EnrollmentMessageSyntax
{ iso(1) identified-organization(3) dod(6) internet(1) security(5) mechansims(5) pkix(7) id-mod(0) id-mod-cmc2002(23) }

ÉTIQUETTES DE DÉFINITIONS IMPLICITES ::=
DÉBUT

-- EXPORTE TOUT --
-- Les types et valeurs définis dans ce module sont exportés pour être utilisés dans les autres modules ASN.1. D'autres applications peuvent les utiliser pour leurs propres besoins.

IMPORTE

-- PKIX Partie 1 - Implicite de la [RFC3280]
  GeneralName, CRLReason, ReasonFlags
    DE PKIX1Implicit88 {iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0) id-pkix1-implicit(19)}

-- PKIX Partie 1 - Explicite de la [RFC3280]
    AlgorithmIdentifier, Extension, Name, CertificateSerialNumber
   DE PKIX1Explicit88 {iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0) id-pkix1-explicit(18)}

-- Syntaxe de message cryptographique, de la [RFC3852]
```

```
   ContentInfo, Attribute, IssuerAndSerialNumber
   DE CryptographicMessageSyntax2004 { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
modules(0) cms-2004(24)}

-- CRMF, de la [RFC4211]
   CertReqMsg, PKIPublicationInfo, CertTemplate
  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)};

-- Types globaux
   UTF8String ::= [UNIVERSAL 12] IMPLICIT CHAÎNE D'OCTETS
-- Le contenu de ce type se conforme à la RFC 2279.

IDENTIFIANT D'OBJET id-pkix  ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5)
pkix(7) }

IDENTIFIANT D'OBJET id-cmc ::= {id-pkix 7}                  -- commandes de CMC
IDENTIFIANT D'OBJET id-cct ::= {id-pkix 12}                 -- types de contenu de CMC

-- Les commandes suivantes ont le type CHAÎNE D'OCTETS

IDENTIFIANT D'OBJET id-cmc-identityProof ::= {id-cmc 3}
IDENTIFIANT D'OBJET id-cmc-dataReturn ::= {id-cmc 4}
IDENTIFIANT D'OBJET id-cmc-regInfo ::= {id-cmc 18}
IDENTIFIANT D'OBJET id-cmc-responseInfo ::= {id-cmc 19}
IDENTIFIANT D'OBJET id-cmc-queryPending ::= {id-cmc 21}
IDENTIFIANT D'OBJET id-cmc-popLinkRandom ::= {id-cmc 22}
IDENTIFIANT D'OBJET id-cmc-popLinkWitness ::= {id-cmc 23}

-- Les commandes suivantes ont le type UTF8String

IDENTIFIANT D'OBJET  id-cmc-identification ::= {id-cmc 2}

-- Les commandes suivantes ont le type ENTIER

IDENTIFIANT D'OBJET id-cmc-transactionId ::= {id-cmc 5}

-- Les commandes suivantes ont le type CHAÎNE D'OCTETS

IDENTIFIANT D'OBJET id-cmc-senderNonce ::= {id-cmc 6}
IDENTIFIANT D'OBJET id-cmc-recipientNonce ::= {id-cmc 7}

-- Le type de contenu suivant est utilisé pour un message de demande dans le protocole :

IDENTIFIANT D'OBJET id-cct-PKIData ::= { id-cct 2 }

PKIData ::= SEQUENCE {
    controlSequence      SEQUENCE TAILLE(0..MAX) DE TaggedAttribute,
    reqSequence          SEQUENCE TAILLE(0..MAX) DE TaggedRequest,
    cmsSequence          SEQUENCE TAILLE(0..MAX) DE TaggedContentInfo,
    otherMsgSequence     SEQUENCE TAILLE(0..MAX) DE OtherMsg
 }

  bodyIdMax ENTIER ::= 4294967295

  BodyPartID ::= ENTIER(0..bodyIdMax)

 TaggedAttribute ::= SEQUENCE {
    bodyPartID     BodyPartID,
    attrType       IDENTIFIANT D'OBJET,
    attrValues     ENSEMBLE DE AttributeValue
```

```
 }

AttributeValue ::= ANY

TaggedRequest ::= CHOIX {
    tcr             [0] TaggedCertificationRequest,
    crm             [1] CertReqMsg,
    orm             [2] SEQUENCE {
       bodyPartID               BodyPartID,
       requestMessageType       IDENTIFIANT D'OBJET,
       requestMessageValue      ANY DÉFINI PAR requestMessageType
    }
 }

TaggedCertificationRequest ::= SEQUENCE {
    bodyPartID            BodyPartID,
    certificationRequest  CertificationRequest
 }

CertificationRequest ::= SEQUENCE {
   certificationRequestInfo    SEQUENCE {
     version                     ENTIER,
     subject                     Name,
     subjectPublicKeyInfo        SEQUENCE {
       algorithm                   AlgorithmIdentifier,
       subjectPublicKey            CHAÎNE BINAIRE },
     attributes                [0] ENSEMBLE IMPLICITE DE Attribute },
   signatureAlgorithm    AlgorithmIdentifier,
   signature             CHAÎNE BINAIRE
 }

TaggedContentInfo ::= SEQUENCE {
    bodyPartID              BodyPartID,
    contentInfo             ContentInfo
}

OtherMsg ::= SEQUENCE {
    bodyPartID          BodyPartID,
    otherMsgType        IDENTIFIANT D'OBJET,
    otherMsgValue       ANY DÉFINI PAR otherMsgType }

-- Ceci définit le message de réponse dans le protocole
IDENTIFIANT D'OBJET id-cct-PKIResponse ::= { id-cct 3 }

ResponseBody ::= PKIResponse

PKIResponse ::= SEQUENCE {
    controlSequence     SEQUENCE TAILLE(0..MAX) DE TaggedAttribute,
    cmsSequence         SEQUENCE TAILLE(0..MAX) DE TaggedContentInfo,
    otherMsgSequence    SEQUENCE TAILLE(0..MAX) DE OtherMsg
 }

-- Utilisé pour retourner l'état dans une réponse

IDENTIFIANT D'OBJET id-cmc-statusInfo  ::= {id-cmc 1}

CMCStatusInfo ::= SEQUENCE {
    cMCStatus       CMCStatus,
    bodyList        SEQUENCE TAILLE (1..MAX) DE BodyPartID,
    statusString    UTF8String FACULTATIF,
    otherInfo       CHOIX {
```

```
      failInfo         CMCFailInfo,
      pendInfo         PendInfo } FACULTATIF
}

PendInfo ::= SEQUENCE {
    pendToken       CHAÎNE D'OCTETS,
    pendTime        GeneralizedTime
}

CMCStatus ::= ENTIER {
    success             (0),
    failed              (2),
    pending             (3),
    noSupport           (4),
    confirmRequired     (5),
    popRequired         (6),
    partial             (7)
}

-- Note : L'orthographe de unsupportedExt est corrigé dans cette version. Dans la RFC 2797, c'était unsuportedExt.

 CMCFailInfo ::= ENTIER {
    badAlg              (0)
    badMessageCheck     (1)
    badRequest          (2)
    badTime             (3)
    badCertId           (4)
    unsupportedExt      (5)
    mustArchiveKeys     (6)
    badIdentity         (7)
    popRequired         (8)
    popFailed           (9)
    noKeyReuse          (10)
    internalCAError     (11)
    tryLater            (12)
    authDataFail        (13)
}

-- Utilisé pour que les RA ajoutent des extensions aux demandes de certification
IDENTIFIANT D'OBJET id-cmc-addExtensions  ::= {id-cmc 8}

 AddExtensions ::= SEQUENCE {
    pkiDataReference    BodyPartID,
    certReferences      SEQUENCE DE BodyPartID,
    extensions          SEQUENCE DE Extension
}

IDENTIFIANT D'OBJET id-cmc-encryptedPOP  ::= {id-cmc 9}
IDENTIFIANT D'OBJET id-cmc-decryptedPOP  ::= {id-cmc 10}

EncryptedPOP ::= SEQUENCE {
    request             TaggedRequest,
    cms                 ContentInfo,
    thePOPAlgID         AlgorithmIdentifier,
    witnessAlgID        AlgorithmIdentifier,
    witness             CHAÎNE D'OCTETS
}

DecryptedPOP ::= SEQUENCE {
    bodyPartID          BodyPartID,
    thePOPAlgID         AlgorithmIdentifier,
```

```
    thePOP              CHAÎNE D'OCTETS
}

IDENTIFIANT D'OBJET id-cmc-lraPOPWitness  ::= {id-cmc 11}

LraPopWitness ::= SEQUENCE {
     pkiDataBodyid   BodyPartID,
     bodyIds             SEQUENCE OF BodyPartID
}

IDENTIFIANT D'OBJET id-cmc-getCert  ::= {id-cmc 15}

GetCert ::= SEQUENCE {
    issuerName      GeneralName,
    serialNumber    ENTIER }

IDENTIFIANT D'OBJET id-cmc-getCRL  ::= {id-cmc 16}

GetCRL ::= SEQUENCE {
    issuerName    Name,
    cRLName      GeneralName FACULTATIF,
    time              GeneralizedTime FACULTATIF,
    reasons          ReasonFlags FACULTATIF }

IDENTIFIANT D'OBJET id-cmc-revokeRequest ::= {id-cmc 17}

RevokeRequest ::= SEQUENCE {
    issuerName         Name,
    serialNumber       ENTIER,
    reason               CRLReason,
    invalidityDate     GeneralizedTime       FACULTATIF,
    passphrase         CHAÎNE D'OCTETS FACULTATIF,
    comment             UTF8String              FACULTATIF }

IDENTIFIANT D'OBJET id-cmc-confirmCertAcceptance  ::= {id-cmc 24}

 CMCCertId ::= IssuerAndSerialNumber

-- Ce qui suit est utilisé pour demander que les extensions de V3 soient ajoutées à un certificat.

IDENTIFIANT D'OBJET id-ExtensionReq ::= {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) 14}

ExtensionReq ::= SEQUENCE TAILLE (1..MAX) DE Extension

-- Ce qui suit existe pour permettre aux messages de demande de certification Diffie-Hellman d'être bien formés.

IDENTIFIANT D'OBJET id-alg-noSignature  ::= {id-pkix id-alg(6) 2}

NoSignatureValue ::= CHAÎNE D'OCTETS

-- Attribut non authentifié pour porter des données supprimables. Cela pourrait être utilisé dans une mise à jour de "Extension
de CMC : côté serveur -- Génération et courtage de clé" (février 2005) et dans d'autres documents.

IDENTIFIANT D'OBJET id-aa ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2)}
IDENTIFIANT D'OBJET id-aa-cmc-unsignedData ::= {id-aa 34}

 CMCUnsignedData ::= SEQUENCE {
    bodyPartPath   BodyPartPath,
    identifier         IDENTIFIANT D'OBJET,
    content           ANY DÉFINI PAR identifier
 }
```

```
-- Remplace les informations d'état de CMC --

IDENTIFIANT D'OBJET id-cmc-statusInfoV2  ::= {id-cmc 25}

CMCStatusInfoV2 ::= SEQUENCE {
   cMCStatus         CMCStatus,
   bodyList          SEQUENCE TAILLE (1..MAX) DE BodyPartReference,
   statusString      UTF8String FACULTATIF,
   otherInfo         CHOIX {
     failInfo            CMCFailInfo,
     pendInfo            PendInfo,
     extendedFailInfo    SEQUENCE {
       failInfoOID         IDENTIFIANT D'OBJET,
       failInfoValue       AttributeValue
   }
  } FACULTATIF
}

BodyPartReference ::= CHOIX {
   bodyPartID        BodyPartID,
   bodyPartPath      BodyPartPath
}

BodyPartPath ::= SEQUENCE TAILLE (1..MAX) DE BodyPartID

-- Permet la distribution des ancres de confiance --

IDENTIFIANT D'OBJET id-cmc-trustedAnchors ::= {id-cmc 26}

PublishTrustAnchors ::= SEQUENCE {
    seqNumber       ENTIER,
    hashAlgorithm   AlgorithmIdentifier,
    anchorHashes    SEQUENCE DE CHAÎNE D'OCTETS
}

IDENTIFIANT D'OBJET id-cmc-authData ::= {id-cmc 27}

AuthPublish ::= BodyPartID

-- Ces deux éléments utilisent BodyPartList
IDENTIFIANT D'OBJET id-cmc-batchRequests ::= {id-cmc 28}
IDENTIFIANT D'OBJET id-cmc-batchResponses ::= {id-cmc 29}

BodyPartList ::= SEQUENCE TAILLE (1..MAX) DE BodyPartID

IDENTIFIANT D'OBJET id-cmc-publishCert  ::= {id-cmc 30}

CMCPublicationInfo ::= SEQUENCE {
    hashAlg          AlgorithmIdentifier,
    certHashes       SEQUENCE DE CHAÎNE D'OCTETS,
    pubInfo          PKIPublicationInfo
}

IDENTIFIANT D'OBJET id-cmc-modCertTemplate ::= {id-cmc 31}

ModCertTemplate ::= SEQUENCE {
    pkiDataReference     BodyPartPath,
    certReferences       BodyPartList,
    replace              BOOLÉEN PAR DÉFAUT VRAI,
    certTemplate         CertTemplate
```

```
}

-- Des informations suivent aux serveurs qu'une ou plusieurs commandes ont déjà été traitées.

IDENTIFIANT D'OBJET id-cmc-controlProcessed ::= {id-cmc 32}

ControlsProcessed ::= SEQUENCE {
    bodyList            SEQUENCE TAILLE(1..MAX) DE BodyPartReference
 }

-- Commande Preuve d'identité avec agilité d'algorithme

IDENTIFIANT D'OBJET id-cmc-identityProofV2 ::= { id-cmc 34 }

IdentifyProofV2 ::= SEQUENCE {
    proofAlgID      AlgorithmIdentifier,
    macAlgId        AlgorithmIdentifier,
    witness         CHAÎNE D'OCTETS
}

IDENTIFIANT D'OBJET id-cmc-popLinkWitnessV2  ::= { id-cmc 33 }
 PopLinkWitnessV2 ::= SEQUENCE {
    keyGenAlgorithm   AlgorithmIdentifier,
    macAlgorithm       AlgorithmIdentifier,
    witness               CHAÎNE D'OCTETS
}

FIN
```

Appendice B. Flux de messages d'adhésion

Cette section est pour information. Son objet est de présenter, dans une version abrégée, les messages qui pourraient s'écouler entre le client et le serveur pour plusieurs cas courants différents.

B.1 Demande d'un certificat signant

Ce paragraphe décrit les messages qui s'écouleraient dans le cas d'une adhésion qui se produirait pour une clé seulement de signature. Si le certificat a été conçu à la fois pour la signature et le chiffrement, la seule différence serait l'extension de l'usage de clé dans la demande de certification.

Message n° 2 du client au serveur :

```
ContentInfo.contentType = id-signedData
  ContentInfo.content
    SignedData.encapContentInfo
      eContentType = id-ct-PKIData
      eContent
        controlSequence
          {102, id-cmc-identityProof, valeur calculée}
          {103, id-cmc-senderNonce, 10001}
        reqSequence
          certRequest
            certReqId = 201
            certTemplate
              subject = Mon DN proposé
              publicKey = Ma clé publique
              extensions
                {id-ce-subjectPublicKeyIdentifier, 1000}
```

```
            {id-ce-keyUsage, digitalSignature}
    SignedData.SignerInfos
     SignerInfo
      sid.subjectKeyIdentifier = 1000
```

Réponse du serveur au client :

```
  ContentInfo.contentType = id-signedData
  ContentInfo.content
    SignedData.encapContentInfo
     eContentType = id-ct-PKIResponse
     eContent
      controlSequence
        {102, id-cmc-statusInfoV2, {succès, 201}}
        {103, id-cmc-senderNonce, 10005}
        {104, id-cmc-recipientNonce, 10001}
    certificats
     Nouveau certificat produit
     Autres certificats
    SignedData.SignerInfos
     Signés par la CA
```

B.2 Une seule demande de certification, mais modifiée par la RA

Ce paragraphe décrit les messages qui sécouleraient dans le cas d'une adhésion qui a une RA au milieu du flux de données. Cette RA va modifier la demande de certification avant de la passer à la CA.

Message du client à la RA :

```
ContentInfo.contentType = id-signedData
ContentInfo.content
  SignedData.encapContentInfo
   eContentType = id-ct-PKIData
   eContent
    controlSequence
      {102, id-cmc-identityProof, valeur calculée}
      {103, id-cmc-senderNonce, 10001}
    reqSequence
     certRequest
      certReqId = 201
      certTemplate
       subject = Mon DN proposé
       publicKey = Ma clé publique
       extensions
         {id-ce-subjectPublicKeyIdentifier, 1000}
         {id-ce-keyUsage, digitalSignature}
 SignedData.SignerInfos
  SignerInfo
   sid.subjectKeyIdentifier = 1000
```

Message de la RA à la CA :

```
ContentInfo.contentType = id-signedData
ContentInfo.content
 SignedData.encapContentInfo
  eContentType = id-ct-PKIData
  eContent
   controlSequence
     { 102, id-cmc-batchRequests, { 1, 2} }
     { 103, id-cmc-addExtensions,
```

```
        { {1, 201, {id-ce-certificatePolicies, anyPolicy}}
          {1, 201, {id-ce-subjectAltName, {données d'extension}}
          {2, XXX, {id-ce-subjectAltName, {données d'extension}}}
```

La valeur XXX n'est pas connue ici ; elle ferait référence dans la seconde demande du client, qui n'est pas affichée ci-dessus.

```
    cmsSequence
      { 1, <Message du client à RA n° 1> }
      { 2, <Message du client à RA n° 2> }
  SignedData.SignerInfos
    SignerInfo
     sid = clé de la RA.
```

Réponse de la CA à la RA:

```
ContentInfo.contentType = id-signedData
ContentInfo.content
  SignedData.encapContentInfo
    eContentType = id-ct-PKIResponse
    eContent
      controlSequence
        {102, id-cmc-BatchResponse, {999, 998}}
        {103, id-cmc-statusInfoV2, {échec, 2, badIdentity}}
      cmsSequence
       { bodyPartID = 999
         contentInfo
           ContentInfo.contentType = id-signedData
           ContentInfo.content
             SignedData.encapContentInfo
               eContentType = id-ct-PKIResponse
               eContent
                 controlSequence
                   {102, id-cmc-statusInfoV2, {succès, 201}}
             certificats
               Nouveau certificat produit
               Autres certificats
             SignedData.SignerInfos
               Signés par la CA
       }
       { bodyPartID = 998,
         contentInfo
           ContentInfo.contentType = id-signedData
           ContentInfo.content
             SignedData.encapContentInfo
               eContentType = id-ct-PKIResponse
               eContent
                 controlSequence
                   {102, id-cmc-statusInfoV2, {échec, badAlg}}
             certificats
               Nouveau certificat produit
               Autres certificats
             SignedData.SignerInfos
               Signés par la CA
       }
      SignedData.SignerInfos
        Signés par la CA
```

Réponse de la RA au client :

```
ContentInfo.contentType = id-signedData
ContentInfo.content
  SignedData.encapContentInfo
    eContentType = id-ct-PKIResponse
```

```
    eContent
      controlSequence
        {102, id-cmc-statusInfoV2, {succès, 201}}
  certificats
    Nouveau certificat produit
    Autres certificats
  SignedData.SignerInfos
    Signés par la CA
```

B.3 POP direct pour un certificat RSA

Ce paragraphe décrit les messages qui s'écouleraient dans le cas d'une adhésion faite pour un certificat seulement de chiffrement en utilisant une méthode POP directe. Pour simplifier, il est supposé que le demandeur de certification a déjà le certificat seulement de signature.

Le fait qu'un second aller-retour soit requis est implicite plutôt qu'explicite. Le serveur détermine cela sur la base du fait qu'il n'existe pas d'autre POP pour la demande de certification.

Message n° 1 du client au serveur :

```
ContentInfo.contentType = id-signedData
ContentInfo.content
  SignedData.encapContentInfo
    eContentType = id-ct-PKIData
    eContent
      controlSequence
        {102, id-cmc-transactionId, 10132985123483401}
        {103, id-cmc-senderNonce, 10001}
        {104, id-cmc-dataReturn, <paquet de données binaires qui identifient où est la clé en question.>}
      reqSequence
        certRequest
          certReqId = 201
          certTemplate
            subject = <Mon DN provenant de mon certificat signant>
            publicKey = Ma clé publique
            extensions
              {id-ce-keyUsage, keyEncipherment}
          popo
            keyEncipherment
              subsequentMessage
  SignedData.SignerInfos
    SignerInfo
      Signées par le certificat signant du demandeur
```

Réponse n° 1 du serveur au client :

```
ContentInfo.contentType = id-signedData
ContentInfo.content
  SignedData.encapContentInfo
    eContentType = id-ct-PKIResponse
    eContent
      controlSequence
        {101, id-cmc-statusInfoV2, {échec, 201, popRequired}}
        {102, id-cmc-transactionId, 10132985123483401}
        {103, id-cmc-senderNonce, 10005}
        {104, id-cmc-recipientNonce, 10001}
        {105, id-cmc-encryptedPOP, {
          request {
            certRequest
              certReqId = 201
```

```
            certTemplate
              subject = <Mon DN provenant de mon certificat signant>
              publicKey = Ma clé publique
              extensions
                {id-ce-keyUsage, keyEncipherment}
            popo
              keyEncipherment
              subsequentMessage
          }
          cms
           contentType = id-envelopedData
           content
            recipientInfos.riid.issuerSerialNumber = <NULL, 201>
            encryptedContentInfo
              eContentType = id-data
              eContent = <Valeur chiffrée de 'y'>
          thePOPAlgID = HMAC-SHA1
          witnessAlgID = SHA-1
          witness <valeur hachée de 'y'>}}
        {106, id-cmc-dataReturn, <paquet de données binaires identifiant où est la clé en question.>}
  certificats
    Autres certificats (facultatif)
  SignedData.SignerInfos
    Signées pa la CA

ContentInfo.contentType = id-signedData
ContentInfo.content
  SignedData.encapContentInfo
    eContentType = id-ct-PKIData
    eContent
      controlSequence
        {102, id-cmc-transactionId, 10132985123483401}
        {103, id-cmc-senderNonce, 100101}
        {104, id-cmc-dataReturn, <paquet de données binaires identifiant où est la clé en question .>}
        {105, id-cmc-recipientNonce, 10005}
        {107, id-cmc-decryptedPOP, {
          bodyPartID 201,
          thePOPAlgID HMAC-SHA1,
          thePOP <la valeur calculée du HMAC vient ici>}}
      reqSequence
        certRequest
          certReqId = 201
          certTemplate
            subject = <Mon DN provenant de mon certificat signant>
            publicKey = Ma clé publique
            extensions
              {id-ce-keyUsage, keyEncipherment}
          popo
            keyEncipherment
              subsequentMessage

  SignedData.SignerInfos
    SignerInfo
      Signées par le certificat signant du demandeur

Réponse n° 2 du serveur au client :

ContentInfo.contentType = id-signedData
ContentInfo.content
  SignedData.encapContentInfo
    eContentType = id-ct-PKIResponse
```

eContent
controlSequence
{101, id-cmc-transactionId, 10132985123483401}
{102, id-cmc-statusInfoV2, {succès, 201}}
{103, id-cmc-senderNonce, 10019}
{104, id-cmc-recipientNonce, 100101}
{105, id-cmc-dataReturn, <paquet de données binaires identifiant où est la clé en question.>}
certificats
Nouveau certificat produit
Autres certificats
SignedData.SignerInfos
Signées par la CA

Appendice C. Production de demandes de certification de clé publique Diffie-Hellman

La signature sur la demande fait partie d'une demande de certification ; Diffie-Hellman est un algorithme d'accord de clé et ne peut pas être utilisé pour produire directement l'objet signature requis. La [RFC2875] fournit deux façons de produire la valeur de signature nécessaire. Le présent document définit aussi un algorithme de signature qui ne fournit pas de valeur de POP, mais peut être utilisé pour produire la valeur de signature nécessaire.

C.1 Mécanisme de signature No-Signature

Des clés privées de gestion de clés (chiffrement/déchiffrement) ne peuvent pas toujours être utilisées pour produire des types de valeur de signature car elles peuvent être dans un appareil seulement déchiffreur. Les demandes de certification exigent que le champ Signature soit rempli. Ce paragraphe fournit un algorithme de signature spécifique pour cela. L'identifiant d'objet et la valeur de signature suivants sont utilisés pour identifier ce type de signature :

IDENTIFIANT D'OBJET id-alg-noSignature ::= {id-pkix id-alg(6) 2}

NoSignatureValue ::= CHAÎNE D'OCTETS

Les paramètres pour id-alg-noSignature DOIVENT être présents et DOIVENT être codés comme NULL. NoSignatureValue contient le hachage de la demande de certification. Il est important de réaliser qu'il n'y a pas de sécurité associée à ce type de signature. Si ce type de signature est sur une demande de certification et si la politique de l'autorité de certification exige une preuve de possession de la clé privée, le mécanisme POP défini au paragraphe 6.7 DOIT être utilisé.

Adresse des auteurs

Jim Schaad
Soaring Hawk Consulting
PO Box 675
Gold Bar, WA 98251 USA
téléphone : (425) 785-1031
mél : jimsch@nwlink.com

Michael Myers
TraceRoute Security, Inc.
mél : mmyers@fastq.com

Déclaration complète de droits de reproduction

informations ci-encloses ne violent aucun droit ou aucune garantie implicite de commercialisation ou d'aptitude à un objet particulier.

Propriété intellectuelle
L'IETF ne prend pas position sur la validité et la portée de tout droit de propriété intellectuelle ou autres droits qui pourraient être revendiqués au titre de la mise en œuvre ou l'utilisation de la technologie décrite dans le présent document ou sur la mesure dans laquelle toute licence sur de tels droits pourrait être ou n'être pas disponible ; pas plus qu'elle ne prétend avoir accompli aucun effort pour identifier de tels droits. Les informations sur les procédures de l'ISOC au sujet des droits dans les documents de l'ISOC figurent dans les BCP 78 et BCP 79.

Des copies des dépôts d'IPR faites au secrétariat de l'IETF et toutes assurances de disponibilité de licences, ou le résultat de tentatives faites pour obtenir une licence ou permission générale d'utilisation de tels droits de propriété par ceux qui mettent en œuvre ou utilisent la présente spécification peuvent être obtenues sur le répertoire en ligne des IPR de l'IETF à http://www.ietf.org/ipr .

L'IETF invite toute partie intéressée à porter son attention sur tous copyrights, licences ou applications de licence, ou autres droits de propriété qui pourraient couvrir les technologies qui peuvent être nécessaires pour mettre en œuvre la présente norme. Prière d'adresser les informations à l'IETF à ietf- ipr@ietf.org .

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