Groupe de travail Réseau
Request for Comments : 4211
RFC rendue obsolète : 2511
Catégorie : Sur la voie de la normalisation

J. Schaad, Soaring Hawk Consulting
septembre 2005

Traduction Claude Brière de L'Isle

Format de message de demande de certificat (CRMF) pour l'infrastructure de clé publique X.509 de l'Internet

Statut de ce mémoire
Le présent document spécifie un protocole en cours de normalisation de l'Internet pour la communauté de l'Internet, et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "Normes officielles des protocoles de l'Internet" (STD 1) pour connaître l'état de la normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.
(Cette traduction incorpore les errata publiés jusqu'en avril 2020).

Résumé
Le présent document décrit la syntaxe et la sémantique du format de message de demande de certificat (CRMF, *Certificate Request Message Format*). Cette syntaxe est utilisée pour porter une demande d'un certificat à une autorité de certification (CA, *Certification Authority*) éventuellement via une autorité d'enregistrement (RA, *Registration Authority*) pour les besoins de production de certificats X.509. La demande va normalement inclure une clé publique et les informations d'enregistrement associées. Le présent document ne définit pas un protocole de demande de certificat.

Table des matières

1. Introduction et terminologie

Le présent document décrit le format de message de demande de certificat (CRMF, *Certificate Request Message Format*). Un objet Message de demande de certificat est utilisé au sein d'un protocole pour porter une demande de certificat à une autorité de certification (CA, *Certification Authority*) éventuellement via une autorité d'enregistrement (RA, *Registration Authority*) pour les besoins de la production de certificat X.509. La demande va normalement inclure une clé publique et les informations d'enregistrement associées.

L'objet Demande de certificat défini dans le présent document n'est pas un protocole autonome. Les informations définies dans ce document sont destinée à être utilisées par un protocole de demande de certificat (CRP, *Certificate Request Protocol*) défini par ailleurs. Le protocole de référence est supposé définir les algorithmes à utiliser, les informations d'enregistrement et les structures de commandes. Beaucoup des exigences du présent document se réfèrent au protocole de demande de certificat de référence.

Les demandes de certificat peuvent être soumises par une RA demandant un certificat au nom d'un sujet, par une CA demandant un certificat croisé à une autre CA, ou directement par une entité d'extrémité (EE, *End Entity*).

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].

2. Généralités

La construction d'une demande de certification implique les étapes suivantes :

a) Un objet CertRequest *(demande de certification)* est construit. Cet objet peut inclure la clé publique, tout ou partie du nom du sujet, les autres champs du certificat demandé, et des informations de commande supplémentaires relatives au processus d'enregistrement. Selon le CRP, ces informations peuvent être spécifiées par le sujet et potentiellement modifiées par une RA, ou spécifiées par la RA sur la base de sa connaissance du sujet ou de la documentation présentée par le sujet.

b) Si c'est exigé, est calculée une valeur de preuve de possession (de la clé privée correspondant à la clé publique pour laquelle un certificat est demandé).

c) Des informations d'enregistrement supplémentaires peuvent être combinées avec la valeur de preuve de possession et la structure CertRequest pour former un CertReqMessage *(message de demande de certification)*. Des informations d'enregistrement supplémentaires peuvent être ajoutées aussi bien par le sujet que par une RA.

d) Le CertReqMessage est communiqué de façon sûre à une CA. Des moyens spécifiques de transport sûr sont à spécifier par chaque CRP qui se réfère au présent document.

2.1 Changements depuis la RFC 2511

1. Ajout d'une section d'introduction.
2. Ajout du concept de CRP et de langage relatif aux CRP.
3. Au paragraphe 6.2, on a changé regToken *(jeton d'enregistrement)* en authentificateur.
4. Ajout d'informations décrivant le contenu de la structure EncryptedValue *(valeur chiffrée)*.
5. Changement du nom et du contenu d'OID {id-regInfo 1}.
6. Ajout de texte détaillant ce qui entre dans les champs des différentes structures définies dans le document.
7. Remplacement de l'Appendice A par une référence à la [RFC2875]. La seule différence est que l'ancien texte spécifiait l'utilisation de "subject alt name" au lieu de "subject name" si "subject name" était vide. Ceci n'est pas possible pour un certificat de CA produit avec PKIX. Il serait cependant utile de mettre à jour la RFC 2875 pour avoir cette position de repli.
8. Insertion de l'Appendice C décrivant pourquoi la preuve de possession (POP, *Proof-Of-Possession*) est nécessaire et ce que sont certaines attaques de POP.
9. Le champ pop dans la structure CertReqMsg a été renommé en popo pour éviter la confusion entre POP et pop.
10. L'utilisation de la structure EncryptedValue a été déconseillée en faveur de la structure EnvelopedData.
11. Ajout de détails sur la façon dont les clés privées sont à structurer lors du chiffrement.
12. Admission de la POP sur les algorithmes d'accord de clé autres que DH.

3. Syntaxe de CertReqMessage

Un message de demande de certificat se compose d'une demande de certificat, d'un champ facultatif de preuve de possession, et d'un champ facultatif d'informations d'enregistrement.

```
CertReqMessages ::= SEQUENCE TAILLE (1..MAX) DE CertReqMsg

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

Les champs de CertReqMsg ont la signification suivante :

certReq contient le gabarit du certificat demandé. Le gabarit est rempli par (ou au nom du) le sujet. Tous les champs du gabarit n'ont pas besoin d'être spécifiés. On trouvera des détails sur ce champ à la Section 5.

popo contient la valeur utilisée pour démontrer que l'entité qui sera identifiée comme sujet du certificat est en fait en possession de la clé privée correspondante. Ce champ varie dans sa structure et son contenu sur la base de l'algorithme de clé publique et du mode (chiffrement ou signature) selon lequel l'algorithme est utilisé, comme spécifié dans le champ KeyUsage du certificat à produire. On trouvera les détails sur ce champ à la Section 4.

Le champ regInfo DEVRAIT contenir seulement des informations supplémentaires relatives au contexte de la demande de certificat, lorsque ces informations sont nécessaires pour satisfaire la demande. Ces informations peuvent inclure des informations pour contacter l'utilisateur, des informations de facturation, ou autres informations auxiliaires utiles pour l'accomplissement de la demande.

Les informations directement relatives au contenu du certificat DEVRAIENT être incluses dans le contenu de certReq. Cependant, l'inclusion de contenu supplémentaire de certReq par les RA peut invalider le champ popo (selon les détails de la méthode POP utilisée). Donc, les données destinées au contenu du certificat PEUVENT être fournies dans regInfo.

Il est de la responsabilité d'un CRP de définir les détails de ce qui peut être spécifié dans le champ regInfo. Le présent document décrit une méthode de codage des informations qui se trouvent dans ce champ. Les détails de ce codage se trouvent à l'Appendice A.

4. Preuve de possession (POP)

Afin d'empêcher certaines attaques (voir à l'Appendice C) et pour permettre à une CA/RA de vérifier correctement la validité du lien entre un sujet et une paire de clés, la structure de gestion de PKI spécifiée ici rend possible à un sujet de prouver qu'il est en possession de (c'est-à-dire, qu'il est capable d'utiliser) la clé privée correspondante de la clé publique pour laquelle un certificat est demandé. Un CRP est libre de choisir comment appliquer la POP (par exemple, par des procédures hors bande ou par un message de CRMF dans la bande) dans ses échanges de certification. Dans un certain CRP, les CA et RA sont libres de choisir parmi les méthodes de POP fournies (c'est-à-dire, c'est une question de politique locale pour une RA/CA). Un CRP DEVRAIT définir quelles méthodes de POP sont exigées, ou spécifier un mécanisme pour que les clients découvrent les méthodes de POP prises en charge.

Tout CRP qui fait référence au présent document DOIT appliquer la POP par un moyen quelconque. Il y a actuellement de nombreux protocoles opérationnels non PKIX (divers protocoles de messagerie électronique en sont un exemple) qui ne vérifient pas explicitement le lien entre l'entité d'extrémité et la clé privée. Jusqu'à ce qu'il existe des protocoles opérationnels qui vérifient bien le lien (pour la signature, le chiffrement, et les paires de clé d'accord de clé) et qu'ils soient omniprésents, ce lien ne peut pas être supposé avoir été vérifié par la CA/RA. Donc, on ne peut pas vraiment savoir si le lien de la clé publique et de l'identité dans le certificat est en fait correct.

La POP est accomplie de différentes façons selon le type de clé pour lequel un certificat est demandé. Si une clé peut être utilisée pour plusieurs objets (par exemple, une clé de signature et une clé RSA de déchiffrement) alors toutes les méthodes PEUVENT être utilisées. Les concepteurs de protocole doivent savoir qu'il peut y avoir des limitations de matériel sur les méthodes de POP qui sont utilisables, par exemple, si la clé privée est conservée dans un jeton matériel.

La présente spécification permet que la POP soit validée par la CA, par la RA, ou par les deux. Certaines politiques exigent que la CA vérifie la POP durant la production du certificat, et dans ce cas, la RA DOIT transmettre inaltérés à la CA les champs CertRequest et ProofOfPossession de l'entité d'extrémité. (Dans ce cas, la RA pourrait vérifier la POP et rejeter les demandes de certificat défaillantes plutôt que de les transmettre à la CA.) Si la CA n'est pas obligée par la politique de vérifier la POP, la RA DEVRAIT alors transmettre la demande et la preuve de l'entité d'extrémité inaltérées à la CA comme ci-dessus. Si ce n'est pas possible (par exemple parce que la RA vérifie la POP par une méthode hors bande) la RA utilise alors l'élément raVerified pour attester à la CA que la preuve requise a été validée. Si la CA/RA utilise une méthode hors bande pour vérifier la POP (comme une livraison physique des clé privées générées par la CA/RA) le champ ProofOfPossession est alors omis.

```
ProofOfPossession ::= CHOIX {
   raVerified          [0] NULL,
   signature           [1] POPOSigningKey,
   keyEncipherment     [2] POPOPrivKey,
   keyAgreement        [3] POPOPrivKey }
```

Les champs de ProofOfPossession ont la signification suivante :

raVerified indique que la RA a effectué la POP requise sur la demande de certificat. Ce champ est utilisé par une RA quand 1) la CA n'est pas obligée de faire sa propre vérification de POP et 2) la RA a besoin de changer le contenu du champ certReq. Les CRP DOIVENT fournir une méthode pour que la RA signe le champ ProofOfPossession. Un demandeur NE DOIT PAS établir ce champ et une RA/CA NE DOIT PAS accepter un champ ProofOfPossession où le demandeur a établi ce champ.

signature est utilisée pour effectuer la POP avec des clés de signature. Les détails de ce champ sont traités au paragraphe 4.1.

keyEncipherment est utilisé pour effectuer la POP avec des clés fondées sur le chiffrement de clés de chiffrement (c'est-à-dire, RSA). Les détails de ce champ sont traités au paragraphe 4.2.

keyAgreement est utilisé pour effectuer la POP avec des clés de chiffrement de type accord de clé (c'est-à-dire, Diffie-Hellman). Les détails de ce champ sont traités au paragraphe 4.3.

4.1 Preuve de possession de clé de signature

La POP pour une clé de signature est accomplie en effectuant une opération de signature sur un élément de données contenant l'identité pour laquelle le certificat est désiré.

Il y a trois cas qui doivent être examinés lorsque on fait une preuve de possession pour une clé de signature :

1. Le sujet du certificat n'a pas encore établi une identité authentifiée avec une CA/RA, mais a un mot de passe et une chaîne d'identité provenant de la CA/RA. Dans ce cas, la structure POPOSigningKeyInput va être remplie en utilisant le choix publicKeyMAC pour authInfo, et le mot de passe et l'identité vont être utilisés pour calculer la valeur du publicKeyMAC. La clé publique pour le certificat qui est demandé va être placée dans les deux structures POPOSigningKeyInput et gabarit de certificat. Le champ "signature" est calculé sur le codage en DER de la structure POPOSigningKeyInput.

2. La CA/RA a établi une identité authentifiée pour le sujet du certificat, mais le demandeur ne la place pas dans la demande de certificat. Dans ce cas, la structure POPOSigningKeyInput va être remplie en utilisant le choix de l'envoyeur pour authInfo. La clé publique pour le certificat demandé va être placée dans les deux structures POPOSigningKeyInput et gabarit de certificat. Le champ "signature" est calculé sur le codage en DER de la structure POPOSigningKeyInput.

3. Le sujet du certificat place son nom dans la structure de gabarit de certificat avec la clé publique. Dans ce cas, le champ poposkInput est omis de la structure POPOSigningKey. Le champ signature est calculé sur le codage en DER de certReq.

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

Les champs de POPOSigningKey ont la signification suivante :

poposkInput contient les données à signer, lorsque présentes. Ce champ DOIT être présent quand le gabarit de certificat ne contient ni la valeur de clé publique ni une valeur de nom de sujet.

algorithmIdentifier identifie l'algorithme de signature et les paramètres associés utilisés pour produire la valeur de POP.

signature contient la valeur de POP produite. Si poposkInput est présent, la signature est calculée sur la valeur codée en DER de poposkInput. Si poposkInput est absent, la signature est calculée sur la valeur codée en DER de certReq.

```
POPOSigningKeyInput ::= SEQUENCE {
  authInfo          CHOIX {
     sender                 [0] GeneralName,
-- utilisé seulement si une identité authentifiée a été étable pour l'envoyeur (par exemple, un DN provenant d'un certificat produit  précédemment et actuellement valide).
      publicKeyMAC     PKMACValue },
-- utilisé si aucun GeneralName authentifié n'existe actuellement pour l'envoyeur ; publicKeyMAC contient un code d'authentification de message (MAC, Message Authentication Code) fondé sur un mot de passe de la valeur codée en DER de la clé publique.
  publicKey       SubjectPublicKeyInfo }                -- provenant de CertTemplate.
```

Les champs de POPOSigningKeyInput ont la signification suivante :

sender contient une identité authentifiée qui a été précédemment établie pour le sujet.

publicKeyMAC contient une valeur calculée qui utilise un secret partagé entre la CA/RA et le demandeur du certificat.

publicKey contient une copie de la clé publique provenant du gabarit de certificat. Ce DOIT être exactement la même valeur que celle contenue dans le gabarit de certificat.

```
PKMACValue ::= SEQUENCE {
  algId  AlgorithmIdentifier,
  value  CHAÎNE BINAIRE }
```

Les champs de PKMACValue ont la signification suivante :

algId identifie l'algorithme utilisé pour calculer la valeur de MAC. Toutes les mises en œuvre DOIVENT prendre en charge id-PasswordBasedMAC. Les détails de cet algorithme sont présentés au paragraphe 4.4.

value contient la valeur calculée du MAC. La valeur du MAC est calculée sur la clé publique codée en DER du sujet du certificat.

La CA/RA identifie le secret partagé à utiliser en regardant 1) le champ de nom général dans la demande de certificat ou 2) les commandes soit jeton d'enregistrement (paragraphe 6.1) soit jeton authentifié (paragraphe 6.2).

4.2 Clés de chiffrement de clé,

La preuve de possession pour les clés de chiffrement de clé est accomplie par une de trois différentes méthodes. La clé privée peut être fournie à la CA/RA, un défi chiffré provenant de la CA/RA peut être déchiffré (méthode directe) ou le certificat créé peut être retourné chiffré et utilisé comme réponse au défi (méthode indirecte).

```
POPOPrivKey ::= CHOIX {
   thisMessage          [0] CHAÎNE BINAIRE,          -- déconseillé
   subsequentMessage    [1] SubsequentMessage,
   dhMAC                [2] CHAÎNE BINAIRE,          -- déconseillé
   agreeMAC             [3] PKMACValue,
   encryptedKey         [4] EnvelopedData }
-- pour keyAgreement (seulement), la possession est prouvée dans ce message (qui contient un MAC (sur la valeur codée en DER du paramètre certReq dans CertReqMsg, qui doit inclure le sujet et la clé publique) fondé sur une clé déduite de la clé DH privée de l'entité d'extrémité et la clé publique DH de la CA) ; la valeur dhMAC DOIT être calculée selon les directives données dans la RFC 2875 pour la preuve de possession DH statique.
```

```
SubsequentMessage ::= ENTIER {
   encrCert       (0),
   challengeResp (1) }
```

Les champs de POPOPrivKey ont la signification suivante :

thisMessage contient la clé privée chiffrée pour laquelle un certificat doit être produit. La possession de la clé privée est prouvée par sa fourniture à la CA/RA. Ce champ était rédigé de façon incorrecte lors de la première production de cette spécification. La façon correcte d'utiliser ce champ est de créer une structure EncryptedValue où le contenu chiffré est la clé privée, la structure EncryptedValue est alors enveloppée dans le type CHAÎNE BINAIRE. Ce champ a été déconseillé en faveur de encryptedKey.

subsequentMessage est utilisé pour indiquer que la POP sera effectuée en déchiffrant un message provenant de la CA/RA et en retournant une réponse. Le type de message à déchiffrer est indiqué par la valeur utilisée.

encrCert indique que le certificat produit est à retourner sous une forme chiffrée. Le demandeur est obligé de déchiffrer le certificat et de prouver sa réussite à la CA/RA. Les détails de cette opération sont fournis par le CRP.

challengeResponse indique qu'un message de défi est à envoyer de la CA/RA au demandeur. Les détails du message de défi et sa réponse sont à fournir par le CRP.

dhMAC est utilisé pour les clés d'accord de clé Diffie-Hellman. Il contient un MAC calculé qui est obtenu en utilisant la clé privée du demandeur et la clé publique de la CA/RA. L'utilisation de ce champ est déconseillée en faveur du champ agreeMAC. Les détails sont traités au paragraphe 4.3.

agreeMAC est utilisé pour les clés d'accord de clé. Il contient un MAC calculé qui est obtenu en utilisant la clé privée du demandeur et une clé publique correspondante de la CA/RA. Les détails sont traités au paragraphe 4.3.

macAlg contient l'algorithme qui identifie la méthode utilisée pour calculer la valeur du MAC.

macValue contient la valeur du MAC calculée.

encryptedKey contient la clé privée chiffrée qui correspond à la clé publique pour laquelle le certificat est à produire. Il contient aussi une valeur d'identification pour indiquer qu'il a été construit par le demandeur du certificat. Le type de contenu enveloppé DOIT être id-ct-encKeyWithID.

On s'attend à ce que les protocoles qui incorporent la présente spécification incluront les messages de confirmation et de défi-réponse nécessaires pour un protocole complet.

4.2.1 Type de contenu d'informations de clé privée

Ce type de contenu est utilisé pour 1) prouver la possession des clés privées et 2) la récupération de clés privées (en utilisant les commandes d'options d'archive du paragraphe 6.4). Cette structure se fonde sur la structure d'informations de clé privée de [PKCS8] mais avec une différence délibérée. Il y a une attaque potentielle sur les agents de tiers de confiance si ils déchiffrent la clé privée mais ne savent pas à qui la clé chiffrée est supposée appartenir. Un attaquant pourrait intercepter la clé privée chiffrée, construire une demande de certificat autour d'elle et demander ensuite une opération de récupération de la clé privée.

Ce type de contenu et sa structure sont :

```
IDENTIFIANT D'OBJET  id-ct-encKeyWithID ::= {id-ct 21}

EncKeyWithID ::= SEQUENCE {
  privateKey          PrivateKeyInfo,
  identifier CHOIX {
    string             UTF8String,
    generalName        GeneralName
  } FACULTATIF
}

PrivateKeyInfo ::= SEQUENCE {
   version               ENTIER,
```

```
  privateKeyAlgorithm      AlgorithmIdentifier,
  privateKey               CHAÎNE D'OCTETS,
  attributes               [0] IMPLICITE Attributes FACULTATIF
}

Attributes ::= ENSEMBLE DE Attribute
```

Les champs de EncKeyWithID sont définis comme :

privateKey contient la clé privée codée. Les définitions de trois formats de clé privée sont inclus dans ce document. Les spécifications des algorithmes asymétriques doivent inclure les définitions de clé publique et de clé privée pour être cohérentes.

identifier contient un nom que la CA/RA peut associer au demandeur. Cela sera généralement une portion d'un nom de sujet ou d'un nom de remplacement de sujet (soit d'un certificat existant, soit de la demande de certificat) ou un jeton de texte passé et connu du demandeur et de la CA/RA. Ce champ DOIT être présent si l'objet est de prouver la possession de la clé privée. Le champ DEVRAIT être présent si c'est l'archivage d'une clé et si le tiers de confiance est supposé déchiffrer la clé.

Les champs de PrivatekeyInfo sont définis comme :

version DOIT être la valeur 0.

privateKeyAlgorithm contient l'identifiant pour l'objet clé privée.

privateKey est une chaîne d'octets dont le contenu est la clé privée et dont le format est défini par la valeur de privateKeyAlgorithm.

attributes est un ensemble d'attributs. Ce sont des extensions d'informations qui font partie des informations de clé privée.

4.2.2 Structures de clé privée

On définit ici les structures à utiliser pour trois algorithmes.

4.2.2.1 Clés privées D-H

Lors de la création de PrivateKeyInfo pour une clé D-H, les règles suivantes s'appliquent :

1. privateKeyAlgorithm DOIT être réglé à id-dh-private-number. Le paramètre pour id-dh-private-number est DomainParameters (importé de la [RFC3279]).
2. La structure ASN pour privateKey DOIT être DH-PrivateKey ::= ENTIER
3. Le champ attributes DOIT être omis.

4.2.2.2 Clés privées DSA

Lors de la création de PrivateKeyInfo pour une clé DSA, les règles suivantes s'appliquent :

1. privateKeyAlgorithm DOIT être réglé à id-dsa. Le paramètre pour id-dsa est Dss-Parms (importé de la [RFC3279]).
2. La structure ASN pour privateKey DOIT être DSA-PrivateKey ::= ENTIER
3. Le champ attributes DOIT être omis.

4.2.2.3 Clés privées RSA

Lors de la création de PrivateKeyInfo pour une clé RSA, les règles suivantes s'appliquent :

1. privateKeyAlgorithm DOIT être réglé à rsaEncryption.
2. La structure ASN pour privateKey DOIT être RSAPrivateKey (défini dans la [RFC3447])
3. Le champ attributes DOIT être omis.

4.2.3 Lignes directrices du défi-réponse

Ce sont des lignes directrices pour les auteurs de protocole d'adhésion sur la façon dont une preuve de possession indirecte est supposée fonctionner et sur certains des domaines où on doit être prudent lors du dessin des messages pour mettre en œuvre cette méthode de POP.

1. La demande d'engagement originale comporte une preuve d'identité d'un certain type et la portion publique de la clé de chiffrement. Noter que la preuve d'identité doit couvrir la portion publique de la clé de chiffrement pour empêcher les attaques de substitution (où l'attaquant met sa clé publique à la place de la votre).

2. Le message de réponse provenant du serveur comporte une valeur de données chiffrées d'un certain type. Cette valeur doit être authentifiée d'une certaine façon comme venant du serveur. La spécification doit inclure les spécificités de comment cette valeur est retournée pour les différents types de clés. Pour les clés RSA, la valeur peut être spécifiée comme étant chiffrée directement par la clé publique RSA ; cela ne va pas fonctionner pour une clé D-H où on a besoin de spécifier un mécanisme indirect pour chiffrer la valeur.

3. Le second message de demande inclut un hachage de la valeur déchiffrée. Ce message NE DOIT PAS être juste le hachage de la valeur chiffrée, car on ne devrait jamais "signer" une valeur complètement aléatoire. Il est souhaitable d'inclure des informations telles que la chaîne d'identité dans le processus de hachage afin que cela puisse être fait explicitement. Cette valeur retournée DOIT être incluse dans une seconde preuve d'identité.

Il est fortement suggéré que les identifiants de transaction et les valeurs de nom occasionnel soient exigées quant on effectue une POP indirecte, car cela permet 1) de lier ensemble les différents messages dans le processus, et 2) de laisser chaque entité injecter une certaine quantité de données aléatoires dans le processus de preuve d'identité.

4.3 Clés d'accord de clé

La preuve de possession pour les clés d'accord de clé est accomplie par une des quatre méthodes suivantes. Les trois premières sont identiques à celles présentées ci-dessus pour les clés de chiffrement de clé. La quatrième méthode tire parti du fait qu'un secret partagé est produit et que la valeur peut être utilisée pour des information de MAC.

Lorsque les méthodes de chiffrement directes ou indirectes présentées ci-dessus sont utilisées, la CA/RA va devoir créer une clé éphémère pour les cas où les paramètres de l'algorithme de chiffrement ne correspondent pas entre la CA/RA et le demandeur.

L'entité d'extrémité peut aussi faire un MAC pour la demande de certificat (en utilisant une clé secrète partagée déduite du calcul) comme quatrième solution pour démontrer la possession. Cette option ne peut être utilisée que si la CA/RA a déjà un certificat connu de l'entité d'extrémité et si le sujet est capable d'utiliser les paramètres de la CA/RA.

Pour l'algorithme d'accord de clé DH, toutes les mises en œuvre DOIVENT prendre en charge la preuve de possession statique DH. Les détails de cet algorithme se trouvent à la Section 3 de la [RFC2875]. Note : si le nom de sujet ou de producteur est vide dans le certificat de CA, le nom de remplacement devrait être utilisé à sa place.

4.4 Utilisation de MAC fondé sur un mot de passe

Cet algorithme de MAC a été conçu pour produire un secret partagé (un mot de passe) et l'utiliser à calculer une valeur de vérification sur un élément d'information. L'hypothèse est que, sans le mot de passe, la valeur de vérification correcte ne peut pas être calculée. L'algorithme calcule plusieurs fois la fonction unidirectionnelle (OWF, *one-way function*) afin de ralentir toute attaque de dictionnaire contre la valeur du mot de passe.

L'identifiant d'algorithme et la structure de paramètres utilisée pour le MAC fondé sur le mot de passe est :

IDENTIFIANT D'OBJET id-PasswordBasedMAC ::= { 1 2 840 113533 7 66 13}

```
PBMParameter ::= SEQUENCE {
   salt                CHAÎNE D'OCTETS,
   owf                 AlgorithmIdentifier,
   iterationCount      ENTIER,
   mac                 AlgorithmIdentifier
 }
```

Les champs de PBMParameter ont la signification suivante :

salt contient une valeur générée au hasard utilisée pour calculer la clé du processus de construction du MAC. Le sel DEVRAIT faire au moins 8 octets (64 bits).

owf identifie l'algorithme et les paramètres associés utilisés pour calculer la clé utilisée dans le processus de MAC. Toutes les mises en œuvre DOIVENT prendre en charge SHA-1.

iterationCount identifie le nombre de fois que le hachage est appliqué durant le processus de calcul de la clé. iterationCount DOIT être au minimum de 100. Nombreux sont ceux qui suggèrent d'utiliser des valeurs aussi élevées que 1000 itérations comme valeur minimum. Le compromis est ici entre protection du mot de passe contre les attaques et temps passé par le serveur à traiter toutes les différentes itérations de déduction des mots de passe. Le hachage est généralement considéré comme une opération peu coûteuse mais ce peut n'être pas vrai à l'avenir avec toutes les fonctions de hachage.

mac identifie l'algorithme et les paramètres associés de la fonction de MAC à utiliser. Toutes les mises en œuvre DOIVENT prendre en charge HMAC-SHA1 [RFC2104]. Toutes les mises en œuvre DEVRAIENT prendre en charge DES-MAC et Triple-DES-MAC [PKCS11].

Le pseudo-code pour l'algorithme est :

Entrées :
pw - chaîne d'octets contenant le mot de passe de l'utilisateur.
data - chaîne d'octets contenant la valeur à couvrir par le MAC.
Iter - compte des itérations.

Résultat :
MAC - chaîne d'octets contenant la valeur de MAC résultante.

1. Générer une valeur de sel aléatoire de S
2. Ajouter le sel à pw. K = pw || sel.
3. Hacher la valeur de K. K = HASH(K)
4. Si Iter est supérieur à zéro. Iter = Iter - 1. revenir à l'étape 3.
5. Calculer un HMAC comme documenté dans la [RFC2104].

MAC = HASH(K OUX opad, HASH(K OUX ipad, données))

Où opad et ipad sont définis dans la [RFC2104].

5. Syntaxe de CertRequest

La syntaxe de CertRequest consiste en un identifiant de demande, un gabarit de contenu de certificat, et en une séquence facultative d'informations de contrôle.

```
CertRequest ::= SEQUENCE {
  certReqId      ENTIER,                    -- identifiant pour confronter demande et réponse.
  certTemplate   CertTemplate,              -- champs choisis du certificat à produire.
  controls       Controls FACULTATIF }      -- attributs affectant 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 }

OptionalValidity ::= SEQUENCE {
  notBefore  [0] Time FACULTATIF,
  notAfter   [1] Time FACULTATIF }          -- au moins un doit être présent.

Time ::= CHOIX {
```

```
utcTime       UTCTime,
generalTime   GeneralizedTime }
```

Les champs de CertRequest ont la signification suivante :

certReqId contient une valeur entière qui est utilisée par le demandeur du certificat pour associer une demande de certificat spécifique à une réponse de certificat.

certTemplate contient un gabarit d'un certificat X.509. Le demandeur remplit les champs pour lesquels il désire des valeurs spécifiques. Les détails de ces champs sont donnés plus loin.

controls contient les attributs qui ne font pas partie du certificat, mais contrôlent le contexte dans lequel le certificat est à produire. Les détails sur les commandes définies dans ce document se trouvent à la Section 6. D'autres documents peuvent définir d'autres commandes. Les CRP sont chargés de spécifier les commandes requises.

Les champs de CertTemplate ont la signification suivante :

version DOIT être 2 si il est fourni. Il DEVRAIT être omis.

serialNumber DOIT être omis. Ce champ est alloué par la CA durant la création de certificat.

signingAlg DOIT être omis. Ce champ est alloué par la CA durant la création de certificat.

issuer est normalement omis. Il serait rempli par la CA dont le demandeur souhaite qu'elle produise le certificat dans les situations où une RA dessert plus d'une CA.

validity est normalement omis. Il peut être utilisé pour demander que les certificats commencent à un certain instant à venir, ou expire à un instant spécifique. Un cas où ce champ serait couramment utilisé est quand un certificat croisé est produit pour une CA. Dans ce cas, la validité d'un certificat existant va être placée dans ce champ afin que le nouveau certificat ait la même période de validité que le certificat existant. Si validity n'est pas omis, au moins un de ses sous champs DOIT être spécifié. Les sous champs sont :

notBefore contient l'heure de début demandée pour le certificat. L'heure suit les mêmes règles que l'heure notBefore dans la [RFC3280].

notAfter contient l'heure d'expiration demandée pour le certificat. L'heure suit les mêmes règles que l'heure notAfter dans la [RFC3280].

subject est rempli avec le nom suggéré pour le demandeur. Cela va normalement être rempli avec un nom qui a été produit précédemment au demandeur par la CA.

publicKey contient la clé publique pour laquelle le certificat est créé. Ce champ DOIT être rempli si le demandeur génère sa propre clé. Le champ est omis si la clé est générée par la RA/CA.

issuerUID DOIT être omis. Ce champ a été déconseillé dans la [RFC3280].

subjectUID DOIT être omis. Ce champ a été déconseillé dans la [RFC3280].

extensions contient des extensions que le demandeur veut voir placées dans le certificat. Ces extensions vont généralement traiter de choses comme le réglage de l'usage de la clé à keyEncipherment.

À l'exception du champ publicKey, il est permis à la CA/RA d'altérer tout champ demandé. Le certificat retourné doit être vérifié par le demandeur pour voir si les champs ont été réglés d'une façon acceptable. La CA/RA DEVRAIT utiliser les champs du gabarit si possible.

Il y a des cas où tous les champs du gabarit vont être omis. Si la génération de clé est faite à la CA/RA et si la preuve d'identité est placée dans un site différent (comme le jeton id-regCtrl-regToken ci-dessous) il n'y a alors pas de champ qui doive être spécifié par le demandeur de certificat.

6. Syntaxe des commandes

Le générateur d'une CertRequest peut inclure une ou plusieurs valeurs de commandes relevant du traitement de la demande.

Controls ::= SEQUENCE TAILLE(1..MAX) DE AttributeTypeAndValue

Les commandes suivantes sont définies dans le présent document : regToken (paragraphe 6.1) ; authentifiant (paragraphe 6.2) ; pkiPublicationInfo (paragraphe 6.3) ; pkiArchiveOptions (paragraphe 6.4) ; oldCertID (paragraphe 6.5) ; protocolEncrKey (paragraphe 6.6). Chaque CRP DOIT définir l'ensemble de commandes pris en charge. Des commandes supplémentaires pourront être définies par de futures RFC ou par le CRP lui-même.

6.1 Commandes de jeton d'enregistrement

Une commande regToken contient des informations à usage unique (fondées sur une valeur secrète ou autres informations partagées) destinées à être utilisées par la CA pour vérifier l'identité du sujet avant de produire un certificat. À réception d'une demande de certification contenant une valeur pour regToken, la CA receveuse vérifie les informations afin de confirmer l'identité revendiquée dans la demande de certification.

La valeur pour regToken peut être générée par la CA et fournie hors bande à l'abonné, ou peut autrement être disponible à la fois à la CA et à l'abonné. La sécurité de tout échange hors bande devrait être proportionnée au risque que la CA va tolérer à l'égard de l'acception d'une valeur interceptée provenant de quelqu'un d'autre que l'abonné prévu. La valeur de regToken n'est pas chiffrée au retour ; si les données sont considérées comme sensibles, elle devront être dissimulées par le demandeur.

La commande regToken n'est utilisée que pour l'initialisation d'une entité d'extrémité au sein de la PKI, tandis que la commande authentifiant (paragraphe 6.2) peut être utilisée pour la demande de certification initiale aussi bien que les suivantes.

Dans certaines instances d'utilisation, la valeur pour regToken pourrait être une chaîne de texte ou une quantité numérique comme un nombre aléatoire. Dans le premier cas, la valeur est codée comme la représentation de chaîne de texte de la quantité binaire. Le codage de regToken DEVRA être UTF8String.

IDENTIFIANT D'OBJET id-regCtrl-regToken ::= { id-regCtrl 1 }

Sans accord préalable entre l'abonné et les agents de la CA, cette valeur va être un secret partagé textuel d'un certain type. Si une valeur calculée fondée sur ce secret partagé est plutôt utilisée, il est suggéré que le CRP définisse une nouvelle commande d'enregistrement pour ce calcul spécifique.

6.2 Commande authentifiant

Une commande authentifiant contient des informations utilisées au fil de l'eau pour établir une vérification non cryptographique de l'identité en communication avec la CA. Des exemples sont : le nom de jeune fille de la mère, les quatre derniers chiffres du numéro de sécurité sociale, ou autres informations, fondées sur la connaissance, partagées avec la CA de l'abonné, un hachage de telles informations, ou d'autres information produites à cette fin. La valeur d'une commande authentifiant peut être générée par l'abonné ou par la CA.

Dans certaines instances d'utilisation, la valeur de l'authentifiant pourrait être une chaîne de texte ou une quantité numérique comme un nombre aléatoire. Dans le premier cas, la valeur est codée comme la représentation de chaîne de texte de la quantité binaire. Le codage de authentifiant DEVRA être UTF8String.

IDENTIFIANT D'OBJET id-regCtrl-authenticator ::= { id-regCtrl 2 }

Pour décider si on utilise un authentifiant ou un regToken, voici des lignes directrices : si la valeur est une valeur d'usage unique, regToken devrait être utilisé ; si la valeur a un usage à long terme, la commande authentifiant sera utilisée.

6.3 Commande Informations de publication

La commande pkiPublicationInfo permet aux abonnés d'influencer la publication par la CA/RA du certificat. Cette commande est considérée comme pour information et peut être ignorée par les CA/RA. Elle est définie par l'OID et la syntaxe suivants :

IDENTIFIANT D'OBJET id-regCtrl-pkiPublicationInfo ::= { id-regCtrl 3 }

PKIPublicationInfo ::= SEQUENCE {

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

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

Les champs de PKIPublicationInfo ont la signification suivante :

action indique si le demandeur souhaite ou non que la CA/RA publie le certificat. Les valeurs et leur signification sont :
dontPublish indique que le demandeur souhaite que la CA/RA ne publie pas le certificat (cela peut indiquer que le demandeur a l'intention de publier lui-même le certificat). Si dontPublish est utilisé, le champ pubInfos DOIT être omis.
pleasePublish indique que le demandeur souhaite que la CA/RA publie le certificat.

pubInfos contient les sites où le demandeur désire que la CA/RA publie le certificat. Ce champ est omis si le choix dontPublish est retenu. Si le demandeur veut spécifier des sites pour la publication du certificat, et pour permettre à la CA/RA de publier sur d'autres sites, il va spécifier plusieurs valeurs de la structure SinglePubInfo, dont une serait dontCare.

Les champs de SinglePubInfo ont la signification suivante :

pubMethod indique le type d'adresse pour le site sur lequel le demandeur désire que le certificat soit placé par la CA/RA.

dontCare indique que la CA/RA peut publier le certificat dans tout site qu'elle choisit. Si dontCare est utilisé, le champ pubLocation DOIT être omis.

x500 indique que le demandeur souhaite que la CA/RA publie le certificat sur un site spécifique. Le site est indiqué dans le champ x500 de pubLocation.

ldap indique que le demandeur souhaite que la CA/RA publie le certificat sur un site spécifique. Le site est indiqué dans le champ ldap de pubLocation.

web indique que le demandeur souhaite que la CA/RA publie le certificat sur un site spécifique. Le site est indiqué dans le champ http de pubLocation.

pubLocation contient l'adresse à laquelle le certificat sera placé. Le choix dans le champ nom général est dicté par le choix de pubMethod dans cette structure.

Les sites de publication peuvent être fournis dans n'importe quel ordre. Tous les sites sont à traiter par la CA pour les besoins de la publication.

6.4 Commande Options d'archive

La commande pkiArchiveOptions permet aux abonnés de fournir les informations nécessaires pour établir une archive de la clé privée correspondant à la clé publique de la demande de certification. Elle est définie par l'OID et la syntaxe suivants :

IDENTIFIANT D'OBJET id-regCtrl-pkiArchiveOptions ::= { id-regCtrl 4 }

```
PKIArchiveOptions ::= CHOIX {
  encryptedPrivKey       [0] EncryptedKey,         -- valeur réelle de la clé privée.
  keyGenParameters       [1] KeyGenParameters,  -- paramètres qui permettent que la clé privée soit re-générée.
archiveRemGenPrivKey [2] BOOLÉEN }
-- réglé à VRAI si l'envoyeur souhaite que le receveur archive la clé privée d'une paire de clés que le receveur génère en
   réponse à cette demande ; réglé à FAUX si aucun archivage n'est désiré.
```

```
EncryptedKey ::= CHOIX {
  encryptedValue       EncryptedValue,                  -- déconseillé
  envelopedData     [0] EnvelopedData }
-- La clé privée chiffrée DOIT être placée dans la CHAÎNE D'OCTETS envelopedData encryptedContentInfo
   encryptedContent.

EncryptedValue ::= SEQUENCE {
  intendedAlg   [0] AlgorithmIdentifier  FACULTATIF,   -- algorithme prévu pour lequel la valeur va être utilisée.
  symmAlg       [1] AlgorithmIdentifier  FACULTATIF,   -- algorithme symétrique utilisé pour chiffrer la valeur.
  encSymmKey [2] CHAÎNE BINAIRE FACULTATIF,   -- clé symétrique (chiffrée) utilisée pour chiffrer la valeur.
  keyAlg        [3] AlgorithmIdentifier  FACULTATIF,   -- algorithme utilisé pour chiffrer la clé symétrique
  valueHint     [4] CHAÎNE D'OCTETS   FACULTATIF,
-- brève description ou identifiant du contenu de encValue (peut n'avoir de signification que pour l'entité envoyeuse, et
   utilisé seulement si EncryptedValue peut être réexaminé plus tard par l'entité envoyeuse).
  encValue      CHAÎNE BINAIRE }
-- L'utilisation du champ EncryptedValue a été déconseillée en faveur de la structure EnvelopedData.
-- Quand EncryptedValue est utilisé pour porter une clé privée (par opposition à un certificat) les mises en œuvre
   DOIVENT prendre en charge le champ encValue contenant des PrivateKeyInfo chiffrées comme défini dans
   [PKCS11], section 12.11. Si encValue contient d'autres formats/codages pour la clé privée, le premier octet de
   valueHint PEUT être utilisé pour indiquer le format/codage (mais noter que les valeurs possibles de cet octet ne sont
   pas spécifiées pour l'instant). Dans tous les cas, le champ intendedAlg DOIT être utilisé pour indiquer au moins l'OID
   de l'algorithme prévu pour la clé privée, sauf si cette information est connue a priori de l'envoyeur et du receveur par
   d'autres moyens.

KeyGenParameters ::= CHAÎNE D'OCTETS
```

Les champs de PKIArchiveOptions ont la signification suivante :

encryptedPrivKey contient une version chiffrée de la clé privée.

keyGenParameters contient les informations nécessaires au demandeur pour re-générer la clé privée. Par exemple, pour de nombreuses mises en œuvre RSA on pourrait envoyer le premier nombre aléatoire essayé comme nombre premier. La structure pour y arriver n'est pas définie dans le présent document. Les CRP qui définissent un contenu pour cette structure DOIVENT définir non seulement le contenu qui va là mais aussi comment ces données sont dissimulées à un accès non autorisé.

archiveRemGenPrivKey indique que le demandeur désire que la clé générée par la CA/RA au nom du demandeur soit archivée.

Les champs de EncryptedKey ont la signification suivante :

encryptedValue n'est plus utilisé. Ce champ a été déconseillé ainsi que la structure EncryptedValue.

envelopedData contient la valeur chiffrée de la clé privée. Les CRP qui utilisent cette structure DOIVENT définir la ou les entités pour qui les données sont à chiffrer (l'EE, les tiers de confiance, les CA) et comment la clé ou l'ensemble de clés va être déterminé. Les détails de la construction d'une structure EnvelopedData se trouvent dans la [RFC3852]. Le contenu chiffré DOIT être un id-ct-encKeyWithID. L'identifiant peut être omis sauf si cette structure est aussi utilisée pour faire la preuve de possession.

6.5 Commande Identifiant de vieux certificat

Si présente, la commande OldCertID spécifie le certificat à mettre à jour par la demande de certification en cours. L'OID et la syntaxe sont :

```
IDENTIFIANT D'OBJET  id-regCtrl-oldCertID ::= { id-regCtrl 5 }

CertId ::= SEQUENCE {
    issuer           GeneralName,
    serialNumber     ENTIER
  }
```

6.6 Commande Clé de chiffrement de protocole

Si présente, la commande protocolEncrKey spécifie une clé que la CA va utiliser pour le chiffrement en réponse aux messages de demande de certification. L'OID pour cette commande est id-regCtrl-protocolEncrKey. La structure de paramètre pour ce champ est SubjectPublicKeyInfo. (Cette structure est définie dans la [RFC3280].)

IDENTIFIANT D'OBJET id-regCtrl-protocolEncrKey ::= { id-regCtrl 6 }

Cette commande est utilisée quand une CA a des informations à envoyer à l'abonné et qu'elles doivent être chiffrées. De telles informations incluent une clé privée générée par la CA pour l'usage de l'abonné.

7. Commande RegInfo

Cette section traite des commandes qui sont à placer dans le champ regInfo de la structure CertReqMsg.

7.1 utf8Pairs

Cette commande est utilisée pour porter des informations fondées sur le texte provenant du sujet et allant à une RA et à une CA qui produit un certificat. L'OID pour cette structure est id-regInfo-utf8Pairs et a un type de UTF8String.

IDENTIFIANT D'OBJET id-regInfo-utf8Pairs ::= { id-regInfo 1 }

Le nom se termine par le caractère point d'interrogation ("?"). La valeur se termine par le caractère pour cent "%". Les paires nom-valeur peuvent être répétées. Donc la syntaxe est : Nom?Valeur%[Nom?Valeur%]*

Le mécanisme %xx du paragraphe 2.1 du STD 66 [RFC3986] est utilisé pour coder '?' (%3f) et '%' (%25) si ils ne sont pas utilisés pour leur usage normal. Les noms NE DOIVENT PAS commencer par un caractère numérique.

Cette commande peut apparaître plusieurs fois dans la structure regInfo. La résolution des conflits d'informations est une affaire de politique locale de la RA/CA.

L'Appendice A contient un ensemble de noms communs et de formats de données correspondants aux champs qui apparaissent couramment dans les certificats et les répertoires.

7.2 certReq

Cette commande est destinée à traiter le problème d'une RA qui a besoin de modifier le gabarit de certificat proposé par un sujet, mais le sujet a utilisé le gabarit de certificat au titre de son calcul de POP. Dans ce cas, la RA peut placer un nouveau gabarit de certificat dans la séquence regInfo.

Cette commande a l'OID id-regInfo-certReq et la structure CertRequest. Il ne peut y avoir qu'une seule instance de cet attribut dans la séquence regInfo. Si cette commande existe dans la structure regInfo, le gabarit de certificat dans la demande est alors ignoré. La RA DOIT copier toutes les données provenant du gabarit cœur dans cet attribut.

IDENTIFIANT D'OBJET id-regInfo-certReq ::= { id-regInfo 2 }

8. Identifiants d'objet

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

-- arc pour les protocoles de PKI Internet X.509 et leurs composants :
IDENTIFIANT D'OBJET id-pkip :: { id-pkix pkip(5) }

-- arc pour les commandes d'enregistrement dans CRMF :
IDENTIFIANT D'OBJET id-regCtrl ::= { id-pkip regCtrl(1) }

```
-- arc pour les informations d'enregistrement dans CRMF :
IDENTIFIANT D'OBJET id-regInfo ::= { id-pkip id-regInfo(2) }
```

9. Considérations sur la sécurité

Les protocoles d'engagement, par leur nature même, impliquent de grandes quantités d'informations privées. Cela peut inclure des clés privées, des numéros d'identité, des numéros de carte de crédit, et autres de même sorte. La sécurité de tout CRP se fonde sur les mécanismes de sécurité du protocole et/ou processus utilisé pour les communications entre CA, RA et EE. Tous les protocoles doivent assurer le masquage, soit via le chiffrement, soit par un traitement hors ligne, de toutes les informations sensibles de l'utilisateur.

De nombreux protocoles d'engagement assurent l'établissement initial d'identité entre la CA/RA et l'EE par l'utilisation d'un jeton. Généralement ce jeton est livré en utilisant une méthode de livraison hors bande (comme le système de messagerie gouvernemental). La sécurité de tout échange hors bande doit être proportionnée aux risques que la CA/RA va tolérer à l'égard de l'interception du jeton par un tiers.

Les mises en œuvre doivent appliquer les valeurs de preuve de possession durant le traitement d'un engagement de certificat. Un bon algorithme de POP doit fournir la preuve de deux choses : 1) que la clé est liée à un utilisateur spécifique et 2) que l'utilisateur a l'usage de la clé en question. Manquer à mettre en œuvre la POP permettrait à des gens de créer des certificats où la clé publique et les valeurs de noms ne sont pas correctement liées. Cela permet une usurpation d'identité sur les clés de signature et l'interception des messages chiffrés.

Les mises en œuvre doivent utiliser des générateur de nombres aléatoire à forte entropie pour produire les clés privées. Les mises en œuvre doivent générer au hasard les clés de chiffrement de contenu, les clés d'authentification de message, les valeurs d'initialisation (IV), le sel, et le bourrage. L'utilisation d'un générateur de nombres pseudo aléatoires (PRNG, *pseudo-random number generator)* non adéquat pour générer des clés de chiffrement peut résulter en peu ou pas du tout de sécurité. Un attaquant peut trouver beaucoup plus facile de reproduire l'environnement du PRNG qui a produit les clés, cherchant dans le petit ensemble de possibilités résultantes, plutôt qu'une recherche en force brute sur tout l'espace de clés. La génération de nombres aléatoires de qualité est difficile. La [RFC4086] offre des lignes directrices importantes dans ce domaine et l'Appendice 3 de FIPS Pub 186 [DSS] fournit une technique de PRNG de qualité.

Les mises en œuvre doivent protéger les clés privées. La compromission de la clé privée du signataire permet à des tiers de se faire passer pour le signataire. La compromission d'une clé privée de déchiffrement permet l'interception des messages par un tiers.

Une caractéristique de la syntaxe du message de demande de certificat est que la génération de la clé soit effectuée à distance de la création de la demande de certificat. Cette caractéristique ne devrait jamais être utilisée pour la génération des clés de signature. Si des clés de signature sont générées pour l'utilisateur, un élément de répudiation entre alors en jeu. L'utilisateur peut prétendre qu'un élément a été signé par l'entité qui a généré la clé tout comme une entité qui peut avoir vu la valeur de la clé durant le transfert du générateur à l'EE. Il faut veiller à ce que le générateur de la clé distante protège les clés de chiffrement afin de se protéger contre l'interception des clés par un tiers. Cela signifie que les algorithmes de chiffrement utilisés doivent être sûrs, et une clé de chiffrement de contenu ou une clé de chiffrement de clé doit être utilisée pour masquer la clé privée durant le transport de retour à l'utilisateur. Les CRP ne doivent jamais supposer qu'une clé de signature générée par l'usager peut être utilisée pour déchiffrer le paquetage dans lequel est transportée une clé privée de chiffrement.

Le présent document décrit une méthode par laquelle on peut avoir un tiers de confiance. Plusieurs problèmes doivent être pris en compte lors de l'utilisation d'un tiers de confiance. D'abord, le client doit être capable d'identifier correctement l'entité à laquelle la clé va être confiée, ou le CRP doit donner une méthode pour que le client puisse découvrir cette information. Un CRP ne peut pas supposer que l'agent de tiers de confiance et la CA sont la même entité et ont donc le même nom. Ensuite, les algorithmes utilisés pour masquer la clé privée ou d'autres informations de génération de clé durant le transport à l'agent tiers de confiance doivent être proportionnés à la valeur des données protégées par la clé. Ensuite l'agent tiers de confiance doit fournir des garanties suffisante qu'une clé placée en dépôt ne sera retournée qu'à l'entité qui devrait être capable d'obtenir la clé privée. Généralement, cela devrait être restreint à l'entité qui a archivé les données. Enfin, les données archivées doivent être mémorisées de manière sûre. Une méthode courante pour ce faire est de rechiffrer les données en des clés auxquelles seul l'agent tiers de confiance a accès. Dans ces cas, on peut aussi avoir besoin d'un tiers de confiance pour archiver la clé du tiers de confiance. L'accès à l'agent tiers de confiance ou à la clé archivée reviendrait à avoir l'accès à toutes les clés privées qui ont été archivées auprès de cet agent.

10. Références

10.1 Références normatives

[PKCS11] RSA Laboratories, The Public-Key Cryptography Standards -"PKCS #11 v2.11: Cryptographic Token Interface Standard", RSA Security Inc., juin 2001.

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

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

[RFC2875] H. Prafullchandra, J. Schaad, "Algorithmes de preuve de possession Diffie-Hellman", juillet 2000. *(P.S.) (Remplacée par* RFC6955)

[RFC3279] L. Bassham, W. Polk et R. Housley, "Algorithmes et identifiants pour le profil de certificat d'infrastructure de clé publique X.509 et de liste de révocation de certificat (CRL) pour l'Internet", avril 2002.

[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)

[RFC3447] J. Jonsson et B. Kaliski, "Normes de cryptographie à clés publiques (PKCS) n° 1 : Spécifications de la cryptographie RSA version 2.1", février 2003. *(Obsolète, remplacée par* RFC8017) *(Information)*

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

[RFC3986] T. Berners-Lee, R. Fielding et L. Masinter, "Identifiant de ressource uniforme (URI) : Syntaxe générique", STD 66, janvier 2005.

10.2 Références pour information

[DSS] National Institute of Standards and Technology, FIPS Pub 186 : "Digital Signature Standard", mai 1994.

[PKCS8] RSA Laboratories, "PKCS #8: Private-Key Information Syntax Standard", PKCS #8 v1.2, novembre 1993.

[RFC1738] T. Berners-Lee et autres, "Localisateurs uniformes de ressource (URL)", décembre 1994. *(P.S., Obsolète, voir les* RFC4248 et 4266 *; MàJ par* RFC8089)

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

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

11. Remerciements

Le groupe de travail tient à remercier Michael Myers, Carlisle Adams, Dave Solo, et David Kemp, qui sont les auteurs de la version d'origine du présent document.

Le groupe de travail est aussi reconnaissant de leurs contributions à Barbara Fox, Warwick Ford, Russ Housley, et John Pawling, dont la relecture et les commentaires ont significativement précisé et amélioré l'utilité de la présente spécification. Les membres de la liste de diffusion "ca-talk" ont aussi fourni des apports significatifs à l'égard des essais d'interopérabilité.

Le texte de l'Appendice C (Pourquoi faire la preuve de possession) a été tiré d'un message de Al Arsenault et faisait à l'origine partie du document de feuille de route pour PKIX.

Appendice A. Utilisation de RegInfo pour les paires Nom-Valeur

Le champ "value" de la chaîne id-regInfo-utf8Pairs (avec le champ "tag" égal à 12 et le champ "Longueur" approprié) va contenir une série de paires nom-valeur UTF.

Le présent Appendice donne quelques exemples courants de telles paires pour les besoins de la promotion de l'interopérabilité entre des mises en œuvre indépendantes de la présente spécification. Cette liste n'est pas exhaustive et va grossir avec le temps et l'expérience de mise en œuvre

A.1 Noms définis

Le tableau suivant définit un ensemble recommandé d'éléments désignés. La valeur dans la colonne "Valeur de nom" est la chaîne de texte exacte qui va apparaître dans regInfo.

Valeur de nom	Commentaire
version	-- version de cette variante d'utilisation de regInfo.
corp_company	-- entreprise employeuse de l'abonné.
org_unit	-- département.
mail_firstName	-- composant de nom personnel.
mail_middleName	-- composant de nom personnel.
mail_lastName	-- composant de nom personnel.
mail_email	-- adresse de messagerie électronique de l'abonné.
jobTitle	-- titre du poste de l'abonné.
employeeID	-- numéro ou chaîne d'identification de l'abonné.
mailStop	-- nom de boîte aux lettres.
issuerName	-- nom de la CA.
subjectName	-- nom du sujet.
validity	-- intervalle de validité.

Par exemple :
version?1%corp_company?Example, Inc.%org_unit?Engineering%mail_firstName?John%mail_lastName?Smith
%jobTitle?Team Leader%mail_email?john@exemple.com%

A.2 Codage de valeur de IssuerName, SubjectName, et Validity

Quand ils apparaissent dans la syntaxe id-regInfo-utf8Pairs comme des éléments nommés, les codages des valeurs pour issuerName, subjectName, et validity DEVRONT utiliser la syntaxe qui suit. Les caractères [] indiquent un champ facultatif, ::= et | ont leur signification BNF habituelle, et tous les autres symboles (excepté les espaces, qui n'ont pas de signification) en dehors des noms non terminaux sont terminaux. Les noms alphabétiques sont sensibles à la casse.

issuerName ::= <names>
subjectName ::= <names>

<names> ::= <name> | <names>:<name>

<validity> ::= validity ? [<notbefore>]-[<notafter>]

<notbefore> ::= <time>
<notafter> ::= <time>

Où <time> est l'heure UTC sous la forme AAAAMMJJ[HH[MM[SS]]]. HH, MM, et SS sont 00 par défaut et sont omis si ils sont en fin et de valeur 00.

Exemple de codage de validity : validity?-19991231%

est un intervalle de validité sans valeur pour notBefore, et une valeur de 31 décembre 1999 pour notAfter.

Chaque nom comporte un identifiant de forme nom d'un seul caractère, suivi par une valeur de nom d'un ou plusieurs caractères UTF-8. Au sein d'une valeur de nom, quand il est nécessaire de préciser un caractère qui a une signification de format à un niveau externe, la séquence d'échappement %xx DEVRA être utilisée, où xx représente la valeur hexadécimale pour le codage concerné. Le symbole "pour cent" est représenté par %%.

<name> ::= X<xname>|O<oname>|E<ename>|D<dname>|U<uname>|I<iname>

Les formes de nom et les formats de valeur sont les suivants :

Forme de nom de répertoire X.500 (identifiant "X") :

<xname> ::= <rdns>
<rdns> ::= <rdn> | <rdns> , <rdn>
<rdn> ::= <avas>
<avas> ::= <ava> | <avas> + <ava>
<ava> ::= <attyp> = <avalue>
<attyp> ::= OID.<oid> | <stdat>

Le type d'attribut standard <stdat> est un identifiant de type d'attribut alphabétique provenant de l'ensemble suivant :
C (pays)
L (localité)
ST (état ou province)
O (organisation)
OU (département)
CN (nom commun)
STREET (adresse dans la rue)
E (adresse de messagerie électronique).

<avalue> est un composant de nom sous forme d'une chaîne de caractères UTF-8 de 1 à 64 caractères, avec la restriction que dans le sous ensemble IA5 de UTF-8 seuls les caractères ASN.1 PrintableString peuvent être utilisés.

Autre forme de nom (identifiant "O") : <oname> ::= <oid> , <utf8string>

Forme de nom Adresse de messagerie électronique (rfc822name) (identifiant "E") : <ename> ::= <ia5string>

Forme de nom DNS (identifiant "D") : <dname> ::= <ia5string>

Forme de nom URI (identifiant "U") : <uname> ::= <ia5string>

Adresse IP (identifiant "I") : <iname> ::= <oid>

Par exemple :
issuerName?XOU=Notre CA,O=Exemple,C=US% subjectName?XCN=John Smith, O=Example, C=US, E=john@exemple.com%

Appendice B. Structures ASN.1 et OID

PKIXCRMF-2005 {iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0) id-mod-crmf2005(36)}

ÉTIQUETTES IMPLICITES DE DÉFINITIONS ::=
DÉBUT

IMPORTE
-- Cadre d'authentification de l'Annuaire (X.509)
Version, AlgorithmIdentifier, Name, Time,
SubjectPublicKeyInfo, Extensions, UniqueIdentifier, Attribute
DE PKIX1Explicit88 {iso(1) identified-organization(3) dod(6)
internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
id-pkix1-explicit(18)} -- se trouve dans la [RFC3280]

-- Extensions de certificat (X.509)
GeneralName
DE PKIX1Implicit88 {iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7)
id-mod(0) id-pkix1-implicit(19)} -- se trouve dans la [RFC3280]

```
-- Syntaxe de message cryptographique :
    EnvelopedData
      DE CryptographicMessageSyntax2004 { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
         smime(16) modules(0) cms-2004(24) };  -- se trouve dans la [RFC3852]

-- La définition peut être sans commentaire pour les compilateurs ASN.1 qui ne comprennent pas UTF8String.

-- UTF8String ::= [UNIVERSAL 12] CHAÎNE D'OCTETS IMPLICITE
-- Le contenu de ce type correspond à la RFC 2279.

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

-- arc pour protocoles PKI Internet X.509 et leurs composants :

IDENTIFIANT D'OBJET id-pkip  ::= { id-pkix 5 }

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

IDENTIFIANT D'OBJET id-ct ::= { id-smime  1 }  -- types de contenu.

-- Cœur des définitions pour ce module :

CertReqMessages ::= SEQUENCE TAILLE (1..MAX) DE CertReqMsg

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,                             -- ID pour confronter demande et réponse.
 certTemplate  CertTemplate,                       -- Champs choisis du certificat à produire.
 controls      Controls FACULTATIF }               -- Attributs affectant 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 }

OptionalValidity ::= SEQUENCE {
 notBefore  [0] Time FACULTATIF,
 notAfter   [1] Time FACULTATIF }                  -- au moins un DOIT être présent

Controls  ::= SEQUENCE TAILLE(1..MAX) DE AttributeTypeAndValue
AttributeTypeAndValue ::= SEQUENCE {
 type       IDENTIFIANT D'OBJET,
 value      TOUT DÉFINI PAR type }

ProofOfPossession ::= CHOIX {
 raVerified          [0] NUL,          -- utilisé si la RA a déjà vérifié que le demandeur est en possession de la clé privée.
 signature           [1] POPOSigningKey,
 keyEnciphermen      [2] POPOPrivKey,
 keyAgreement        [3] POPOPrivKey }

POPOSigningKey ::= SEQUENCE {
```

```
 poposkInput           [0] POPOSigningKeyInput FACULTATIF,
 algorithmIdentifier   AlgorithmIdentifier,
 signature             CHAÎNE BINAIRE }
```

-- La signature (en utilisant "algorithmIdentifier") est sur la valeur codée en DER de poposkInput. Note : si le CertReqMsg certReq CertTemplate contient les valeurs de sujet et de clé publiqvalue, poposkInput DOIT alors être omis et la signature DOIT être calculée sur la valeur codée en DER de CertReqMsg certReq. Si le CertReqMsg certReq CertTemplate ne contient pas les valeurs à la fois de clé publique et de sujet (c'est-à-dire, si il contient seulement une d'elles, ou aucune des deux) poposkInput DOIT alors être présent et DOIT être signé.

```
POPOSigningKeyInput ::= SEQUENCE {
  authInfo          CHOIX {
    sender            [0] GeneralName,
```

-- utilisé seulement si une identité authentifiée a été établie pour l'envoyeur (par exemple, un DN provenant d'un certificat fourni antérieurement et encore valide).

```
    publicKeyMAC      PKMACValue },
```

-- utilisé si aucun GeneralName authentifié n'existe actuellement pour l'envoyeur ; publicKeyMAC contient un MAC fondé sur le mot de passe sur la valeur codée en DER de publicKey.

```
 publicKey          SubjectPublicKeyInfo }          -- provenant du gabarit de certificat.

PKMACValue ::= SEQUENCE {
  algId         AlgorithmIdentifier,
```

-- la valeur de l'algorithme devra être PasswordBasedMac {1 2 840 113533 7 66 13}; la valeur du paramètre est PBMParameter.

```
  value         CHAÎNE BINAIRE }

PBMParameter ::= SEQUENCE {
   salt                 CHAÎNE D'OCTETS,
   owf                  AlgorithmIdentifier,        -- AlgId pour une fonction unidirectionnelle (SHA-1 recommandé).
   iterationCount       ENTIER,                     -- nombre de fois que l'OWF est appliquée.
   mac                  AlgorithmIdentifier
```

-- AlgId du MAC (par exemple, DES-MAC, Triple-DES-MAC [PKCS11], ou HMAC [RFC2104], [RFC2202])

```
}

POPOPrivKey ::= CHOIX {
 thisMessage           [0] CHAÎNE BINAIRE,         -- Déconseillé.
```

-- La possession est prouvée dans ce message (qui contient la clé privée elle-même (chiffrée pour la CA)).

```
 subsequentMessage [1] SubsequentMessage,          -- La possession sera prouvée dans un message suivant.
 dhMAC                 [2] CHAÎNE BINAIRE,         -- Déconseillé.
 agreeMAC              [3] PKMACValue,
 encryptedKey          [4] EnvelopedData }
```

-- pour l'accord de clé (seulement), la possession est prouvée dans ce message (qui contient un MAC (sur la valeur codée en DER du paramètre certReq dans CertReqMsg, qui DOIT inclure le sujet et la clé publique) sur la base d'une clé déduite de la clé privée DH de l'entité d'extrémité et de la clé publique DH de la CA) ;

```
SubsequentMessage ::= ENTIER {
 encrCert (0),
```

-- demande que le certificat résultant soit chiffré pour l'entité d'extrémité (suivant laquelle la POP sera établie dans un message de confirmation).

```
 challengeResp (1) }
```

-- demande que la CA engage un échange défi-réponse avec l'entité d'extrémité afin de prouver la possession de la clé privée.

-- Allocations d'identifiants d'objet --
-- Commandes d'enregistrement dans CRMF :

```
IDENTIFIANT D'OBJET id-regCtrl ::= { id-pkip 1 }

IDENTIFIANT D'OBJET id-regCtrl-regToken ::= { id-regCtrl 1 }
```

-- avec la syntaxe : RegToken ::= UTF8String

```
IDENTIFIANT D'OBJET id-regCtrl-authenticator ::= { id-regCtrl 2 }
```

```
-- avec la syntaxe : Authenticator ::= UTF8String

IDENTIFIANT D'OBJET id-regCtrl-pkiPublicationInfo ::= { id-regCtrl 3 }
-- avec la syntaxe :

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 }

IDENTIFIANT D'OBJET id-regCtrl-pkiArchiveOptions  ::= { id-regCtrl 4 }
-- avec la syntaxe : PKIArchiveOptions ::= CHOIX {
 encryptedPrivKey          [0] EncryptedKey,        - valeur réelle de la clé privée.
 keyGenParameters          [1] KeyGenParameters,    - paramètres qui permettent de générer à nouveau la clé privée.
 archiveRemGenPrivKey [2] BOOLÉEN }
-- VRAI si l'envoyeur souhaite que le receveur archive la clé privée d'une paire de clé que le receveur génère en réponse à
    cette demande ; FAUX si aucun archivage n'est désiré.

EncryptedKey ::= CHOIX {
 encryptedValue    EncryptedValue,                  -- Déconseillé.
 envelopedData     [0] EnvelopedData }
-- La clé privée chiffrée DOIT être placée dans la CHAÎNE D'OCTETS envelopedData encryptedContentInfo
    encryptedContent.

EncryptedValue ::= SEQUENCE {
 intendedAlg   [0] AlgorithmIdentifier  FACULTATIF,     -- algorithme prévu pour lequel la valeur sera utilisée.
 symmAlg       [1] AlgorithmIdentifier  FACULTATIF,     -- algorithme symétrique utilisé pour chiffrer la valeur.
 encSymmKey [2] CHAÎNE BINAIRE  FACULTATIF,     -- clé symétrique (chiffrée) utilisée pour chiffrer la valeur.
 keyAlg        [3] AlgorithmIdentifier  FACULTATIF,     -- algorithme utilisé pour chiffrer la clé symétrique.
 valueHint     [4] CHAÎNE D'OCTETS  FACULTATIF,
 -- brève description ou identifiant du contenu de encValue (peut n'avoir de signification que pour l'entité envoyeuse, et
    utilisé seulement si EncryptedValue peut être réexaminé à l'avenir par l'entité envoyeuse).
 encValue      CHAÎNE BINAIRE }                         -- valeur chiffrée elle-même.
-- Quand EncryptedValue est utilisé pour porter une clé privée (par opposition à un certificat) les mises en œuvre
    DOIVENT prendre en charge le champ encValue contenant un PrivateKeyInfo chiffré comme défini dans [PKCS11],
    section 12.11. Si encValue contient un autre format/codage pour la clé privée, le premier octet de valueHint PEUT
    être utilisé pour indiquer le format/codage (mais noter que les valeurs possibles de cet octet ne sont pas spécifiées
    pour l'instant). Dans tous les cas, le champ intendedAlg DOIT être utilisé pour indiquer au moins l'OID de
    l'algorithme prévu pour la clé privée, sauf si cette information est connue a priori de l'envoyeur et du receveur par
    d'autres moyens.

KeyGenParameters ::= CHAÎNE D'OCTETS

IDENTIFIANT D'OBJET id-regCtrl-oldCertID ::= { id-regCtrl 5 }
-- avec la syntaxe : OldCertId ::= CertId
CertId ::= SEQUENCE {
 issuer          GeneralName,
 serialNumber    ENTIER }

IDENTIFIANT D'OBJET id-regCtrl-protocolEncrKey ::= { id-regCtrl 6 }
-- avec la syntaxe : ProtocolEncrKey ::= SubjectPublicKeyInfo
```

```
-- Informations d'enregistrement dans CRMF :
IDENTIFIANT D'OBJET id-regInfo ::= { id-pkip 2 }

IDENTIFIANT D'OBJET id-regInfo-utf8Pairs  ::= { id-regInfo 1 }
-- avec la syntaxe : UTF8Pairs ::= UTF8String

IDENTIFIANT D'OBJET id-regInfo-certReq  ::= { id-regInfo 2 }
-- avec la syntaxe : CertReq ::= CertRequest

-- id-ct-encKeyWithID est un nouveau type de contenu pour les objets de CMS. Il contient une clé privée et un identifiant
     pour que l'agent de tiers de confiance vérifie les demandeurs de récupération.

IDENTIFIANT D'OBJET id-ct-encKeyWithID ::= {id-ct 21}

EncKeyWithID ::= SEQUENCE {
  privateKey           PrivateKeyInfo,
  identifier   CHOIX {
    string               UTF8String,
    generalName     GeneralName
  } FACULTATIF
}

PrivateKeyInfo ::= SEQUENCE {
   version                 ENTIER,
   privateKeyAlgorithm    AlgorithmIdentifier,
   privateKey                 CHAÎNE D'OCTETS,
   attributes                 [0] IMPLICITE Attributes FACULTATIF
}

Attributes ::= ENSEMBLE DE Attribute

FIN
```

Appendice C. Pourquoi faire la preuve de possession (POP)

La preuve de possession (POP) signifie que la CA est convaincue de façon adéquate que l'entité qui demande un certificat pour la clé publique Y a accès à la clé privée X correspondante.

La POP est importante parce que elle fournit un niveau d'assurance approprié du fonctionnement correct de la PKI dans son ensemble. À son plus bas niveau, la POP contre le "déni de service auto infligé" ; c'est-à-dire, une entité obtient volontairement un certificat qui ne peut pas être utilisé pour signer ou chifrer/déchiffrer des informations. Cependant, comme le montent les deux exemples qui suivent, la POP contre aussi des menaces moins directes, mais plus sévères.

POP pour clés de signature : il est important de fournir la POP pour les clés utilisées à signer du matériel, afin de fournir la non répudiation des transactions. Par exemple, supposons qu'Alice ait légitimement la clé privée X et sa clé publique Y correspondante. Alice a un certificat de Charlie, une CA, contenant Y. Alice utilise X pour signer une transaction T. Sans POP, Mal pourrait aussi obtenir un certificat de Charlie contenant la même clé publique, Y. Maintenant, il y a deux menaces possibles : Mal pourrait prétendre avoir été le signatairre réel de T; ou Alice peut faussement nier avoir signé T, prétendant que c'était Mal. Comme persnonne ne peut fiablement prouver que Mal possédait X ou ne l'a jamais possédée, aucune de ces prétentions ne peut être réfutée, et donc le service fourni et la confiance dans la PKI ont été défaites. (Bien sûr, si Mal possède réellement X, la clé privée d'Alice, aucun mécanisme de POP au monde ne pourra l'aider, mais c'est un problème différent.)

Noter qu'on peut gagner un niveau de protection en faisant qu'Alice (comme véritable signataire de la transaction) inclue dans les informations signées, son certificat ou un identifiant de son certificat (par exemple, un hachage de son certificat). Cela peut rendre plus difficile à Mal d'en revendiquer la paternité ; il faudrait qu'il affirme avoir incorrectement inclus le certificat d'Alice à la place du sien. Cependant, cela n'empêcherait pas Alice de répudier faussement ses actions. Comme le certificat lui-même est un élément public, Mal pourrait bien sûr avoir inséré le certificat ou identifiant d'Alice dans la transaction signée, et donc sa présence n'indique pas qu'Alice est celle qui a participé à la transaction maintenant répudiée. La seule façon fiable d'arrêter cette attaque est d'exiger que Mal prouve qu'il possède X avant que son certificat soit produit.

Pour les clés de signature utilisées seulement pour l'authentification, et non pour la non répudiation, la menace est plus faible parce que les utilisateurs peuvent ne pas se soucier qu'Alice répudie après coup, et donc la POP devient moins importante. Cependant, la POP DEVRAIT quand même être faite chaque fois que faisable dans cet environnement, par des moyens en ligne ou hors ligne.

POP pour clés de gestion de clé : de façon similaire, la POP pour les clés de gestion de clé (c'est-à-dire, les clés utilisées pour l'accord de clé ou l'échange de clés) peut aider à empêcher de miner la confiance dans la PKI. Supposons que Al soit un nouvel instructeur dans le département informatique de l'université locale. Al a créé un projet d'examen final pour le cours "Introduction au réseautage" qu'il enseigne. Il veut envoyer une copie du projet final à Dorothy, la chef du département, pour qu'elle le relise avant de le donner à l'examen. Cet envoi sera bien sûr chiffré, car plusieurs étudients ont accès au système informatique. Cependant, une rapide recherche dans le répertoire des certificats (par exemple, en cherchant dans le répertoire tous les enregistrements avec la valeur "subjectPublicKey=Dorothy") met en évidence que plusieurs étudiants ont des certificats contenant la même clé publique de gestion de clé que Dorothy. À ce point, si aucune POP n'a été faite par la CA, Al n'a aucun moyen de savoir si tous les étudiants ont simplement créé ces certificats sans connaître la clé privée correspondante (et donc s'il est sûr d'envoyer l'examen chiffré à Dorothy) ou si les étudiants ont d'une certaine façon acquis la clé privée de Dorothy (et donc qu'il n'est certainement pas sûr de l'envoyer). Donc, le service à fournir par la PKI permettant aux utilisateurs de communiquer les uns avec les autres, en sachant de confiance avec qui ils communiquent, a été totalement déjoué. Si la CA fournit la POP, alors soit aucun étudiant n'a de tels certificats, soit Al peut savoir avec certitude que les étudiants connaissent bien la clé privée de Dorothy, et agir en conséquence.

Adresse de l'auteur

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

Déclaration complète de droits de reproduction

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