Groupe de travail Réseau
Request for Comments : 5055
Catégorie : Sur la voie de la normalisation
Traduction Claude Brière de L'Isle

T. Freeman, Microsoft Corp
R. Housley, Vigil Security
A. Malpani, Malpani Consulting Services
D. Cooper, NIST
W. Polk, NIST
décembre 2007

Protocole de validation de certificat fondé sur le serveur (SCVP)

Résumé
Le protocole de validation de certificat fondé sur le serveur (SCVP, *Server-Based Certificate Validation Protocol*) permet à un client de déléguer la construction et la validation du chemin de certification à un serveur. La construction ou la validation du chemin (par exemple, en s'assurant qu'aucun des certificats dans le chemin n'est révoqué) est effectuée en accord avec une politique de validation, qui contient une ou plusieurs ancres de confiance. Il permet une simplification des mises en œuvre de client et l'utilisation d'un ensemble de politiques de validation prédéfinies.

Table des matières

1. Introduction

La validation de certificat est complexe. Si le traitement de certificat va être largement déployé dans diverses applications et environnements, la quantité de traitement dont a besoin une application avant de pouvoir accepter un certificat doit être réduite. Diverses applications peuvent faire usage de certificats de clé publique, mais ces applications sont encombrées par les frais généraux de construction et de validation des chemins de certification. SCVP réduit ces frais généraux pour deux classes d'applications qui utilisent des certificats.

La première classe d'applications veut juste deux choses : la confirmation que la clé publique appartient à l'identité désignée dans le certificat et la confirmation que la clé publique peut être utilisée aux fins prévues. Ces clients peuvent déléguer complètement la construction du chemin de certification et sa validation au serveur SCVP. Ceci est souvent appelé la validation de chemin déléguée (DPV, *Delegated Path Validation*).

La seconde classe d'applications peut effectuer la validation de chemin de certification, mais manque d'une méthode fiable

ou efficace pour construire un chemin de certification valide. Ces clients délèguent la construction du chemin de certification au serveur SCVP, mais pas la validation du chemin de certification retourné. Ceci est souvent appelé la découverte de chemin déléguée (DPD, *Delegated Path Discovery*).

1.1 Terminologie

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].

1.2 Vue d'ensemble de SCVP

Les buts principaux de SCVP sont de rendre plus facile le déploiement d'applications à capacité d'infrastructure de clé publique (PKI, *clé publique Infrastructure*) en déléguant la découverte de chemin et/ou le traitement de validation à un serveur, et de permettre une administration centralisée des politiques de validation au sein d'une organisation. SCVP peut être utilisé par des clients qui font eux-mêmes une grande partie du traitement de certification mais veulent simplemnt qu'un serveur qui n'est pas de confiance collecte les informations pour eux. Cependant, quand le client a une complète confiance dans le serveur SCVP, SCVP peut être utilisé pour déléguer le travail de construction du chemin de certification et de validation, et SCVP peut être utilisé pour s'assurer que les politiques sont appliquées de façon cohérente dans toute une organisation.

Les serveurs SCVP qui ne sont pas de confiance peuvent frounir aux clients les chemins de certification. Ils peuvent aussi fournir aux clients les informations de révocation, comme les listes de révocation de certificats (CRL, *Certificate Revocation List*) et les réponses du protocole d'état de certificat en ligne (OCSP, *Online Certificate Status Protocol*) dont les clients ont besoin pour valider les chemins de certification construits par le serveur SCVP. Ces services peuvent être précieux pour les clients qui ne mettent pas en œuvre les protocoles nécessaires pour trouver et télécharger les certificats intermédiaires, les CRL, et les réponses OCSP.

Les serveurs SCVP de confiance peuvent effectuer la construction du chemin de certification et la validation pour le client. Pour un client qui utilise ces services, le client éprouve à l'égard du serveur SCVP autant de confiance qu'à son propre logiciel de validation de chemin de certification (si il contient un tel logiciel). Il y a deux raisons principales pour qu'un client puisse vouloir faire confiance à un tel serveur SCVP :

1. Le client ne veut pas supporter les frais généraux de l'inclusion d'un logiciel de validation de chemin de certification et de le faire fonctionner pour chaque certificat qu'il reçoit.
2. Le client est dans une organisation ou communauté qui veut centraliser la gestion des politiques de validation. Ces politiques pourraient imposer que soient utilisés des ancres de confiance particulières et les types de vérification de politique qui sont à effectuer durant la validation de chemin de certification.

1.3 Exigences pour SCVP

SCVP satisfait les exigences documentées dans la [RFC3379] pour DPV et DPD.

Noter que la RFC 3379 déclare les exigences suivantes : la réponse DPD DOIT indiquer un des états suivants :

1) un ou plusieurs chemins de certification ont été trouvés en accord avec la politique de découverte de chemin, avec toutes les informations de révocation demandées présentes ;
2) un ou plusieurs chemins de certification ont été trouvés en accord avec la politique de découverte de chemin, avec un sous ensemble des informations de révocation demandées présent ;
3) un ou plusieurs chemins de certification ont été trouvés en accord avec la politique de découverte de chemin, avec aucune des informations de révocation demandées présente ;
4) aucun chemin de certification n'a été trouvé en accord avec la politique de découverte de chemin ;
5) la construction de chemin ne peut pas être effectuée à cause d'une erreur.

Les réponses de DPD construites par les serveurs SCVP ne font pas de différence entre les états 2) et 3). Cette propriété a été discutée sur la liste de diffusion du groupe de travail PKIX et a été déterminée conforme à l'intention de la [RFC3379].

1.4 Politiques de validation

Une politique de validation (comme définie dans la [RFC3379]) spécifie les règles et paramètres à utiliser par le serveur SCVP quand il valide un certificat. Dans SCVP, la politique de validation à utiliser par le serveur peut être pleinement référencée dans la demande par le client (et donc aucun paramètre supplémentaire n'est nécessaire) ou peut être référencée dans la demande du client avec des paramètres supplémentaires.

Les définitions de politique peuvent être assez longues et complexes, et certaines politiques peuvent permettre l'établissement de peu de paramètres. La demande peut donc être très simple si un identifiant d'objet (OID) est utilisé pour spécifier à la fois l'algorithme à utiliser et tous les paramètres associés de la politique de validation. La demande est peut être plus complexe si la politique de validation fixe beaucoup de paramètres mais permet au client de ne spécifier que quelques uns d'entre eux. Quand la politique de validation définit chaque paramètre nécessaire, une demande SCVP a seulement besoin de contenir le certificat à valider, la politique de validation référencée, et tous les paramètres au moment du lancement de la demande.

Un serveur publie les références des politiques de validation qu'il prend en charge. Quand ces politiques ont des paramètres qui peuvent être outrepassés, le serveur communique aussi les valeurs par défaut pour ces paramètres. Le client peut simplifier la demande en omettant un paramètre d'une demande si la valeur par défaut publiée par le serveur pour une certaine référence de politique de validation est acceptable. Cependant, si il y a le désir de montrer à quelqu'un d'autre qu'une politique de validation spécifique avec tous ses paramètres a été utilisée, le client va avoir besoin de demander au serveur l'inclusion de la politique de validation complète avec tous les paramètres dans la réponse.

Les entrées de l'algorithme de base de traitement de chemin de certification utilisées par SCVP sont définies par la [RFC3280] au paragraphe 6.1.1 et comprennent :
le certificat à valider (par valeur ou par référence) ;
l'heure de validation ;
l'ensemble initial de politique ;
le réglage initial de transposition d'inhibition de politique ;
le réglage initial d'inhibition de anyPolicy ; et
le réglage initial de politique exigée explicite.

L'algorithme de base de traitement de chemin de certification prend aussi en charge la spécification d'une ou plusieurs ancres de confiance (par valeur ou référence) en entrée. Lorsque le client demande un chemin de certification ayant pour origine une autorité de certification (CA, *Certification Authority*) spécifique, une seule ancre de confiance est spécifiée. Lorsque le client veut accepter des chemins commençant par une CA quelconque parmi plusieurs, un ensemble d'ancres de confiance est spécifié.

L'algorithme de base de traitement de chemin de certification prend aussi en charge les paramètres suivants, qui sont définis à la Section 4 de la [RFC3280] : l'usage de la clé contenue dans le certificat (par exemple, clé de chiffrement, accord de clé, signature) et les autres objets spécifiques de l'application pour lesquels la clé publique certifiée peut être utilisée.

1.5 Algorithme de validation

L'algorithme de validation est déterminé par accord entre le client et le serveur et est représenté par un OID. L'algorithme définit les vérifications à effectuer par le serveur pour déterminer si le certificat est valide. Un algorithme de validation est un des paramètres d'une politique de validation. SCVP définit un algorithme de validation de base qui met en œuvre l'algorithme de validation de chemin de base comme défini dans la [RFC3280], et il permet au client de demander des informations supplémentaires sur le certificat à valider. De nouveaux algorithmes de validation peuvent être spécifiés pour définir des vérifications supplémentaires si nécessaire. Ces nouveaux algorithmes de validation peuvent spécifier des paramètres supplémentaires. Les valeurs de ces paramètres peuvent être définies par toute politique de validation qui utilise l'algorithme ou peuvent être incluses par le client dans la demande.

Les algorithmes de validation spécifiques d'application, en plus de ceux définis dans le présent document, peuvent être définis pour satisfaire des exigences spécifiques non couvertes par l'algorithme de validation de base. Les algorithmes de validation documentés ici devraient servir de guide pour le développement d'autres algorithmes de validation spécifiques d'application. Par exemple, un nouvel algorithme de validation spécifique d'application pourrait exiger la présence d'une forme de nom particulière dans l'extension de nom de remplacement de sujet du certificat.

1.6 Exigences de validation

Pour qu'un chemin de certification soit considéré comme valide sous une politique de validation particulière, il DOIT être un chemin de certification valide comme défini dans la [RFC3280], et toutes les contraintes de la politique de validation qui s'appliquent au chemin de certification DOIVENT être vérifiées.

La vérification de révocation est un des aspects de la validation de chemin de certification définis dans la [RFC3280]. Cependant, la vérification de révocation est une caractéristique facultative dans la [RFC3280], et les informations de révocation sont distribuées dans de multiples formats. Les clients spécifient dans les demandes si la vérification de révocation devrait être effectuée et si les informations de révocation devraient être retournées dans la réponse.

Les serveurs DOIVENT être capables d'indiquer les sources des informations de révocation qu'ils sont capables de traiter :
1. CRL complètes (ou listes de révocation d'autorité complètes) ;
2. réponse OCSP, en utilisant la [RFC2560] ;
3. CRL delta ; et
4. CRL indirectes.

2. Vue d'ensemble du protocole

SCVP utilise un modèle simple de demande-réponse. C'est-à-dire que le client SCVP crée une demande et l'envoie au serveur SCVP, et ensuite le serveur SCVP crée une seule réponse et l'envoie au client. L'utilisation normale de SCVP esr supposée être sur HTTP [RFC2616], mais elle peut aussi être utilisée avec la messagerie électronique ou tout autre protocole qui peut transporter des objets signés numériquement. Les appendices A et B fournissent les détails nécessaires pour utiliser SCVP avec HTTP.

SCVP inclut deux paires de demande-réponse. La principale paire de demande-réponse traite la validation des certificats. La paire secondaire de demande-réponse est utilisée pour déterminer la liste des politiques de validation et des paramètres par défaut pris en charge par un serveur SCVP spécifique.

La Section 3 définit la demande de validation de certificat. La Section 4 définit la réponse correspondante de validation de certificat. La Section 5 définit la demande des politiques de validation. La Section 6 définit la réponse correspondante de politiques de validation. L'Appendice A enregistre les types MIME pour les demandes et réponses SCVP, et l'Appendice B décrit l'utilisation de ces types MIME avec HTTP.

3. Demande de validation

Une demande de client SCVP au serveur DOIT être un seul élément CVRequest *(demande de validation de certificat)*. Quand un CVRequest est encapsulé dans une partie de corps MIME, application/scvp-cv-demande DOIT être utilisé. Il y a deux formes de demande SCVP : non protégée et protégée. Une demande protégée est utilisée pour authentifier le client auprès du serveur ou pour fournir l'intégrité d'un client anonyme sur la paire demande-réponse. La protection est fournie par une signature numérique ou un code d'authentification de message (MAC, M*essage Authentication Code*). Dans ce dernier cas, la clé de MAC est déduite en utilisant un algorithme d'accord de clé, comme Diffie-Hellman. Si la clé publique du client est contenue dans un certificat, elle peut alors être utilisée pour authentifier le client. Plus couramment, la clé publique d'accord de clé du client va être éphémère, prenant en charge la protection de l'intégrité du client anonyme.

Un serveur PEUT exiger que toutes les demandes soient protégées, et un serveur PEUT éliminer toutes les demandes non protégées. Autrement, un serveur PEUT choisir de traiter les demandes non protégées.

Le demande non protégée consiste en une CVRequest encapsulée dans un ContentInfo *(informations de contenu)* de syntaxe de message cryptographique (CMS, *Cryptographic Message Syntax*) [RFC3852]. On donne ci-dessous une vue d'ensemble de cette structure qui n'est qu'à des fins d'illustration. L'ASN.1 définitif se trouve dans la [RFC3852]. De nombreux détails ne sont pas montrés, mais la façon dont SCVP utilise la CMS y est clairement illustrée.

```
ContentInfo {
  contentType   id-ct-scvp-certValRequest, -- (1.2.840.113549.1.9.16.1.10)
  content       CVRequest }
```

La demande protégée consiste en une CVRequest encapsulée dans des SignedData *(données signées)* ou AuthenticatedData

(données authentifiées) qui à leur tour sont encapsulées dans des ContentInfo *(informations de contenu)*. C'est-à-dire que le champ EncapsulatedContentInfo *(informations de contenu encapsulé)* des SignedData ou AuthenticatedData consiste en un champ eContentType d'une valeur de id-ct-scvp-certValRequest et un champ eContent qui contient une CVRequest codée selon les règles de codage distinctif (DER, *Distinguished Encoding Rules*). SignedData est utilisé quand la demande est signée numériquement. AuthenticatedData est utilisé avec un code d'authentication de message (MAC).

Tous les clients et serveurs SCVP DOIVENT prendre en charge SignedData pour les demandes et réponses signées. Les clients et serveurs SCVP DEVRAIENT prendre en charge AuthenticatedData pour les demandes et réponses protégées par MAC.

Si le client utilise SignedData, il DOIT avoir une clé publique qui a été liée à une identité de sujet par un certificat qui se conforme au profil de PKIX [RFC3280], et ce certificat DOIT convenir pour signer la demande SCVP. C'est à dire :

1. Si l'extension d'usage de clé est présente, la signature numérique ou le bit de non répudiation DOIT être affirmé.
2. Si l'extension d'usage de clé étendu est présente, elle DOIT contenir l'OID du client SCVP (voir le paragraphe 3.11) l'OID anyExtendedKeyUsage, ou un autre OID acceptable au serveur SCVP.

Le client DOIT mettre une référence non ambiguë à son certificat dans les SignedData qui encapsulent la demande. Le client DEVRAIT inclure son certificat dans la demande, mais PEUT omettre le certificat pour réduire la taille de la demande. Le client PEUT inclure d'autres certificats dans la demande pour aider la validation de ses certificats par le serveur SCVP. Le champ signerInfos des SignedData DOIT inclure exactement une SignerInfo. Les SignedData NE DOIVENT PAS inclure de champ unsignedAttrs *(atttributs non signés)*.

Le client DOIT mettre sa clé publique d'accord de clé, ou une référence non ambiguë à un certificat qui contient sa clé publique d'accord de clé, dans les AuthenticatedData qui encapsulent la demande. Si une paire de clés d'accord de clé éphémère est utilisée, alors la clé publique est portée dans le champ originatorKey de KeyAgreeRecipientInfo, qui exige que le client obtienne la clé publique d'accord de clé du serveur avant de calculer le code d'authentification du message (MAC). La clé d'accord de clé d'un serveur SCVP est incluse dans son message de réponse de politique de validation (voir à la Section 6). Le champ recipientInfos des AuthenticatedData DOIT inclure exactement une RecipientInfo, qui contient les informations pour le serveur SCVP. Les AuthenticatedData NE DOIVENT PAS inclure de champ unauthAttrs.

La syntaxe et la sémantique de SignedData, AuthenticatedData, et ContentInfo sont définies dans la [RFC3852]. La syntaxe et la sémantique de CVRequest sont définies ci-dessous. L'élément CVRequest contient la demande du client. La CVRequest contient les éléments cvRequestVersion et query ; la CVRequest PEUT aussi contenir les éléments requestorRef, requestNonce, requestorName, responderName, requestExtensions, signatureAlg, et hashAlg.

La CVRequest DOIT avoir la syntaxe suivante :

```
CVRequest ::= SEQUENCE {
  cvRequestVersion      ENTIER DEFAUT 1,
  query                 Query,
  requestorRef          [0] GeneralNames FACULTATIF,
  requestNonce          [1] CHAINE D'OCTETS FACULTATIF,
  requestorName         [2] GeneralName FACULTATIF,
  responderName         [3] GeneralName FACULTATIF,
  requestExtensions [4] Extensions FACULTATIF,
  signatureAlg          [5] AlgorithmIdentifier FACULTATIF,
  hashAlg               [6] IDENTIFIANT D'OBJET FACULTATIF,
  requestorText         [7] UTF8String (TAILLE (1..256)) FACULTATIF }
```

Les clients conformes DOIVENT être capables de construire des demandes avec cvRequestVersion et query. Les clients conformes DOIVENT coder en DER la CVRequest dans les messages protégés et non protégés pour faciliter la référence non ambiguë fondée sur le hachage dans le message de réponse correspondant. Les clients SCVP qui insistent pour la création d'une réponse fraîche (par exemple, pour se protéger contre une attaque en répétition ou s'assurer que les informations sont à jour) DOIVENT prendre en charge requestNonce *(nom occasionnel de demande)*. La prise en charge des éléments restants est facultative dans les mises en œuvre de client.

Les serveurs conformes DOIVENT être capables d'analyser les CVRequest qui contiennent des éléments facultatifs.

Chacun des éléments de la CVRequest est décrit dans les paragraphes suivants.

3.1 cvRequestVersion

L'élément cvRequestVersion définit la version de la CVRequest SCVP utilisée dans une demande. La réponse qui suit DOIT utiliser le même numéro de version. La valeur de l'élément cvRequestVersion DOIT être un (1) pour un client qui met en œuvre la présente spécification. Les futures mises à jour de cette spécification devront spécifier d'autres valeurs si il y a des changements à la syntaxe ou la sémantique. Cependant, de nouvelles extensions peuvent être définies sans changer le numéro de version.

Les clients SCVP DOIVENT prendre en charge l'affirmation de cette valeur et les serveurs SCVP DOIVENT être capables de traiter cette valeur.

3.2 Query

L'élément Query *(interrogation)* spécifie un ou plusieurs certificats qui sont le sujet de la demande ; les certificats peuvent être des certificats de clé publique [RFC3280] ou des certificats d'attribut [RFC3281]. Une Query DOIT contenir un élément queriedCerts ainsi qu'un élément checks *(vérifications)* et un élément validationPolicy ; une Query PEUT aussi contenir des éléments wantBack, responseFlags, serverContextInfo, validationTime, intermediateCerts, revInfos, producedAt, et queryExtensions.

Un élément Query DOIT avoir la syntaxe suivante :

```
Query ::= SEQUENCE {
  queriedCerts          CertReferences,
  checks               CertChecks,
-- Note : l'étiquette [0] n'est pas utilisée --
  wantBack             [1] WantBack FACULTATIF,
  validationPolicy       ValidationPolicy,
  responseFlags         ResponseFlags FACULTATIF,
  serverContextInfo    [2] CHAINE D'OCTETS FACULTATIF,
  validationTime       [3] GeneralizedTime FACULTATIF,
  intermediateCerts    [4] CertBundle FACULTATIF,
  revInfos             [5] RevocationInfos FACULTATIF,
  producedAt           [6] GeneralizedTime FACULTATIF,
  queryExtensions      [7] Extensions FACULTATIF }
```

La liste des références de certificat dans l'élément queriedCerts dit au serveur le ou les certificats pour lesquels le client veut des informations. L'élément checks spécifie la vérification que le client veut effectuer. L'élément wantBack spécifie les objets que le client veut que le serveur retourne dans la réponse. L'élément validationPolicy spécifie la politique de validation que le client veut que le serveur emploie. L'élément responseFlags permet au client de demander des caractéristiques facultatives pour la réponse. L'élément serverContextInfo dit au serveur que des informations supplémentaires provenant d'une demande-réponse précédentes sont désirées. L'élément validationTime dit la date et l'heure relatives auxquelles le client veut que le serveur effectue les vérifications. Les éléments intermediateCerts et revInfos fournissent le contexte de la demande du client. L'élément queryExtensions se réfère à une expansion future de la syntaxe de query. La syntaxe et la sémantique de chacun de ces éléments sont discutées dans les paragraphes qui suivent.

Les clients conformes DOIVENT être capables de construire une Query avec un élément queriedCerts qui spécifie au moins un certificat, checks, et validationPolicy. Les clients conformes à SCVP PEUVENT prendre en charge la spécification de plusieurs certificats et PEUVENT prendre en charge les éléments facultatifs dans la structure Query.

Les clients SCVP qui prennent en charge la découverte de chemin déléguée (DPD) comme défini dans la [RFC3379] DOIVENT prendre en charge wantBack et responseFlags. Les clients SCVP qui insistent pour la création d'une réponse fraîche (par exemple, pour se protéger contre une attaque en répétition ou s'assurer que les informations sont à jour) DOIVENT prendre en charge responseFlags.

Les serveurs conformes DOIVENT être capables de traiter une Query qui contient des éléments facultatifs quelconques, et DOIVENT être capables de traiter une Query qui spécifie plusieurs certificats.

3.2.1 queriedCerts

L'élément queriedCerts est une SÉQUENCE de un ou plusieurs certificats, dont chacun est un sujet de la demande. Les

certificats spécifiés sont soit des certificats de clé publique, soit des certificats d'attribut ; si plus d'un certificat est spécifié, tous doivent être du même type. Chaque certificat est soit directement inclus, soit référencé. Quand il est référencé, une valeur de hachage de l'élément référencé est incluse pour s'assurer que le client SCVP et le serveur SCVP ont tous deux obtenu le même certificat quand ils vont chercher le certificat référencé. Les références de certificat utilisent le type SCVPCertID, qui est décrit ci-dessous. Une seule demande PEUT contenir à la fois des certificats directement inclus et des certificats référencés.

CertReferences a la syntaxe suivante :

```
CertReferences ::= CHOIX {
  pkcRefs    [0] TAILLE DE SÉQUENCE (1..MAX) DE PKCReference,
  acRefs     [1] TAILLE DE SÉQUENCE (1..MAX) DE ACReference }

PKCReference ::= CHOIX {
  cert       [0] Certificate,
  pkcRef     [1] SCVPCertID }

ACReference ::= CHOIX {
  attrCert   [2] AttributeCertificate,
  acRef      [3] SCVPCertID }

SCVPCertID ::= SÉQUENCE {
  certHash          CHAINE D'OCTETS,
  issuerSerial      SCVPIssuerSerial,
  hashAlgorithm  AlgorithmIdentifier DEFAUT { algorithme sha-1 } }
```

La définition ASN.1 de Certificate est importée de la [RFC3280] et la définition de AttributeCertificate est importée de la [RFC3281].

Quand on crée un SCVPCertID, le certHash est calculé sur le certificat entier codé en DER incluant la signature. L'algorithme de hachage utilisé pour calculer certHash est spécifié dans hashAlgorithm. L'algorithme de hachage utilisé pour calculer certHash DEVRAIT être un des algorithmes de hachage spécifiés dans l'élément hashAlgorithms du message de réponse de politique de validation du serveur.

Quand on code SCVPIssuerSerial, serialNumber est le numéro de série qui identifie de façon univoque le certificat. Pour les certificats de clé publique, l'élément issuer DOIT contenir seulement le nom du producteur provenant du certificat codé dans le choix directoryName de GeneralNames. Pour les certificats d'attribut, le issuer DOIT contenir le champ Nom du producteur provenant du certificat d'attribut.

Les clients conformes DOIVENT être capables de référencer un certificat par inclusion directe. Les clients DEVRAIENT être capables de spécifier un certificat en utilisant le SCVPCertID. Les clients conformes PEUVENT être capables de référencer plusieurs certificats et PEUVENT être capables de référencer des certificats de clé publique et d'attribut.

Les mises en œuvre conformes de serveur SCVP DOIVENT être capables de traiter des CertReferences avec plusieurs certificats. Les mises en œuvre conformes de serveur SCVP DOIVENT être capables d'analyse des CertReferences qui contiennent des certificats de clé publique ou d'attribut. Les mises en œuvre conformes de serveur SCVP DOIVENT être capables d'analyser les choix cert et pkcRef dans PKCReference. Les mises en œuvre conformes de serveur SCVP qui traitent les certificats d'attribut DOIVENT être capables d'analyser les choix attrCert et acRef dans ACReference.

3.2.2 checks

L'élément checks décrit les vérifications que le client SCVP veut que le serveur SCVP effectue sur le ou les certificats dans l'élément queriedCerts. L'élément checks contient une séquence d'identifiants d'objet (OID). Chaque OID dit au serveur SCVP quelle vérification le client attend que le serveur effectue. Pour chaque vérification spécifiée dans la demande, le serveur SCVP DOIT effectuer la vérification demandée, ou retourner une erreur. Un serveur peut choisir d'effectuer des vérifications additionnelles (par exemple, un serveur à qui il est seulement demandé de construire un chemin de certification validé peut choisir d'effectuer aussi des vérifications d'état de révocation) bien que le serveur ne puisse pas indiquer dans la réponse que des vérifications supplémentaires ont été effectuées, sauf dans le cas d'une réponse d'erreur.

L'élément checks utilise le type CertChecks, qui a la syntaxe suivante :

CertChecks ::= TAILLE DE SÉQUENCE (1..MAX) DE IDENTIFIANT D'OBJET

Pour les certificats de clé publique, les vérifications suivantes sont définies dans le présent document :
- id-stc-build-pkc-path : construit un chemin de certification prospectif à une ancre de confiance (comme défini au paragraphe 6.1 de la [RFC3280]) ;
- id-stc-build-valid-pkc-path : construit un chemin de certification validé à une ancre de confiance (vérification de révocation non exigée) ;
- id-stc-build-status-checked-pkc-path : construit un chemin de certification validé à une ancre de confiance et effectue les vérifications d'état de révocation sur le chemin de certification.

Les mises en œuvre conformes de serveur SCVP qui prennent en charge la découverte de chemin déléguée (DPD) comme défini dans la [RFC3379] DOIVENT prendre en charge la vérification id-stc-build-pkc-path. Les mises en œuvre conformes de serveur SCVP qui prennent en charge la validation de chemin déléguée (DPV) comme défini dans la [RFC3379] DOIVENT prendre en charge les vérifications id-stc-build-valid-pkc-path et id-stc-build-status-checked-pkc-path.

Pour les certificats d'attribut, les vérifications suivantes sont définies dans le présent document :
- id-stc-build-aa-path : construire un chemin de certification prospectif à une ancre de confiance pour le producteur de certificat d'attribut (AC, *Attribute Certificate*) ;
- id-stc-build-valid-aa-path : construire un chemin de certification validé à une ancre de confiance pour le producteur d'AC ;
- id-stc-build-status-checked-aa-path : construire un chemin de certification validé à une ancre de confiance pour le producteur d'AC et effectuer les vérifications d'état de révocation sur le chemin de certification pour le producteur d'AC
- id-stc-status-check-ac-and-build-status-checked-aa-path : construire un chemin de certification validé à une ancre de confiance pour le producteur d'AC et effectuer les vérifications d'état de révocation sur le AC ainsi que sur le chemin de certification pour le producteur d'AC.

Les mises en œuvre conformes de serveur SCVP PEUVENT prendre en charge les vérifications de certificat d'attribut.

À cette fin, les OID suivants sont définis :

IDENTIFIANT D'OBJET id-stc ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) 17 }

id-stc-build-pkc-path	IDENTIFIANT D'OBJET ::= { id-stc 1 }
id-stc-build-valid-pkc-path	IDENTIFIANT D'OBJET ::= { id-stc 2 }
id-stc-build-status-checked-pkc-path	IDENTIFIANT D'OBJET ::= { id-stc 3 }
id-stc-build-aa-path	IDENTIFIANT D'OBJET ::= { id-stc 4 }
id-stc-build-valid-aa-path	IDENTIFIANT D'OBJET ::= { id-stc 5 }
id-stc-build-status-checked-aa-path	IDENTIFIANT D'OBJET ::= { id-stc 6 }
id-stc-status-check-ac-and-build-status-checked-aa-path	IDENTIFIANT D'OBJET ::= { id-stc 7 }

D'autres spécifications pourront définir des vérifications supplémentaires.

Les mises en œuvre de client conformes DOIVENT prendre en charge l'assertion d'au moins une des vérifications standard. Les clients conformes PEUVENT prendre en charge l'assertion de multiples vérifications. Les clients conformes n'ont pas besoin de prendre en charge toutes les vérifications définies dans cette section.

3.2.3 wantBack

L'élément facultatif wantBack *(veut en retour)* décrit toutes les informations que le client SCVP veut obtenir du serveur SCVP pour le ou les certificats dans l'élément queriedCerts en plus des résultats des vérifications spécifiées dans l'élément checks. Si il est présent, l'élément wantBack DOIT contenir une séquence d'identifiants d'objet (OID). Chaque OID dit au serveur SCVP ce que le client veut savoir sur l'élément queriedCerts. Pour chaque type d'information spécifié dans la demande, le serveur DOIT retourner les informations concernant ses trouvailles (dans une réponse de succès).

Par exemple, une demande pourrait inclure un élément checks qui spécifie seulement la construction du chemin de certification et inclure un élément wantBack qui demande le retour du chemin de certification construit par le serveur. Dans ce cas, la réponse ne va pas inclure un état pour la validation du chemin de certification, mais elle va inclure un chemin de certification prospectif. Un client qui veut effectuer sa propre validation de chemin de certification pourrait utiliser une

demande de cette forme.

Autrement, une demande pourrait inclure un élément checks qui demande au serveur de construire un chemin de certification et de le valider, incluant la vérification de révocation, et ne pas inclure d'élément wantBack. Dans ce cas, la réponse va seulement inclure un état pour la validation du chemin de certification. Un client qui délègue complètement la validation du chemin de certification pourrait utiliser une demande de cette forme.

L'élément wantBack utilise le type WantBack, qui a la syntaxe suivante :

WantBack ::= TAILLE DE SÉQUENCE (1..MAX) DE IDENTIFIANT D'OBJET

Pour les certificats de clé publique, les éléments wantBack suivants sont définis dans le présent document :
id-swb-pkc-cert : certificat qui était le sujet de la demande ;
id-swb-pkc-best-cert-path : chemin de certification construit pour le certificat incluant le certificat qui a été validé ;
id-swb-pkc-revocation-info : preuve de l'état de révocation pour chaque certificat dans le chemin de certification ;
id-swb-pkc-public-key-info : clé publique provenant du certificat qui était le sujet de la demande ;
id-swb-pkc-all-cert-paths : ensemble de chemins de certification pour le certificat sujet de la demande ;
id-swb-pkc-ee-revocation-info : preuve de l'état de révocation pour le certificat d'entité d'extrémité dans le chemin de certification ;
id-swb-pkc-CAs-revocation-info : preuve de l'état de révocation pour chaque certificat de CA dans le chemin de certification.

Toutes les mises en œuvre conformes de serveur SCVP DOIVENT prendre en charge les éléments wantBack id-swb-pkc-cert et id-swb-pkc-public-key-info. Les mises en œuvre conformes de serveur SCVP qui prennent en charge la découverte de chemin déléguée (DPD) comme défini dans la [RFC3379] DOIVENT prendre en charge les éléments wantBack id-swb-pkc-best-cert-path et id-swb-pkc-revocation-info.

SCVP fournit deux méthodes pour qu'un client obtienne plusieurs chemins de certification pour un certificat. Le client pourrait utiliser serverContextInfo pour demander un chemin à la fois (voir au paragraphe 3.2.6). Après l'obtention de chaque chemin, le client pourrait soumettre le serverContextInfo provenant de la demande précédente pour obtenir un autre chemin jusqu'à ce que soit le client trouve un chemin convenable, soit que le serveur indique (en ne retournant pas de serverContextInfo) qu'il n'y a plus de chemins disponibles. Autrement, le client pourrait envoyer une seule demande avec un élément wantBack id-swb-pkc-all-cert-paths, et le serveur retournerait tous les chemins disponibles dans une seule réponse.

Le serveur peut, à sa discrétion, limiter le nombre de chemins qu'il retourne en réponse au id-swb-pkc-all-cert-paths. Quand la demande inclut un wantBack id-swb-pkc-all-cert-paths, la réponse NE DEVRAIT PAS inclure de serverContextInfo.
Pour les certificats d'attribut, les wantBack suivants sont définis dans le présent document :
id-swb-ac-cert : certificat d'attribut qui était le sujet de la demande ;
id-swb-aa-cert-path : chemin de certification construit pour l'AC producteur du certificat ;
id-swb-ac-revocation-info : preuve de l'état de révocation pour chaque certificat dans le chemin de certification de l'AC productrice ; et
id-swb-aa-revocation-info : preuve de l'état de révocation pour le certificat d'attribut.

Les mises en œuvre conformes de serveur SCVP PEUVENT prendre en charge les éléments wantBack de certificat d'attribut.

Le wantBack suivant peut être utilisé pour les clés publiques ou les certificats d'attribut :
id-swb-relayed-responses : toute réponse SCVP reçue par le serveur utilisée pour générer la réponse à cette interrogation.

Les serveurs SCVP conformes PEUVENT prendre en charge le wantBack id-swb-relayed-responses.

À cette fin, les OID suivants sont définis :

```
IDENTIFIANT D'OBJET id-swb ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5)
                                 pkix(7) 18 }
IDENTIFIANT D'OBJET id-swb-pkc-best-cert-path   ::= { id-swb 1 }
IDENTIFIANT D'OBJET id-swb-pkc-revocation-info ::= { id-swb 2 }
IDENTIFIANT D'OBJET id-swb-pkc-public-key-info ::= { id-swb 4 }
IDENTIFIANT D'OBJET id-swb-aa-cert-path         ::= { id-swb 5 }
IDENTIFIANT D'OBJET id-swb-aa-revocation-info   ::= { id-swb 6 }
```

```
IDENTIFIANT D'OBJET id-swb-ac-revocation-info   ::= { id-swb 7 }
IDENTIFIANT D'OBJET id-swb-relayed-responses    ::= { id-swb 9 }
IDENTIFIANT D'OBJET id-swb-pkc-cert             ::= { id-swb 10}
IDENTIFIANT D'OBJET id-swb-ac-cert              ::= { id-swb 11}
IDENTIFIANT D'OBJET id-swb-pkc-all-cert-paths   ::= { id-swb 12}
IDENTIFIANT D'OBJET id-swb-pkc-ee-revocation-info ::= { id-swb 13}
IDENTIFIANT D'OBJET id-swb-pkc-CAs-revocation-info ::= { id-swb 14}
```

D'autres spécifications pourront définir des wantBack supplémentaires.

Les mises en œuvre de client conformes qui prennent en charge la validation de chemin déléguée (DPV, *Delegated Path Validation*) comme défini dans la [RFC3379] DEVRAIENT prendre en charge l'assertion d'au moins un wantBack. Les mises en œuvre de client conformes qui prennent en charge la découverte de chemin déléguée (DPD) comme défini dans la [RFC3379] DOIVENT prendre en charge l'assertion d'au moins un wantBack. Les clients conformes PEUVENT prendre en charge l'assertion de plusieurs wantBack. Les clients conformes n'ont pas besoin de prendre en charge tous les wantBack définis dans cette section.

3.2.4 validationPolicy

L'élément validationPolicy définit la politique de validation que le client veut que le serveur SCVP utilise durant la validation de certificat. Si cette politique ne peut pas être utilisée pour une raison quelconque, alors le serveur DOIT retourner une réponse d'erreur.

Une politique de validation DOIT définir des valeurs par défaut pour tous les paramètres nécessaires pour traiter une demande SCVP. Pour chaque paramètre, une politique de validation peut permettre au client de spécifier une valeur non par défaut ou interdire d'utiliser une valeur non par défaut. Si le client souhaite utiliser les valeurs par défaut pour tous les paramètres, alors le client a seulement besoin de fournir une référence à la politique dans cet élément. Si le client souhaite utiliser les valeurs non par défaut pour un ou plusieurs paramètres, alors le client fournit une référence à la politique plus tous les paramètres nécessaires pour achever la demande dans cet élément. Si il y a des conflits entre la politique référencée dans la demande et des valeurs de paramètres fournies dans la demande, le serveur DOIT alors retourner une réponse d'erreur.

La syntaxe de l'élément validationPolicy est :

```
ValidationPolicy ::= SEQUENCE {
  validationPolRef        ValidationPolRef,
  validationAlg             [0] ValidationAlg FACULTATIF,
  userPolicySet             [1] TAILLE DE SÉQUENCE (1..MAX) DE IDENTIFIANT D'OBJET FACULTATIF,
  inhibitPolicyMapping    [2] BOOLEEN FACULTATIF,
  requireExplicitPolicy     [3] BOOLEEN FACULTATIF,
  inhibitAnyPolicy          [4] BOOLEEN FACULTATIF,
  trustAnchors              [5] TrustAnchors FACULTATIF,
  keyUsages                 [6] SEQUENCE DE KeyUsage FACULTATIF,
  extendedKeyUsages       [7] SEQUENCE DE KeyPurposeId FACULTATIF,
  specifiedKeyUsages      [8] SEQUENCE DE KeyPurposeId FACULTATIF }
```

L'élément validationPolRef est exigé, mais les éléments restants sont facultatifs. Les éléments facultatifs sont utilisés pour fournir les paramètres de politique de validation. Quand le client utilise les valeurs de politique de validation par défaut pour tous les paramètres, tous les éléments facultatifs sont absents.

Au minimum, les mises en œuvre conformes de client SCVP DOIVENT prendre en charge l'élément validationPolRef. Les mises en œuvre de client conformes PEUVENT prendre en charge des éléments facultatifs dans ValidationPolicy ou tous.

Les serveurs SCVP conformes DOIVENT prendre en charge le traitement d'une ValidationPolicy qui contient des éléments facultatifs ou tous.

L'élément validationAlg spécifie l'algorithme de validation. L'élément userPolicySet fournit un ensemble acceptable de politiques de certificat. L'élément inhibitPolicyMapping inhibe la transposition de politique de certificat durant la validation de chemin de certification. L'élément requireExplicitPolicy exige au moins une politique de certificat valide dans l'extension de politiques de certificat. L'élément inhibitAnyPolicy indique si l'OID de politique de certificat anyPolicy est

traité ou ignoré lors de l'évaluation de la politique de certificat. L'élément trustAnchors indique les ancres de confiance qui sont acceptables au client. L'élément keyUsages indique l'usage technique de la clé publique qui est à confirmer par le serveur comme acceptable. L'élément extendedKeyUsages indique l'usage spécifique de l'application de la clé publique qui est à confirmer par le serveur comme acceptable. La syntaxe et la sémantique de chacun de ces éléments sont discutées dans les paragraphes qui suivent.

3.2.4.1 validationPolRef

La référence à la politique de validation est un OID qui représente une politique de validation particulière sur laquelle le client et le serveur doivent s'être accordés.

La syntaxe de l'élément validationPolRef est :

```
ValidationPolRef::= SEQUENCE {
  valPolId              IDENTIFIANT D'OBJET,
  valPolParams       ANY DEFINI PAR valPolId FACULTATIF }
```

Lorsque une politique de validation prend en charge des réglages de paramètre spécifiques de la politique supplémentaires, ces valeurs sont spécifiées en utilisant l'élément valPolParams. La syntaxe et la sémantique de la structure des paramètres sont définies par l'identifiant d'objet codé comme le valPolId. Lorsque une politique de validation n'a pas de paramètres, comme la politique de validation par défaut (voir le paragraphe 3.2.4.1.1) cet élément DOIT être omis.

Les paramètres spécifiés dans cet élément sont indépendants de l'algorithme de validation et des paramètres de l'algorithme de validation (voir au paragraphe 3.2.4.2). Par exemple, un serveur peut prendre en charge une politique de validation dans laquelle il valide un certificat en utilisant le nom de l'algorithme de validation et fait aussi une détermination concernant le crédit qu'on peut accorder au sujet. Dans ce cas, les paramètres de la politique de validation pourraient être utilisés pour spécifier la valeur de la transaction. Les paramètres de l'algorithme de validation sont utilisés pour spécifier l'identifiant et le nom d'application pour le nom d'algorithme de validation.

Les mises en œuvre conformes de client SCVP DOIVENT prendre en charge la spécification d'une politique de validation. Les mises en œuvre conformes de client SCVP PEUVENT être capables de spécifier les paramètres d'une politique de validation. Les mises en œuvre conformes de serveur SCVP DOIVENT être capables de traiter les valPolId et PEUVENT être capables de traiter les valPolParams.

3.2.4.1.1 Politique de validation par défaut

Le client peut demander la politique de validation par défaut ou une autre politique de validation du serveur SCVP. La politique de validation par défaut correspond au traitement standard du chemin de certification, comme défini dans la [RFC3280] avec des valeurs par défaut choisies par le serveur (par exemple, avec des réglages de politique et ancres de confiance déterminés par un serveur). Les valeurs par défaut peuvent être distribuées hors bande ou en utilisant le mécanisme de demande de politique (voir à la Section 5). Ce mécanisme permet le déploiement d'un serveur SCVP sans obtenir un nouvel identifiant d'objet.

L'identifiant d'objet qui identifie la politique de validation par défaut est :

```
IDENTIFIANT D'OBJET id-svp ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5)
                                 pkix(7) 19 }

IDENTIFIANT D'OBJET id-svp-defaultValPolicy ::= { id-svp 1 }
```

La politique de validation par défaut DOIT utiliser l'algorithme de validation de base comme algorithme de validation par défaut (paragraphe 3.2.4.2.1) et n'a pas de paramètre de politique de validation (paragraphe 3.2.4.1).

Quand il utilise la politique de validation par défaut, le client peut outrepasser toute valeur par défaut de paramètres en fournissant une valeur spécifique dans la demande. Le serveur SCVP DOIT utiliser les valeurs de paramètres fournies ou retourner une réponse d'erreur.

Les mises en œuvre conformes de serveurs SCVP DOIVENT prendre en charge la politique par défaut. Cependant, un serveur SCVP peut être configuré à envoyer une réponse d'erreur à toutes les demandes qui utilisent la politique par défaut

pour satisfaire aux exigences de la politique locale de sécurité.

3.2.4.2 validationAlg

L'élément facultatif validationAlg définit l'algorithme de validation à utiliser par le serveur SCVP durant la validation de certificat. La valeur de cet élément peut être déterminée par accord entre le client et le serveur. L'algorithme de validation est représenté par un identifiant d'objet.

La syntaxe de l'élément validationAlg est :

```
ValidationAlg ::= SEQUENCE {
  valAlgId          IDENTIFIANT D'OBJET,
  parameters        ANY DEFINI PAR valAlgId FACULTATIF }
```

Le paragraphe qui suit spécifie l'algorithme de base de validation et le nom d'algorithme de validation.

Les serveurs SCVP DOIVENT reconnaître et prendre en charge les deux algorithmes de validation définis dans cette section. Les clients SCVP qui prennent en charge l'assertion explicite de l'algorithme de validation DOIVENT prendre en charge l'algorithme de base de validation et DEVRAIENT prendre en charge le nom d'algorithme de validation. D'autres algorithmes de validation pourront être spécifiés dans d'autres documents à utiliser avec des applications spécifiques. Les clients et les serveurs SCVP PEUVENT prendre en charge chacun de ces algorithmes de validation.

3.2.4.2.1 Algorithme de validation de base

Le client peut demander à utiliser l'algorithme de validation SCVP de base ou un autre algorithme. Pour les certificats d'identité, l'algorithme de validation de base DOIT mettre en œuvre l'algorithme de validation de chemin de certification comme défini dans la Section 6 de la [RFC3280]. Pour les certificats d'attribut, l'algorithme de validation de base DOIT mettre en œuvre la validation de chemin de certification comme défini dans la Section 5 de la [RFC3281]. D'autres algorithmes de validation PEUVENT mettre en œuvre des fonctions sur et au-dessus de celles de l'algorithme de base, mais les algorithmes de validation DOIVENT générer des résultats conformes à l'algorithme de validation de base. C'est-à-dire que aucune des exigences de validation de l'algorithme de base ne peut être omise par de nouveaux algorithmes de validation définis. Cependant, d'autres algorithmes de validation PEUVENT rejeter des chemins qui sont valides en utilisant l'algorithme de validation de base. L'identifiant d'objet pour identifier l'algorithme de validation de base est :

```
IDENTIFIANT D'OBJET id-svp-basicValAlg ::= { id-svp 3 }
```

Quand id-svp-basicValAlg apparaît dans valAlgId, l'élément paramètres DOIT être absent.

3.2.4.2.2 Erreurs d'algorithme de validation de base

Les erreurs suivantes sont définies pour l'algorithme de validation de base pour être incluses dans l'élément validationErrors dans la réponse (paragraphe 4.9.6). Ces erreurs peuvent être utilisées par tout autre algorithme de validation cat tous les algorithmes de validation DOIVENT mettre en œuvre les fonctions de l'algorithme de validation de base.

```
IDENTIFIANT D'OBJET :id-bvae := id-svp-basicValAlg
IDENTIFIANT D'OBJET :id-bvae-expired  ::= { id-bvae 1 }
IDENTIFIANT D'OBJET :id-bvae-not-yet-valid ::= { id-bvae 2 }
IDENTIFIANT D'OBJET :id-bvae-wrongTrustAnchor ::= { id-bvae 3 }
IDENTIFIANT D'OBJET :id-bvae-noValidCertPath ::= { id-bvae 4 }
IDENTIFIANT D'OBJET :id-bvae-revoked ::= { id-bvae 5 }
IDENTIFIANT D'OBJET :id-bvae-invalidKeyPurpose ::= { id-bvae 9 }
IDENTIFIANT D'OBJET :id-bvae-invalidKeyUsage ::= { id-bvae 10 }
IDENTIFIANT D'OBJET :id-bvae-invalidCertPolicy ::= { id-bvae 11 }
```

La valeur id-bvae-expired signifie que l'heure de validation utilisée pour la demande était plus tard que l'heure notAfter dans le certificat de fin (le certificat spécifié dans l'élément queriedCerts).

La valeur id-bvae-not-yet-valid signifie que l'heure de validation utilisée pour la demande était avant l'heure notBefore

dans le certificat de fin.

La valeur id-bvae-wrongTrustAnchor signifie que un chemin de certification ne pourrait pas être construit pour la ou les ancres de confiance spécifiées par le client, mais un chemin existe pour une des ancres de confiance spécifiées dans la politique de validation par défaut du serveur.

La valeur id-bvae-noValidCertPath signifie que le serveur ne pourrait pas construire une séquence de certificats intermédiaires entre l'ancre de confiance et le certificat cible qui satisfaisait la demande.

La valeur id-bvae-revoked signifie que le certificat de fin a été révoqué.

La valeur id-bvae-invalidKeyPurpose signifie que l'extension d'usage de clé étendu ([RFC3280], paragraphe 4.2.1.13) dans le certificat de fin ne satisfait pas la politique de validation.

La valeur id-bvae-invalidKeyUsage signifie que l'extension keyUsage ([RFC3280], paragraphe 4.2.1.3) dans le certificat de fin ne satisfait pas la politique de validation. Par exemple, l'extension keyUsage dans le certificat peut affirmer seulement le bit keyEncipherment, mais la politique de validation spécifie dans l'élément keyUsages que digitalSignature est exigé.

La valeur id-bvae-invalidCertPolicy signifie que le chemin n'est pas valide dans les politiques spécifiées dans la politique de l'utilisateur et que des politiques explicites sont exigées. C'est-à-dire, le valid_policy_tree est NUL et la variable explicit_policy est zéro ([RFC3280], paragraphe 6.1.5).

3.2.4.2.3 Nom d'algorithme de validation

Le nom d'algorithme de validation permet au client de spécifier un ou plusieurs noms de sujet qui DOIVENT apparaître dans le certificat de fin en plus des exigences spécifiées pour l'algorithme de validation de base. Le nom d'algorithme de validation permet au client de fournir un identifiant d'application et un nom au serveur. L'identifiant d'application définit les règles de correspondance de nom à utiliser pour comparer le nom fourni dans la demande avec les noms dans le certificat.

IDENTIFIANT D'OBJET id-svp-nameValAlg ::= { id-svp 2 }

Quand le id-svp-nameValAlg apparaît comme valAlgId, les paramètres DOIVENT utiliser la syntaxe de NameValidationAlgParms :

```
NameValidationAlgParms ::= SEQUENCE {
  nameCompAlgId     IDENTIFIANT D'OBJET,
  validationNames   GeneralNames }
```

GeneralNames est défini dans la [RFC3280].

Si plus d'un nom est fourni dans la valeur de validationNames, tous les noms DOIVENT être du même type. Le certificat doit contenir un nom correspondant pour chacun des noms fournis dans validationNames en accord avec les règles de correspondance de nom associées au nameCompAlgId. Cette spécification définit trois ensembles de règles de correspondance de noms.

Si le nameCompAlgId fourni dans la demande est id-nva-dnCompAlg, alors le GeneralNames fourni dans la demande DOIT être un directoryName, et les règles de correspondance à utiliser sont définies dans la [RFC3280]. Le certificat doit contenir un nom correspondant soit dans le champ subject soit dans un directoryName dans l'extension subjectAltName. Cette spécification définit comme suit l'OID pour id-nva-dnCompAlg :

IDENTIFIANT D'OBJET id-nva-dnCompAlg ::= { id-svp 4 }

Si le nameCompAlgId fourni dans la demande est id-kp-serverAuth [RFC3280], alors le GeneralNames fourni dans la demande DOIT être un dNSName, et les règles de correspondance à utiliser sont définies dans la [RFC3280].

Si une extension subjectAltName est présente et inclut un ou plusieurs noms de type dNSName, une correspondance dans tout nom de l'ensemble est considéré comme acceptable. Si l'extension subjectAltName est omise, ou n'inclut pas de nom de type dNSName, le champ (le plus spécifique) Nom commun dans le champ subject du certificat DOIT être utilisé.

Les noms peuvent contenir le caractère générique *, qui est considéré correspondre à tout composant de nom de domaine.

C'est-à-dire que *.a.com correspond à foo.a.com mais pas à bar.foo.a.com.

Si le nameCompAlgId fourni dans la demande est id-kp-mailProtection [RFC3280], alors le GeneralNames fourni dans la demande DOIT être un rfc822Name, et les règles de correspondance sont définies dans la [RFC3850].

Les serveurs SCVP conformes DOIVENT prendre en charge le nom d'algorithme de validation et les règles de correspondance associées à id-nva-dnCompAlg, id-kp- serverAuth, et id-kp-mailProtection. Les serveurs SCVP PEUVENT prendre en charge d'autres règles de correspondance de noms.

3.2.4.2.4 Erreurs d'algorithme de validation de nom

Les erreurs suivantes sont définies pour l'algorithme de validation de nom :

```
IDENTIFIANT D'OBJET id-nvae ::= id-svp-nameValAlg
IDENTIFIANT D'OBJET id-nvae-name-mismatch  ::= { id-nvae 1 }
IDENTIFIANT D'OBJET id-nvae-no-name        ::= { id-nvae 2 }
IDENTIFIANT D'OBJET id-nvae-unknown-alg    ::= { id-nvae 3 }
IDENTIFIANT D'OBJET id-nvae-bad-name       ::= { id-nvae 4 }
IDENTIFIANT D'OBJET id-nvae-bad-name-type  ::= { id-nvae 5 }
IDENTIFIANT D'OBJET id-nvae-mixed-names    ::= { id-nvae 6 }
```

La valeur id-nvae-name-mismatch signifie que le client a fourni un nom avec la demande, que le serveur a reconnu et le serveur a trouvé un type de com correspondant dans le certificat, mais n'a pas été capable de trouver une correspondance pour le nom fourni. Par exemple, le client a fourni un nom DNS de exemple1.com, et le certificat contenait un nom DNS de exemple.com.

La valeur id-nvae-no-name signifie que le client a fourni un nom avec la demande, que le serveur a reconnu, mais le serveur n'a pas pu trouver le type de nom correspondant dans le certificat. Par exemple, le client a fourni un nom DNS de exemple1.com, et le certificat contenait seulement un rfc822Name de user@exemple.com.

La valeur id-nvae-unknown-alg signifie que le client a fourni un nameCompAlgId que le serveur ne reconnaît pas.

La valeur id-nvae-bad-name signifie que le client a fourni dans la demande soit un nom vide, soit un nom mal formé.

La valeur id-nvae-bad-name-type signifie que le client a fourni un nom inapproprié pour l'identifiant d'application. Par exemple, le client a spécifié un nameCompAlgId de id-kp-serverAuth, et un rfc822Name de user@exemple.com.

La valeur id-nvae-mixed-names signifie que le client a fourni plusieurs noms de différents types dans la demande.

3.2.4.3 userPolicySet

L'élément userPolicySet spécifie une liste d'identifiants de politique de certificat que le serveur SCVP DOIT utiliser quand il construit et valide un chemin de certification. L'élément userPolicySet spécifie le user-initial-policy-set comme défini dans la Section 6 de la [RFC3280]. Un userPolicySet contenant l'OID anyPolicy indique un user-initial-policy-set de any-policy.

Les clients SCVP DEVRAIENT prendre en charge l'élément userPolicySet dans les demandes, et les serveurs SCVP DOIVENT prendre en charge l'élément userPolicySet dans les demandes.

3.2.4.4 inhibitPolicyMapping

L'élément inhibitPolicyMapping spécifie une entrée de l'algorithme de validation de chemin de certification, et il contrôle si la transposition de politique est permise durant la validation de chemin de certification (voir le paragraphe 6.1.1 de la [RFC3280]). Si le client veut que le serveur inhibe les transpositions de politique, inhibitPolicyMapping est réglé à VRAI dans la demande. Les clients SCVP PEUVENT prendre en charge l'inhibition de transposition de politique. Les serveurs SCVP DEVRAIENT prendre en charge l'inhibition de transposition de politique.

3.2.4.5 requireExplicitPolicy

L'élément requireExplicitPolicy spécifie une entrée de l'algorithme de validation de chemin de certification, et il contrôle si il doit y avoit au moins une politique valide dans l'extension de politiques de certificat (voir la [RFC3280], paragraphe 6.1.1). Si le client veut que le serveur exige au moins une politique, requireExplicitPolicy est réglé à VRAI dans la demande.

Les clients SCVP PEUVENT prendre en charge d'exiger des politiques explicites. Les serveurs SCVP DEVRAIENT prendre en charge l'exigence de politiques explicites.

3.2.4.6 inhibitAnyPolicy

L'élément inhibitAnyPolicy spécifie une entrée à l'algorithme de validation de chemin de certification (voir la [RFC3280], paragraphe 6.1.1) et il contrôle si l'OID anyPolicy est traité ou ignoré lors de l'évaluation de la politique de certificat. Si le client veut que le serveur ignore l'OID anyPolicy, inhibitAnyPolicy DOIT être réglé à VRAI dans la demande.

Les clients SCVP PEUVENT prendre en charge d'ignorer l'OID anyPolicy. Les serveurs SCVP DEVRAIENT prendre en charge d'ignorer l'OID anyPolicy.

3.2.4.7 trustAnchors

L'élément trustAnchors spécifie les ancres de confiance auxquelles le chemin de certification doit se terminer si le chemin doit être considéré comme valide pour la demande par le serveur SCVP. Si un élément trustAnchors est présent, le serveur NE DOIT PAS considérer d'autres chemins de certification se terminant à d'autres ancres de confiance comme valides.

Le type TrustAnchors contient une ou plusieurs spécifications d'ancre de confiance. Une référence de certificat peut être utilisée pour identifier l'ancre de confiance par un hachage de certificat et un nom distinctif avec un numéro de série. Autrement, les ancres de confiance peuvent être fournies directement. L'ordre des spécifications d'ancre de confiance au sein de la séquence n'est pas important. Tout certificat de CA qui satisfait les exigences de la [RFC3280] pour les certificats de signature peut être fourni comme ancre de confiance. Si une ancre de confiance est fournie qui ne satisfait pas ces exigences, le serveur DOIT retourner une réponse d'erreur.

L'ancre de confiance elle-même, sans considération de sa forme, NE DOIT PAS être incluse dans un chemin de certification retourné par le serveur SCVP.

TrustAnchors a la syntaxe suivante :

TrustAnchors ::= TAILLE DE SÉQUENCE (1..MAX) DE PKCReference

Les serveurs SCVP DOIVENT prendre en charge trustAnchors. Les clients SCVP DEVRAIENT prendre en charge trustAnchors.

3.2.4.8 keyUsage

L'extension keyUsage ([RFC3280], paragraphe 4.2.1.3) dans le certificat définit l'objet technique (comme le chiffrement, la signature, et la signature de CRL) de la clé contenue dans le certificat. Si le client souhaite confirmer l'usage technique, il peut alors communiquer l'usage qu'il veut valider par la même structure en utilisant la même sémantique que définie dans la [RFC3280]. Par exemple, si le client a obtenu le certificat dans le contexte d'une signature numérique, il peut le confirmer en incluant une structure keyUsage avec le bit signature numérique établi.

Si l'élément keyUsage est présent et contient une séquence vide, il indique que le client n'exige pas d'usage de clé particulier.

Si l'élément keyUsage contient une ou plusieurs définitions de keyUsage, le certificat DOIT alors satisfaire au moins une des définitions de keyUsage spécifiées. Si le client veut accepter plusieurs possibilités, il passe alors dans une séquence de schémas possibles. Chaque keyUsage peut contenir un ensemble de un ou plusieurs bits établis dans la demande, tous les bits DOIVENT être établis dans le certificat pour correspondre à une instance de keyUsage dans la demande SCVP. L'extension keyUsage de certificat peut contenir plus d'usages que demandé. Par exemple, si un client souhaite vérifier la signature numérique ou la non répudiation, il fournit deux valeurs de keyUsage, une avec Signature numérique établi et

l'autre avec Non répudiation établi. Si l'extension keyUsage est absente du certificat, le certificat DOIT être considéré comme bon pour tous les usages et donc tout schéma dans la demande SCVP va correspondre.

Les clients SCVP DEVRAIENT prendre en charge keyUsage, et les serveurs SCVP DOIVENT prendre en charge keyUsage.

3.2.4.9 extendedKeyUsages

L'extension usage de clé étendu ([RFC3280], paragraphe 4.2.1.13) définit des objets techniques plus spécifiques, en plus, ou à la place des objets indiqués dans l'extension usage de clé, pour lesquels la clé publique certifiée peut être utilisée. Si le client va accepter des certificats qui sont cohérents avec une ou des valeurs particulières dans l'extension usage de clé étendu, il peut alors communiquer les usages appropriés en utilisant la même sémantique que définie dans la [RFC3280]. Par exemple, si le client a obtenu le certificat dans le contexte d'un serveur de sécurité de la couche transport (TLS, *Transport Layer Security*) il peut confirmer que le certificat est cohérent avec cet usage en incluant la structure usage de clé étendu avec l'identifiant d'objet id-kp-serverAuth.

Si l'extension est absente, ou est présente et affirme l'OID anyExtendedKeyUsage, alors tous les usages spécifiés dans la demande correspondent. Si l'extension est présente et n'affirme pas l'OID anyExtendedKeyUsage, tous les usages dans la demande DOIVENT être présents dans le certificat. L'extension de certificat peut contenir plus d'usages que demandé.

Lorsque le client n'exige pas d'usage de clé étendu particulier, le client peut spécifier une SEQUENCE vide. Ce peut être utilisé pour outrepasser les exigences d'usage de clé étendu imposées dans la politique de validation spécifiée par valPolId.

Les clients SCVP DEVRAIENT prendre en charge extendedKeyUsages, et les serveurs SCVP DOIVENT prendre en charge extendedKeyUsages.

3.2.4.10 specifiedKeyUsages

L'extension usage de clé étendu ([RFC3280], paragraphe 4.2.1.13) définit des objets techniques plus spécifiques, en plus, ou à la place des objets indiqués dans l'extension usage de clé, pour lesquels la clé publique certifiée peut être utilisée. Si le client exige qu'une ou des valeurs particulières apparaissent dans l'extension usage de clé étendu, il peut alors spécifier le ou les usages exigés en utilisant la même sémantique que définie dans la [RFC3280]. Par exemple, si le client a obtenu le certificat dans le contexte d'un serveur de sécurité de la couche transport (TLS, *Transport Layer Security*) il pourrait exiger que le certificat de serveur inclue la structure usage de clé étendu avec l'identifiant d'objet id-kp-serverAuth. Dans ce cas, le client incluerait un élément specifiedKeyUsages dans la demande et affirmerait l'identifiant d'objet id-kp-serverAuth.

Si un ou plusieurs usages spécifiés sont inclus dans la demande, le certificat DOIT contenir l'extension usage de clé étendu , et tous les usages spécifiés dans la demande DOIVENT être présents dans l'extension de certificat. L'extension de certificat peut contenir plus d'usages que spécifiés dans la demande. Les usages de clé spécifiés ne sont pas satisfaits par la présence de l'OID anyExtendedKeyUsage.

Lorsque le client n'exige pas d'usage de clé étendu particulier, le client peut spécifier une SEQUENCE vide. Ce peut être utilisé pour outrepasser des exigences d'usage de clé spécifié imposées dans la politique de validation spécifiée par valPolId.

Les clients SCVP DEVRAIENT prendre en charge specifiedKeyUsages, et les serveurs SCVP DOIVENT prendre en charge specifiedKeyUsages.

3.2.5 responseFlags

L'élément facultatif responseFlags permet au client d'indiquer quelles caractéristiques facultatives il veut que le serveur inclue dans la CVResponse. Si les valeurs par défaut pour tous les fanions sont utilisées, alors l'élément responseFlags NE DOIT PAS être inclus dans la demande.

La syntaxe de l'élément responseFlags est :

```
ResponseFlags ::= SEQUENCE {
  fullRequestInResponse          [0] BOOLEEN DEFAUT FAUX,
  responseValidationPolByRef [1] BOOLEEN DEFAUT VRAI,
```

```
protectResponse          [2] BOOLEEN DEFAUT VRAI,
cachedResponse           [3] BOOLEEN DEFAUT VRAI }
```

Chaque fanion de la réponse est décrit dans les paragraphes qui suivent.

3.2.5.1 fullRequestInResponse

Par défaut, le serveur inclut un hachage de la demande dans les réponses qui ne sont pas en antémémoire pour permettre au client d'identifier la réponse. Si le client veut que le serveur inclue la demande complète dans la réponse non en antémémoire, fullRequestInResponse est réglé à VRAI. La principale raison pour laquelle un client demanderait au serveur d'inclure la demande complète dans la réponse est pour archiver l'échange demande-réponse dans un seul objet. C'est-à-dire, le client veut archiver un seul objet qui inclut à la fois la demande et la réponse.

Les clients et les serveurs SCVP DOIVENT prendre en charge le comportement par défaut. Les clients SCVP PEUVENT prendre en charge de demander et traiter la demande complète. Les serveurs SCVP DEVRAIENT prendre en charge de retourner la demande complète.

3.2.5.2 responseValidationPolByRef

L'élément responseValidationPolByRef contrôle si la réponse inclut juste une référence à la politique ou une référence à la politique plus tous les paramètres par valeur de la politique utilisés pour traiter la demande. La réponse DOIT contenir une référence à la politique de validation. Si le client veut que les paramètres de politique de validation soient inclus aussi par valeur, alors la réponseValidationPolByRef est réglée à FAUX. La principale raison qu'un client aurait de demander au serveur d'inclure la politique de validation par valeur est d'archiver l'échange de demande-réponse dans un seul objet. C'est-à-dire que le client veut archiver la CVResponse et qu'elle inclue chaque aspect de la politique de validation.

Les clients SCVP DOIVENT prendre en charge de demander et traiter la politique de validation par référence, et les serveurs SCVP DOIVENT prendre en charge de retourner la politique de validation par référence. Les clients SCVP PEUVENT prendre en charge de demander et traiter la politique de validation par valeurs. Les serveurs SVCP DEVRAIENT prendre en charge de retourner la politique de validation par valeurs.

3.2.5.3 protectResponse

L'élément protectResponse indique si le client exige que le serveur protège la réponse. Si le client effectue la pleine validation de chemin de certification sur la réponse et si il n'est pas concerné par la source de la réponse, alors le client ne bénéficie pas d'une signature numérique ou d'un MAC sur la réponse. Dans ce cas, le client peut indiquer au serveur qu'il n'est pas nécessaire de protéger le message. Cependant, il est toujours permis au serveur de retourner une réponse protégée.

Les clients SCVP qui prennent en charge la découverte de chemin déléguée (DPD) comme défini dans la [RFC3379] DOIVENT prendre en charge le réglage de cette valeur à FAUX.

Les clients SCVP qui prennent en charge la validation de chemin déléguée (DPV) comme défini dans la [RFC3379] exigent une réponse authentifiée. Sauf si un mécanisme de transport protégé (comme TLS) est utilisé, ces clients DOIVENT toujours régler cette valeur à VRAI ou omettre entièrement l'élément responseFlags, qui exige que le serveur retourne une réponse protégée.

Les serveurs SCVP DOIVENT prendre en charge le retour de réponses protégées, et les serveurs SCVP DEVRAIENT prendre en charge le retour de réponses non protégées. Sur la base de la politique locale, le serveur peut être configuré à retourner des réponses protégées ou non protégées si cette valeur est réglée à FAUX. Si, sur la base de la politique locale, le serveur est incapable de retourner des réponses protégées, alors le serveur DOIT retourner une erreur si cette valeur est réglée à VRAI.

3.2.5.4 cachedResponse

L'élément cachedResponse indique si le client va accepter une réponse d'antémémoire. Pour améliorer les performances et limiter l'exposition des clés de signature, un service SCVP peut être conçu comme mettant les réponses en antémémoire jusqu'à ce que de nouvelles informations de révocation soient attendues. Lorsque cachedResponse est réglé à VRAI, le client va accepter une réponse mise précédemment en antémémoire.

Les clients peuvent insister pour la création d'une réponse fraîche pour se protéger contre une attaque en répétition et s'assurer que les informations sont à jour. Si cachedResponse est FAUX, le client ne va pas accepter une réponse venant de l'antémémoire. Pour s'assurer qu'une réponse est fraîche, le client DOIT aussi inclure le requestNonce comme défini au paragraphe 3.4.

Les serveurs DOIVENT traiter le fanion cachedResponse. Si cachedResponse est FAUX, les serveurs qui ne peuvent pas produire des réponses fraîches DOIVENT répondre avec un message d'erreur. Les serveurs PEUVENT choisir de fournir des réponses fraîches même lorsque cachedResponse est réglé à VRAI.

3.2.6 serverContextInfo

L'élément facultatif serverContextInfo, si il est présent, contient le contexte provenant d'un précédent échange de demande-réponse avec le même serveur SCVP. Il permet au serveur de retourner plus d'un chemin de certification pour le même certificat au client. Par exemple, si un serveur construit un chemin de certification particulier pour un certificat, mais que le client le trouve inacceptable, le client peut alors renvoyer la même interrogation au serveur avec le serverContextInfo provenant de la première réponse, et le serveur va être capable de fournir un chemin de certification différent (si un autre peut être trouvé).

Le contenu des serverContextInfo est opaque au client SCVP. C'est-à-dire que le client sait seulement qu'il a besoin de retourner la valeur fournie par le serveur avec la demande suivante pour obtenir un chemin de certification différent. Noter que l'interrogation suivante doit être identique à la précédente à l'exception de :
- requestNonce,
- serverContextInfo, et
- la signature numérique ou MAC du client sur la demande.

Les clients SCVP PEUVENT prendre en charge serverContextInfo, et les serveurs SCVP DEVRAIENT prendre en charge serverContextInfo.

3.2.7 validationTime

L'élément facultatif validationTime, si il est présent, dit la date et heure relative à laquelle le client SCVP veut que le serveur effectue les vérifications. Si validationTime n'est pas présent, le serveur DOIT effectuer la validation en utilisant la date et heure à laquelle le serveur traite la demande. Si validationTime est présent, elle DOIT être codée comme un GeneralizedTime. La validationTime fournie DOIT être une heure rétrospective car le serveur peut seulement effectuer une vérification de validité en utilisant l'heure courante (par défaut) ou une heure précédente. Un serveur peut ignorer le validationTime fourni dans la demande si l'heure est dans le biais d'horloge de l'heure courante du serveur.

Les informations d'état de révocation sont obtenues par rapport à l'heure de validation. Quand on spécifie une heure de validation autre que l'heure courante, l'heure de validation ne devrait pas nécessairement être identique à l'heure où la clé privée a été utilisée. L'heure de validation spécifiée par le client peut être ajustée pour compenser :
1) le temps pour que l'entité d'extrémité réalise que sa clé privée a été, ou pourrait être, compromise, et/ou
2) le temps pour que l'entité d'extrémité rapporte la compromission de la clé, et/ou
3) le temps pour que l'autorité de révocation traite la demande de révocation venant de l'entité d'extrémité, et/ou
4) le temps pour que l'autorité de révocation mette à jour et distribue les informations d'état de révocation.

Les valeurs de GeneralizedTime DOIVENT être exprimées en temps universel coordonné (UTC, *Universal Coordinated Time*) (qui est aussi appelé heure moyenne de Greenwich et heure Zoulou) et DOIVENT inclure les secondes (c'est-à-dire, les temps sont de format AAAAMMJJHHMMSSZ) même quand le nombre de secondes est zéro. Les valeurs de GeneralizedTime NE DOIVENT PAS inclure de fraction de seconde.

Les informations dans l'élément CertReply correspondant dans la réponse DOIVENT être formatées comme si le serveur avait créé la réponse au moment indiqué dans validationTime. Cependant, si le serveur n'a pas les informations d'historique appropriées, il DOIT retourner une réponse d'erreur.

Les serveurs SCVP DOIVENT appliquer un biais d'horloge à l'heure de validation pour permettre des erreurs mineures de synchronisation. La valeur par défaut est 10 minutes. Si le serveur utilise une valeur autre que celle par défaut, il DOIT inclure la valeur de biais d'horloge dans la réponse de politique de validation.

Les clients SCVP PEUVENT prendre en charge une validationTime autre que de l'heure courante. Les serveurs SCVP DOIVENT prendre en charge l'utilisation de leur heure courante, et DEVRAIENT prendre en charge le réglage par le client de validationTime dans la demande.

3.2.8 intermediateCerts

L'élément facultatif intermediateCerts peut aider le serveur SCVP à créer des chemins de certification valides. L'élément intermediateCerts, quand il est présent, fournit des certificats que le serveur PEUT utiliser quand il fourmit un chemin de certification. Quand il construit des chemins de certification, le serveur PEUT utiliser les certificats dans l'élément intermediateCerts en plus de tous autres certificats auxquels le serveur peut accéder. Quand il est présent, l'élément intermediateCerts DOIT contenir au moins un certificat, et l'élément intermediateCerts DOIT être structuré comme un CertBundle. Les certificats dans l'élément intermediateCerts NE DOIVENT PAS être considérés comme valides par le serveur juste parce que ils sont présents dans cet élément.

Le type CertBundle contient un ou plusieurs certificats. L'ordre des entrées dans le faiscau n'est pas important. CertBundle a la syntaxe suivante :

```
CertBundle ::= TAILLE DE SÉQUENCE (1..MAX) DE Certificate
```

Les clients SCVP DEVRAIENT prendre en charge intermediateCerts, et les serveurs SCVP DOIVENT prendre en charge intermediateCerts.

3.2.9 revInfos

L'élément facultatif revInfos spécifie des informations de révocation comme les CRL, les CRL deltas [RFC3280], et les réponses OCSP [RFC2560] que le serveur SCVP PEUT utiliser quand il valide les chemins de certification. L'objet de l'élément revInfos est de fournir les informations de révocation auxquelles le serveur ne pourrait autrement pas avoir accès, comme une réponse OCSP que le client a reçu avec le certificat. Noter que les informations dans l'élément revInfos ne pourraient pas être utilisées par le serveur. Par exemple, les informations de révocation pourraient être associées à des certificats que le serveur n'utilise pas dans le chemin de certification qu'il construit.

Les clients DEVRAIENT être courtois avec le serveur SCVP en séparant les CRL et les CRL deltas. Cependant, comme les deux partagent une syntaxe commune, les serveurs SCVP DEVRAIENT accepter les CRL deltas même si elles sont identifiées comme des CRL régulières par le client SCVP.

Les CRL, CRL deltas, et réponses OCSP peuvent être fournies comme des informations de révocation. Si nécessaire, des identifiants d'objet supplémentaires pourront être alloués pour des types d'informations de révocation supplémentaires à l'avenir.

L'élément revInfos utilise le type RevocationInfos, qui a la syntaxe suivante :

```
RevocationInfos ::= TAILLE DE SÉQUENCE (1..MAX) DE RevocationInfo
RevocationInfo ::= CHOIX {
  crl               [0] CertificateList,
  delta-crl         [1] CertificateList,
  ocsp              [2] OCSPResponse,
  autre             [3] OtherRevInfo }

OtherRevInfo ::= SEQUENCE {
  riType              IDENTIFIANT D'OBJET,
  riValue             ANY DEFINI PAR riType }
```

3.2.10 producedAt

Le client PEUT permettre au serveur d'utiliser une réponse SCVP mise en antémémoire. Quand il le fait, le client PEUT utiliser l'élément producedAt pour exprimer les exigences sur la fraîcheur de la réponse mise en antémémoire. L'élément producedAt dit la date et l'heure la plus ancienne à laquelle une réponse mise en antémémoire acceptable pourrait avoir été produite. L'élément producedAt représente la date et l'heure en UTC, en utilisant le type GeneralizedTime. La valeur dans l'élément producedAt est indépendante de l'instant de la validation.

La valeur GeneralizedTime DOIT être exprimée en UTC, comme défini au paragraphe 3.2.7.

Les clients SCVP PEUVENT prendre en charge l'utilisation des valeurs de producedAt dans la demande. Les serveurs SCVP PEUVENT prendre en charge les valeurs de producedAt dans la demande. Les serveurs SCVP qui prennent en charge les réponses mises en antémémoire DEVRAIENT prendre en charge la valeur producedAt dans les demandes.

3.2.11 queryExtensions

L'élément facultatif queryExtensions contient des extensions. Si il est présent, chaque extension dans la séquence étend l'interrogation. La présente spécification ne définit aucune extension ; la facilité est fournie pour permettre à de futures spécifications d'étendre SCVP. La syntaxe pour Extensions est importée de la [RFC3280]. L'élément queryExtensions, quand il est présent, DOIT contenir une séquence d'éléments Extension, et chacune des extensions DOIT contenir des éléments extnID, critical, et extnValue. Chacun d'eux est décrit dans les paragraphes suivants.

3.2.11.1 extnID

L'élément extnID est un identifiant pour l'extension. Il contient l'identifiant d'objet qui désigne l'extension.

3.2.11.2 critical

L'élément critical est un BOOLÉEN. Chaque extension est désignée comme critique (avec une valeur de VRAI) ou non critique (avec une valeur de FAUX). Par défaut, l'extension est non critique. Un serveur SCVP DOIT rejeter l'interrogation si il rencontre une extension critique qu'il ne reconnaît pas ; cependant, une extension non critique PEUT être ignorée si elle n'est pas reconnue, mais DOIT être traitée si elle est reconnue.

3.2.11.3 extnValue

L'élément extnValue contient une CHAINE D'OCTETS. Dans la CHAINE D'OCTETS est la valeur de l'extension. Un type ASN.1 est spécifié pour chaque extension, identifié par l'identifiant d'objet associé extnID.

3.3 requestorRef

L'élément facultatif requestorRef contient une liste de noms identifiant les serveurs SCVP, et il est destiné à être utilisé dans les environnements où un relais SCVP est employé. Bien que requestorRef soit codé comme SEQUENCE, aucun ordre n'est impliqué. L'élément requestorRef est utilisé pour détecter les boucles dans certaines configurations. La valeur et l'utilisation de requestorRef sont décrites à la Section 7.

Les clients SCVP conformes PEUVENT prendre en charge la spécification de la valeur de requestorRef. Les mises en œuvre conformes de serveur SCVP DOIVENT traiter la valeur de requestorRef si elle est présente. Si le client SCVP inclut une valeur de requestorRef dans la demande, le serveur SCVP DOIT alors retourner la même valeur dans une réponse qui n'est pas mise en antémémoire. Le serveur SCVP PEUT omettre la valeur de requestorRef des réponses SCVP mises en antémémoire.

L'élément requestorRef DOIT être une séquence de GeneralName. Aucune disposition n'est prise pour assurer l'unicité des valeurs de GeneralName de requestorRef.

3.4 requestNonce

L'élément facultatif requestNonce contient un identifiant de demande généré par le client SCVP. Si le client inclut une valeur de requestNonce dans la demande, il exprime la préférence que le serveur SCVP DEVRAIT retourner une réponse non mise en antémémoire. Si le serveur retourne une réponse non mise en antémémoire, il DOIT inclure la valeur de requestNonce provenant de la demande dans la réponse comme élément respNonce ; cependant, le serveur PEUT retourner une réponse mise en antémémoire qui NE DOIT PAS avoir un respNonce.

Les clients SCVP qui insistent pour la création d'une réponse fraîche (par exemple, pour se protéger contre une attaque en répétition ou s'assurer que les informations sont à jour) DOIT prendre en charge requestNonce. Les mises en œuvre

conformes de serveur SCVP DOIVENT traiter la valeur de requestNonce si elle est présente.

Si le client inclut un requestNonce et règle aussi le fanion cachedResponse à FAUX comme décrit au paragraphe 3.2.5.4, le client indique que le serveur SCVP DOIT retourner une réponse non mise en antémémoire incluant le respNonce ou une réponse d'erreur. Le client DEVRAIT inclure un élément requestNonce dans chaque demande pour empêcher un attaquant d'agir en interposé en répétant des vieilles réponses provenant du serveur. La valeur de requestNonce DEVRAIT changer à chaque demande envoyée par le client.

Le client NE DOIT PAS régler le fanion cachedResponse à FAUX sans inclure aussi un requestNonce. Un serveur qui reçoit une telle demande DEVRAIT retourner une réponse d'erreur invalidRequest.

L'élément requestNonce, si il est présent, DOIT être une CHAINE D'OCTETS qui a été générée exclusivement pour cette demande.

3.5 requestorName

L'élément facultatif requestorName est utilisé par le client pour inclure un identifiant dans la demande. Le client PEUT inclure cette information pour que le serveur DPV la copie dans la réponse.

Les clients conformes à SCVP PEUVENT prendre en charge la spécification de cet élément dans les demandes. Les serveurs SCVP DOIVENT être capables de traiter les demandes qui incluent cet élément.

3.6 responderName

L'élément facultatif responderName est utilisé par le client pour indiquer l'identité du serveur SCVP dont le client s'attend qu'il signe la réponse SCVP si la réponse est signée numériquement. L'élément responderName DEVRAIT seulement être inclus si :
1. la demande est non protégée ou signée numériquement (c'est-à-dire, n'est pas protégée en utilisant un MAC) et
2. l'élément responseFlags est soit absent, soit présent avec protectResponse réglé à VRAI.

Les clients conformes à SCVP PEUVENT prendre en charge la spécification de cet élément dans les demandes. Les serveurs SCVP DOIVENT être capables de traiter les demandes qui incluent cet élément. Les serveurs SCVP qui tiennent une seule clé privée pour signer les réponses SCVP ou qui ne sont pas capables de retourner des réponses signées numériquement PEUVENT ignorer la valeur de cet élément. Les serveurs SCVP qui tiennent plus d'une clé privée pour signer les réponses SCVP DEVRAIENT soit (a) signer numériquement la réponse en utilisant une clé privée correspondant à un certificat qui inclut le nom spécifié dans responderName dans le champ subject ou l'extension subjectAltName, soit (b) retourner une erreur indiquant que le serveur ne possède pas de certificat affirmant le nom spécifié.

3.7 requestExtensions

L'élément FACULTATIF requestExtensions contient des extensions. Si il est présent, chaque extension dans la séquence étend la demande. La présente spécification ne définit aucune extension ; la facilité est fournie pour permettre que de futures spécifications étendent SCVP. La syntaxe de Extensions est importée de la [RFC3280]. L'élément requestExtensions, quand il est présent, DOIT contenir une séquence d'éléments Extension, et chaque extension DOIT contenir les éléments extnID, critical, et extnValue. Chacun de ces éléments est décrit dans les paragraphes suivants.

3.7.1 extnID

L'élément extnID est un identifiant de l'extension. Il contient l'identifiant d'objet qui désigne l'extension.

3.7.2 critical

L'élément critical est un BOOLÉEN. Chaque extension est désignée comme critique (avec une valeur de VRAI) ou non critique (avec une valeur de FAUX). Par défaut, l'extension est non critique. Un serveur SCVP DOIT rejeter l'interrogation si il rencontre une extension critique qu'il ne reconnaît pas. Une extension non critique PEUT être ignorée si elle n'est pas reconnue, mais DOIT être traitée si elle est reconnue.

3.7.3 extnValue

L'élément extnValue contient une CHAINE D'OCTETS. La valeur de l'extension est dans la CHAINE D'OCTETS. Un type ASN.1 est spécifié pour chaque extension, identifié par l'identifiant d'objet associé extnID.

3.8 signatureAlg

L'élément signatureAlg contient un AlgorithmIdentifier indiquant quel algorithme le serveur devrait utiliser pour signer le message de réponse. L'élément signatureAlg DEVRAIT seulement être inclus si :
1. la demande est non protégée ou signée numériquement (c'est-à-dire, n'est pas protégée en utilisant un MAC) et
2. l'élément responseFlags est soit absent, soit présent avec protectResponse réglé à VRAI.

Si il est inclus, l'élément signatureAlg DEVRAIT spécifier un des algorithmes de signature spécifiés dans l'élément signatureGeneration du message de réponse de politique de validation du serveur.

Les serveurs SCVP DOIVENT être capables de traiter les demandes qui incluent cet élément. Si le serveur retourne une réponse signée numériquement à ce message, alors :
1. Si l'élément signatureAlg est présent et spécifie un algorithme qui est inclus dans l'élément signatureGeneration du message de réponse de politique de validation du serveur, le serveur DOIT signer la réponse en utilisant l'algorithme de signature spécifié dans signatureAlg.
2. Autrement, si l'élément signatureAlg est absent ou est présent mais spécifie un algorithme qui n'est pas pris en charge par le serveur, le serveur DOIT signer la réponse en utilisant l'algorithme de signature par défaut du serveur comme spécifié dans l'élément signatureGeneration du message de réponse de politique de validation du serveur.

3.9 hashAlg

L'élément hashAlg contient un identifiant d'objet qui indique quel algorithme de hachage le serveur devrait utiliser pour calculer la valeur de hachage pour l'élément requestHash dans la réponse. Les clients SCVP NE DEVRAIENT PAS inclure cet élément si fullRequestInResponse est réglé à VRAI. Si il est inclus, l'élément hashAlg DEVRAIT spécifier un des algorithmes de hachage spécifiés dans l'élément hashAlgorithms du message de réponse de politique de validation du serveur.

Les serveurs SCVP DOIVENT être capables de traiter les demandes qui incluent cet élément. Si le serveur retourne une réponse à ce message qui inclut un requestHash, alors :
1. Si l'élément hashAlg est présent et spécifie un algorithme inclus dans l'élément hashAlgorithms du message de réponse de politique de validation du serveur, le serveur DOIT utiliser l'algorithme spécifié dans hashAlg pour calculer le requestHash.
2. Autrement, si l'élément hashAlg est absent ou est présent mais spécifie un algorithme qui n'est pas pris en charge par le serveur, le serveur DOIT calculer le requestHash en utilisant l'algorithme de hachage par défaut du serveur comme spécifié dans l'élément hashAlgorithms du message de réponse de politique de validation du serveur.

3.10 requestorText

Les clients SCVP PEUVENT utiliser l'élément requestorText pour fournir du texte à inclure dans la réponse correspondante. Par exemple, ce champ peut décrire la nature ou la raison de la demande.

Les mises en œuvre conformes de client SCVP PEUVENT prendre en charge l'inclusion de cet élément dans les demandes. Les mises en œuvre conformes de serveur SCVP DOIVENT accepter les demandes qui incluent cet élément. Quand elles génèrent des réponses non mises en antémémoire, les mises en œuvre conformes de serveur SCVP DOIVENT copier le contenu de cet élément dans l'élément requestorText dans la réponse correspondante (voir le paragraphe 4.13).

3.11 Authentification de demande SCVP

La politique de validation que le serveur utilise quand il authentifie les demandes relève de la politique locale. Quand ils authentifient des demandes SCVP protégées, les serveurs SCVP DEVRAIENT utiliser l'algorithme de validation défini à la Section 6 de la [RFC3280].

Si le certificat utilisé pour valider une demande de validation SignedData inclut l'extension d'usage de clé ([RFC3280], paragraphe 4.2.1.3) il DOIT avoir le bit Signature numérique établi, le bit Non répudiation établi, ou les deux bits établis.

Si le certificat utilisé pour valider une demande de validation AuthenticatedData inclut l'extension d'usage de clé, il DOIT avoir le bit Accord de clé établi.

Si le certificat utilisé sur une demande de validation contient l'extension d'usage de clé étendu ([RFC3280], paragraphe 4.2.1.13) le serveur DEVRA vérifier qu'il contient l'OID du client SCVP, l'OID anyExtendedKeyUsage, ou un autre OID acceptable pour le serveur. L'OID de client SCVP est défini comme suit :

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

IDENTIFIANT D'OBJET id-kp-scvpClient ::= { id-kp 16 }
```

Si une demande protégée ne réussit pas à satisfaire la politique de validation du serveur, elle DOIT être traitée comme une demande non authentifiée.

4. Réponse de validation

Une réponse de serveur SCVP au client DOIT être un seul élément CVResponse. Quand une CVResponse est encapsulée dans une partie de corps MIME, application/scvp-cv-response DOIT être utilisé.

Une réponse SCVP peut avoir un certain nombre de formes :

1. Une réponse de succès à une demande qui a protectResponse réglé à FAUX. Ces réponses NE DEVRAIENT PAS être protégées par le serveur.

2. Le serveur DOIT protéger toutes les autres réponses de succès. Si le serveur n'est pas capable de retourner une réponse de succès protégée à cause de la politique locale, il DOIT alors retourner une réponse d'erreur.

3. Une réponse d'erreur à une demande faite sur un transport protégé comme TLS. Ces réponses NE DEVRAIENT PAS être protégées par le serveur.

4. Une réponse d'erreur à une demande qui a protectResponse réglé à FAUX. Ces réponses NE DEVRAIENT PAS être protégées par le serveur.

5. Une réponse d'erreur à une demande authentifiée. Le serveur DEVRAIT protéger ces réponses.

6. Une réponse d'erreur à une demande AuthenticatedData où le MAC est valide. Le serveur DOIT protéger ces réponses.

7. Toutes les autres réponses d'erreur NE DOIVENT PAS être protégées par le serveur.

Des réponses de succès sont faites quand le serveur s'est complètement conformé à la demande. C'est-à-dire que le serveur a été capable de tenter de construire un chemin de certification en utilisant la politique de validation référencée ou en a fourni une, et qu'il a été capable de se conformer à tous les paramètres demandés. Si le serveur est incapable d'effectuer les validations en utilisant la politique de validation requise ou si la demande contient une option non prise en charge, le serveur DOIT alors retourner une réponse d'erreur.

Pour les demandes et réponses protégées, les serveurs SCVP DOIVENT prendre en charge SignedData et DEVRAIENT prendre en charge AuthenticatedData. Les types utilisés sont une affaire de politique locale. Quand une réponse protégée est requise, les serveurs SCVP DOIVENT utiliser SignedData ou AuthenticatedData, même si la transaction est effectuée en utilisant un transport protégé (par exemple, TLS).

Si le serveur fait une réponse protégée à une demande protégée, le serveur DOIT alors utiliser le même mécanisme de protection (SignedData ou AuthenticatedData) que dans la demande.

On donne ci-dessous une vue d'ensemble de la structure utilisée pour une réponse non protégée. De nombreux détails ne sont pas montrés mais la façon dont SCVP utilise la CMS est clairement illustrée.

```
ContentInfo {
```

```
  contentType     id-ct-scvp-certValResponse,          -- (1.2.840.113549.1.9.16.1.11)
  content         CVResponse }
```

La réponse protégée consiste en une CVResponse encapsulée soit dans des SignedData, soit dans des AuthenticatedData, qui sont à leur tour encapsulées dans un ContentInfo. C'est-à-dire, le champ EncapsulatedContentInfo de SignedData ou AuthenticatedData consiste en un champ eContentType avec une valeur de id-ct-scvp-certValResponse et un champ eContent qui contient une CVResponse codée en DER.

Le serveur SCVP DOIT inclure son propre certificat dans le champ certificates au sein des SignedData. D'autres certificats PEUVENT aussi être inclus.

Le serveur SCVP PEUT aussi fournir une ou plusieurs CRL dans le champ crls au sein de SignedData. Le champ signerInfos de SignedData DOIT inclure exactement un SignerInfo. SignedData NE DOIT PAS inclure de champ unsignedAttrs.

Le champ signedAttrs au sein de SignerInfo DOIT inclure les attributs content-type et message-digest définis dans la [RFC3852], et il DEVRAIT inclure l'attribut signing-certificate comme défini dans la [RFC2634]. Dans l'attribut signing-certificate, le premier certificat identifié dans la séquence d'identifiants de certificat DOIT être le certificat du serveur SCVP. L'inclusion d'autres identifiants de certificat dans l'attribut signing-certificate est FACULTATIVE. L'inclusion de politiques dans signing-certificate est FACULTATIVE.

Le champ recipientInfos de AuthenticatedData DOIT inclure exactement un RecipientInfo, qui contient des information pour le client qui a envoyé la demande. AuthenticatedData NE DOIT PAS inclure de champ unauthAttrs.

L'élément CVResponse contient la réponse du serveur. CVResponse DOIT contenir les éléments cvResponseVersion, serverConfigurationID, producedAt, et responseStatus. La CVResponse PEUT aussi contenir les éléments respValidationPolicy, requestRef, requestorRef, requestorName, replyObjects, respNonce, serverContextInfo, et cvResponseExtensions. L'élément replyObjects DOIT contenir exactement un élément CertReply pour chaque certificat demandé. L'élément requestorRef DOIT être inclus si la demande incluait un élément requestorRef et si une réponse non mise en antémémoire est fournie. L'élément respNonce DOIT être inclus si la demande incluait un élément requestNonce et si une réponse non mise en antémémoire est fournie.

CVResponse DOIT avoir la syntaxe suivante :

```
CVResponse ::= SEQUENCE {
  cvResponseVersion       ENTIER,
  serverConfigurationID    ENTIER,
  producedAt              GeneralizedTime,
  responseStatus          ResponseStatus,
  respValidationPolicy    [0] RespValidationPolicy FACULTATIF,
  requestRef              [1] RequestReference FACULTATIF,
  requestorRef            [2] GeneralNames FACULTATIF,
  requestorName           [3] GeneralNames FACULTATIF,
  replyObjects            [4] ReplyObjects FACULTATIF,
  respNonce               [5] CHAINE D'OCTETS FACULTATIF,
  serverContextInfo       [6] CHAINE D'OCTETS FACULTATIF,
  cvResponseExtensions [7] Extensions FACULTATIF,
  requestorText           [8] UTF8String (TAILLE (1..256)) FACULTATIF }
```

Les serveurs SCVP conformes PEUVENT être capables de construire une CVResponse qui inclut les éléments serverContextInfo ou cvResponseExtensions. Les serveurs SCVP conformes DOIVENT être capables de construire une CVResponse avec tous les éléments facultatifs restants. Les clients conformes à SCVP DOIVENT être capables de traiter une CVResponse avec les éléments facultatifs suivants : respValidationPolicy, requestRef, requestorName, replyObjects, et respNonce.

Les clients conformes SCVP qui sont capables d'inclure requestorRef dans une demande DOIVENT être capables de traiter une CVResponse qui inclut l'élément requestorRef. Les clients conformes SCVP DOIVENT être capables de traiter une CVResponse qui inclut les éléments serverContextInfo ou cvResponseExtensions. Les clients conformes DOIVENT être capables de déterminer si des extensions critiques sont présentes dans l'élément cvResponseExtensions.

4.1 cvResponseVersion

La syntaxe et la sémantique de cvResponseVersion sont les mêmes que celles de cvRequestVersion comme décrit au paragraphe 3.1. La cvResponseVersion DOIT correspondre à la cvRequestVersion de la demande. Si le serveur ne peut pas générer une réponse avec un numéro de version correspondant, le serveur DOIT alors retourner une réponse d'erreur qui indique le plus haut numéro de version que le serveur supporte.

4.2 serverConfigurationID

L'élément serverconfigurationID représente la version de la configuration de serveur SCVP quand il traite la demande. Voir les détails au paragraphe 6.4.

4.3 producedAt

L'élément producedAt dit la date et l'heure à laquelle le serveur SCVP a généré la réponse. L'élément producedAt DOIT être exprimé en UTC, et il DOIT être interprété comme défini au paragraphe 3.2.7. Cette valeur est indépendante de l'heure de validation.

4.4 responseStatus

L'élément responseStatus donne des informations d'état au client SCVP sur sa demande. L'élément responseStatus a un code d'état numérique et une chaîne facultative qui est une séquence de caractères de l'ensemble de caractères ISO/CEI 10646-1 codé avec le format de transformation UTF-8 défini dans la [RFC3629].

La chaîne PEUT être utilisée pour transmettre des informations d'état. Le client PEUT choisir d'afficher la chaîne à un utilisateur humain. Cependant, parce qu'il n'y a souvent pas de moyen de savoir les langages compris par un utilisateur humain, la chaîne peut être de peu ou pas d'aide du tout.

L'élément responseStatus utilise le type ResponseStatus, qui a la syntaxe suivante :

```
ResponseStatus ::= SEQUENCE {
  statusCode            CVStatusCode DEFAUT  okay,
  errorMessage          UTF8String FACULTATIF }

CVStatusCode ::= ENUMERATED {
  okay                                  (0),
  skipUnrecognizedItems                 (1),
  tooBusy                              (10),
  invalidRequest                       (11),
  internalError                        (12),
  badStructure                         (20),
  unsupportedVersion                   (21),
  abortUnrecognizedItems               (22),
  unrecognizedSigKey                   (23),
  badSignatureOrMAC                    (24),
  unableToDecode                       (25),
  notAuthorized                        (26),
  unsupportedChecks                    (27),
  unsupportedWantBacks                 (28),
  unsupportedSignatureOrMAC            (29),
  invalidSignatureOrMAC                (30),
  protectedResponseUnsupported         (31),
  unrecognizedResponderName            (32),
  relayingLoop                         (40),
  unrecognizedValPol                   (50),
  unrecognizedValAlg                   (51),
  fullRequestInResponseUnsupported     (52),
  fullPolResponseUnsupported           (53),
  inhibitPolicyMappingUnsupported      (54),
```

```
requireExplicitPolicyUnsupported    (55),
inhibitAnyPolicyUnsupported         (56),
validationTimeUnsupported           (57),
unrecognizedCritQueryExt            (63),
unrecognizedCritRequestExt          (64) }
```

Les valeurs de CVStatusCode ont la signification suivante :

0 La demande a été entièrement traitée.
1 La demande incluait des extensions non critiques non reconnues ; cependant, le traitement a été capable de continuer en les ignorant.
10 Trop occupé; réessayer plus tard.
11 Le serveur a été capable de décoder la demande, mais elle pose d'autres problèmes.
12 Une erreur interne au serveur s'est produite.
20 La structure de la demande est erronnée.
21 La version de la demande n'est pas prise en charge par ce serveur.
22 La demande incluait des éléments non reconnus, et le serveur n'a pas été capable de continuer le traitement.
23 Le serveur n'a pas pu valider la clé utilisée pour protéger la demande.
24 La signature ou le code d'authentification de message ne correspond pas au corps de la demande.
25 Le codage n'a pas été compris.
26 La demande n'a pas été autorisée.
27 La demande incluait des éléments de vérification non pris en charge, et le serveur n'a pas été capable de continuer le traitement.
28 La demande incluait des éléments wantBack non pris en charge, et le serveur n'a pas été capable de continuer le traitement.
29 Le serveur ne prend pas en charge l'algorithme de signature ou le code d'authentification de message utilisé par le client pour protéger la demande.
30 Le serveur n'a pas pu valider la signature ou le code d'authentification de message du client sur la demande.
31 Le serveur n'a pas pu générer une réponse protégée comme demandé par le client.
32 Le serveur n'a pas de certificat correspondant au nom de répondant demandé.
40 La demande a été précédemment relayée par le même serveur.
50 La demande contenait une référence de politique de validation non reconnue.
51 La demande contenait un OID d'algorithme de validation non reconnu.
52 Le serveur ne prend pas en charge le retour de la demande complète dans la réponse.
53 Le serveur ne prend pas en charge le retour de la politique de validation par valeur complète dans la réponse.
54 Le serveur ne prend pas en charge la valeur demandée pour inhiber la transposition de politique.
55 Le serveur ne prend pas en charge la valeur demandée pour exiger une politique explicite.
56 Le serveur ne prend pas en charge la valeur demandée pour inhiber anyPolicy.
57 Le serveur valide seulement les demandes en utilisant l'heure courante.
63 L'élément query dans la demande contient une extension critique dont l'OID n'est pas reconnu.
64 La demande contient une extension de demande critique dont l'OID n'est pas reconnu.

Les codes d'état 0 à 9 sont réservés pour les codes qui indiquent que la demande a été traitée par le serveur et donc DOIVENT être envoyés dans une réponse de succès. Les codes d'état 10 et au dessus indiquent une erreur et DOIVENT donc être envoyés dans une réponse d'erreur.

4.5 respValidationPolicy

L'élément respValidationPolicy contient soit une référence à la pleine politique de validation ou à la pleine politique par valeur utilisée par le serveur pour valider la demande. Il DOIT être présent dans les réponses de succès et NE DOIT PAS être présent dans les réponses d'erreur. Le choix entre retourner la politique par référence ou par valeur est contrôlé par l'élément responseValidationPolByRef dans la demande. La politique de validation qui en résulte est l'union de :

1. les valeurs provenant de la demande ;
2. pour les valeurs qui ne sont pas explicitement incluses dans la demande, les valeurs provenant de la politique de validation spécifiées par référence dans la demande.

La syntaxe de RespValidationPolicy est :

```
RespValidationPolicy ::= ValidationPolicy
```

L'élément validationPolicy est défini au paragraphe 3.2.4. Quand responseValidationPolByRef est réglé à FAUX dans la demande, tous les éléments dans l'élément validationPolicy DOIVENT être remplis. Quand responseValidationPolByRef est réglé à VRAI, les éléments FACULTATIFS dans l'élément validationPolicy ont seulement besoin d'être remplis si la valeur dans la demande diffère de la valeur provenant de la politique de validation référencée.

Les clients conformes à SCVP DOIVENT être capables de traiter la politique de validation par référence. Les clients SCVP PEUVENT être capables de traiter les éléments facultatifs dans la politique de validation.

Les mises en œuvre conformes de serveur SCVP DOIVENT être capables d'affirmer la politique par référence, et DOIVENT être capables d'inclure les éléments facultatifs.

4.6 requestRef

L'élément requestRef permet au client SCVP d'identifier la demande qui correspond à cette réponse du serveur. Il associe la réponse à une demande particulière en utilisant un hachage de la demande ou une copie de la CVRequest provenant de la demande.

L'élément requestRef ne fournit pas d'authentification, mais permet au client de déterminer que la demande n'a pas été modifiée par malvaillance.

L'élément requestRef permet au client d'associer une réponse à une demande. L'élément requestNonce donne un mécanisme de remplacement pour faire correspondre les demandes et les réponses. Quand fullRequest est utilisé, la réponse donne une seule structure de données qui convient pour l'archivage de la transaction.

L'élément requestRef utilise le type RequestReference, qui a la syntaxe suivante :

```
RequestReference ::= CHOIX {
  requestHash      [0] HashValue,              -- hachage de CVRequest
  fullRequest      [1] CVRequest }
```

Les clients SCVP DOIVENT prendre en charge requestHash, et ils PEUVENT prendre en charge fullRequest. Les serveurs SCVP DOIVENT prendre en charge requestHash, et ils DEVRAIENT prendre en charge fullRequest.

4.6.1 requestHash

L'élément requestHash est le hachage de CVRequest. La fonction de hachage unidirectionnel utilisée pour calculer le hachage de la CVRequest est comme spécifié au paragraphe 3.9. L'élément requestHash sert à deux fins. D'abord, il permet à un client de déterminer que la demande n'a pas été modifiée de façon malveillante. Ensuite, il permet au client d'associer une réponse à une demande quand on utilise des protocoles sans connexion. L'élément requestNonce donne un mécanisme de remplacement pour faire correspondre les demandes et les réponses.

L'élément requestHash utilise le type HashValue, qui a la syntaxe suivante :

```
HashValue ::= SEQUENCE {
  algorithm      AlgorithmIdentifier DEFAUT { algorithm sha-1 },
  value          CHAINE D'OCTETS }

IDENTIFIANT D'OBJET sha-1 ::= { iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) 26 }
```

L'identifiant d'algorithme pour SHA-1 est importé de la [RFC3279]. Il est répété ici pour des raisons pratiques.

4.6.2 fullRequest

Comme requestHash, la solution de remplacement fullRequest permet à un client de déterminer que la demande n'a pas été modifiée de façon malveillante. Elle donne aussi une seule structure de données qui convient pour l'archivage de la transaction.

L'élément fullRequest est de type CVRequest. La syntaxe et la sémantique du type CVRequest sont décrites à la Section 3.

4.7 requestorRef

L'élément facultatif requestorRef est utilisé par le client pour identifier le demandeur d'origine dans les cas où un relais SCVP est utilisé. La valeur n'a de signification que locale pour le client. Si le client SCVP inclut une valeur de requestorRef dans la demande, le serveur SCVP DOIT retourner la même valeur si le serveur génère une réponse non mise en antémémoire.

4.8 requestorName

L'élément facultatif requestorName est utilisé par le serveur pour retourner une ou plusieurs identités associées au client dans la réponse.

Le serveur SCVP PEUT choisir d'inclure un des éléments suivants ou tous :
(1) l'identité affirmée par le client dans l'élément requestorName de la demande,
(2) une identitié authentifiée pour le client à partir d'un certificat ou autre accréditif utilisé pour authentifier la demande, ou
(3) un identifiant de client provenant d'un mécanisme hors bande.

Autrement, le serveur SCVP PEUT omettre cet élément.

Dans le cas de réponses non mises en antémémoire à des demandes authentifiées, le serveur SCVP DEVRAIT retourner un nom de demandeur.

Les serveurs SCVP qui prennent en charge les demandes authentifiées DEVRAIENT prendre en charge cet élément.

Les clients SCVP DOIVENT être capables de traiter les réponses qui incluent cet élément, bien que la valeur de l'élément ne puisse en aucune façon impacter le traitement.

4.9 replyObjects

L'élément replyObjects retourne les objets demandés au client SCVP, chacun d'eux disant au client quelque chose sur un seul certificat de la demande. L'élément replyObjects DOIT être présent dans la réponse, sauf si la réponse rapporte une erreur. L'élément CertReply DOIT contenir les éléments cert, replyStatus, replyValTime, replyChecks, et replyWantBacks, et l'élément CertReply PEUT contenir les éléments validationErrors, nextUpdate, et certReplyExtensions.

Une réponse de succès DOIT contenir une CertReply pour chaque certificat spécifié dans l'élément queriedCerts de la demande. L'ordre est important. La première CertReply dans la séquence DOIT correspondre au premier certificat de la demande, la seconde CertReply dans la séquence DOIT correspondre au second certificat de la demande, et ainsi de suite.

L'élément checks dans la demande détermine le contenu de l'élément replyChecks dans la réponse. L'élément wantBack dans la demande détermine le contenu de l'élément replyWantBacks dans la réponse. Les éléments queryExtensions dans la demande contrôlent l'absence ou la présence et le contenu de l'élément certReplyExtensions dans la réponse.

L'élément replyObjects utilise le type ReplyObjects, qui a la syntaxe suivante :

```
ReplyObjects ::= TAILLE DE SÉQUENCE (1..MAX) DE CertReply
CertReply ::= SEQUENCE {
  cert                CertReference,
  replyStatus         ReplyStatus DEFAUT success,
  replyValTime        GeneralizedTime,
  replyChecks         ReplyChecks,
  replyWantBacks      ReplyWantBacks,
  validationErrors    [0] TAILLE DE SÉQUENCE (1..MAX) DE IDENTIFIANT D'OBJET FACULTATIF,
  nextUpdate          [1] GeneralizedTime FACULTATIF,
  certReplyExtensions [2] Extensions FACULTATIF }
```

4.9.1 cert

L'élément cert contient le certificat ou une référence au certificat sur lequel le client demande des informations. Si le certificat a été spécifié par référence dans la demande, la demande incluait soit le id-swb-pkc-cert, soit le id-swb-aa-cert

wantBack, et si le serveur a été capable d'obtenir le certificat référencé, alors cet élément DOIT inclure le certificat. Autrement, cet élément DOIT inclure la même valeur qu'utilisée dans l'élément queriedCerts de la demande.

CertReference a la syntaxe suivante :

```
CertReference ::= CHOIX {
  pkc             PKCReference,
  ac              ACReference }
```

4.9.2 replyStatus

L'élément replyStatus donne des informations d'état au client sur la demande pour le certificat spécifié. Noter que l'élément responseStatus est différent de l'élément replyStatus. L'élément responseStatus est l'état de la demande entière, tandis que l'élément replyStatus est l'état de l'élément d'interrogation individuel.

L'élément replyStatus utilise le type ReplyStatus, qui a la syntaxe suivante :

```
ReplyStatus ::= ENUMERATED {
  success                 (0),
  malformedPKC            (1),
  malformedAC             (2),
  unavailableValidationTime (3),
  referenceCertHashFail   (4),
  certPathConstructFail   (5),
  certPathNotValid        (6),
  certPathNotValidNow     (7),
  wantBackUnsatisfied     (8) }
```

La signification des diverses valeurs de ReplyStatus est :

0 Succès : toutes les vérifications ont été effectuées avec succès.
1 Échec : le certificat de clé publique est mal formé.
2 Échec : le certificat d'attribut est mal formé.
3 Échec : les données d'historique de l'heure de validation demandée ne sont pas disponibles.
4 Échec : le serveur n'a pas pu localiser le certificat de référence ou le certificat référencé ne correspond pas à la valeur de hachage fournie.
5 Échec : aucun chemin de certification n'a pu être construit.
6 Échec : le chemin de certification construit n'est pas valide par rapport à la politique de validation.
7 Échec : le chemin de certification construit n'est pas valide par rapport à la politique de validation, mais une interrogation ultérieure peut réussir.
8 Échec : toutes les vérifications ont été effectuées avec succès; cependant, un ou plusieurs des wantBacks pourraient n'être pas satisfaits.

Les codes 1 et 2 sont utilisés pour dire au client que la demande est correctement formée, mais que le certificat en question ne l'est pas. Ceci est particulièrement utile aux clients qui n'analysent pas les certificats.

Le code 7 est utilisé pour dire au client qu'un chemin de certification valide a été trouvé à l'exception qu'un certificat dans le chemin est en garde, les informations de révocation courantes sont indisponibles, ou l'heure de validation précède l'heure notBefore dans un ou plusieurs des certificats dans le chemin.

Pour les codes 1, 2, 3, et 4, les éléments replyChecks et replyWantBacks ne sont pas remplis (c'est-à-dire, ils DOIVENT être une séquence vide). Pour les codes 5, 6, 7, et 8, les replyChecks DOIVENT inclure une entrée correspondant à chaque vérification dans la demande ; l'élément replyWantBacks n'est pas rempli.

4.9.3 replyValTime

L'élément replyValTime dit l'heure à laquelle les informations dans le CertReply étaient correctes. L'élément replyValTime représente la date et l'heure en UTC, en utilisant le type GeneralizedTime. Les règles de codage pour GeneralizedTime du paragraphe 3.2.7 DOIVENT être utilisées.

Dans la demande, l'élément facultatif validationTime dit la date et heure relative à laquelle le client SCVP veut que le serveur effectue les vérifications. Si validationTime n'est pas présent, le serveur DOIT répondre comme si le client avait fourni la date et heure à laquelle le serveur traite la demande.

Les informations dans l'élément CertReply DOIVENT être formatées comme si le serveur avait créé cette portion de la réponse à l'instant indiqué dans l'élément validationTime de l'interrogation. Cependant, si le serveur n'a pas les informations d'historique appropriées, le serveur PEUT retourner une erreur ou retourner les informations ultérieurement.

4.9.4 replyChecks

L'élément replyChecks contient les réponses à l'élément checks dans l'interrogation. L'élément replyChecks inclut l'identifiant d'objet (OID) provenant de l'interrogation et un entier. La valeur de l'entier indique si la vérification demandée a réussi. Les OID dans l'élément checks de l'interrogation sont utilisés pour identifier les valeurs correspondantes de replyChecks. Chaque OID spécifié dans l'élément checks de la demande DOIT correspondre à un OID dans l'élément replyChecks de la réponse. Dans le cas d'une réponse d'erreur, le serveur PEUT inclure des checks supplémentaires dans la réponse pour mieux expliquer l'erreur. Les clients DOIVENT ignorer tout ReplyCheck non reonnu inclus dans la réponse.

L'élément replyChecks utilise le type ReplyChecks, qui a la syntaxe suivante :

```
ReplyChecks ::= SEQUENCE DE ReplyCheck
ReplyCheck ::= SEQUENCE {
  check          IDENTIFIANT D'OBJET,
  status         ENTIER DEFAUT 0 }
```

La valeur d'état pour la construction de la clé publique du chemin de certification à une racine de confiance, { id-stc 1 }, peut être une des suivantes :
0 : Un chemin est construit
1 : Un chemin n'a pas pu être construit

La valeur d'état pour la construction de la clé publique du chemin de certification à une racine de confiance avec un traitement de simple validation, { id-stc 2 }, peut être une des suivantes :
0 : Valide
1 : Non valide

La valeur d'état pour la construction de la clé publique du chemin de certification à une racine de confiance avec vérification d'état complète, { id-stc 3 }, peut être une des suivantes :
0 : Valide
1 : Non valide
2 : Révocation hors ligne
3 : Révocation indisponible
4 : Pas de source connue pour les informations de révocation

Révocation hors ligne signifie que le serveur ou le point de distribution des informations de révocation a été connecté avec succès sans erreur du réseau mais soit aucune donnée n'a été retournée, soit si des données ont été retournées, elles étaient périmées. Révocation indisponible signifie qu'une erreur du réseau a été retournée quand une tentative a été faite de joindre le serveur ou le point de distribution. Pas de source connue pour les informations de révocation signifie que le serveur a été capable de construire un chemin de certification valide mais n'a pas pu localiser une source pour les informations de révocation pour un ou plusieurs certificats dans le chemin.

La valeur d'état pour la construction d'un chemin de certification produit par une AC à une racine de confiance, { id-stc 4 }, peut être une des suivantes :
0 : Un chemin est construit
1 : Un chemin n'a pas pu être construit

La valeur d'état pour la construction d'un chemin de certification produit par une AC à une racine de confiance avec un traitement simple de validation, { id-stc 5 }, peut être une des suivantes :
0 : Valide
1 : Non valide

La valeur d'état pour la construction d'un chemin de certification produit par une AC à une racine de confiance avec une vérification complète d'état, { id-stc 6 }, peut être une des suivantes :
0 : Valide
1 : Non valide
2 : Révocation hors ligne
3 : Révocation indisponible
4 : Pas de source connue pour les informations de révocation

La valeur d'état pour la vérification d'état de révocation d'une AC ainsi que pour la construction d'un chemin de certification produit par une AC à une racine de confiance avec une vérification complète d'état, { id-stc 7 }, peut être une des suivantes :
0 : Valide
1 : Non valide
2 : Révocation hors ligne
3 : Révocation indisponible
4 : Pas de source connue pour les informations de révocation

4.9.5 replyWantBacks

L'élément replyWantBacks contient les réponses à l'élément wantBack dans la demande. L'élément replyWantBacks inclut l'identifiant d'objet (OID) provenant de l'élément wantBack dans la demande et une CHAINE D'OCTETS. La valeur demandée est dans la CHAINE D'OCTETS. Les OID dans l'élément wantBack dans la demande sont utilisés pour identifier la valeur de réponse correspondante. Les OID dans l'élément replyWantBacks DOIVENT correspondre aux OID dans l'élément wantBack de la demande. Pour une réponse non d'erreur, replyWantBacks DOIT inclure exactement un ReplyWantBack pour chaque wantBack spécifié dans la demande (à l'exclusion de id-swb-pkc-cert et id-swb-ac-cert, où les informations demandées sont incluses dans l'élément cert).

L'élément replyWantBacks utilise le type ReplyWantBacks, qui a la syntaxe suivante :

```
ReplyWantBacks ::= SEQUENCE DE ReplyWantBack
ReplyWantBack   ::= SEQUENCE {
  wb                  IDENTIFIANT D'OBJET,
  value               CHAINE D'OCTETS }
```

La valeur de CHAÎNE D'OCTETS pour le chemin de certification utilisé pour vérifier le certificat dans la demande, { id-swb 1 }, contient le type CertBundle. La syntaxe et la sémantique du type CertBundle sont décrites au paragraphe 3.2.8. Ce CertBundle inclut tous les certificats dans le chemin, en commençant par le certificat de fin et se terminant par le certificat produit par l'ancre de confiance.

La valeur de CHAÎNE D'OCTETS pour la preuve de l'état de révocation, { id-swb 2 }, contient le type RevInfoWantBack. Le type RevInfoWantBack est une SEQUENCE du type RevocationInfos et un CertBundle facultatif. La syntaxe et la sémantique du type RevocationInfos sont décrites au paragraphe 3.2.9. Le CertBundle DOIT être inclus si des certificats requis pour valider les informations de révocation n'étaient pas retournés dans le wantBack id-swb-pkc-best-cert-path ou id-swb-pkc-all-cert-paths. Le CertBundle DOIT inclure tous ces certificats, mais il n'y a pas d'exigence d'ordre.

```
RevInfoWantBack ::= SEQUENCE {
  revocationInfo        RevocationInfos,
  extraCerts            CertBundle FACULTATIF }
```

La valeur de CHAÎNE D'OCTETS pour les informations de clé publique, { id-swb 4 }, contient le type SubjectPublicKeyInfo. La syntaxe et la sémantique du type SubjectPublicKeyInfo sont décrites dans la [RFC3280].

La valeur de CHAÎNE D'OCTETS pour le chemin de certification produit par une AC utilisé pour vérifier le certificat dans la demande, { id-swb 5 }, contient le type CertBundle. La syntaxe et la sémantique du type CertBundle sont décrites au paragraphe 3.2.8. Ce CertBundle inclut tous les certificats sur le chemin, en commençant par le certificat produit par l'AC et finissant par le certificat produit par l'ancre de confiance.

La valeur de CHAÎNE D'OCTETS pour la preuve de l'état de révocation du chemin de certification produit par l'AC, { id-swb 6 }, contient le type RevInfoWantBack. Le type RevInfoWantBack est une SEQUENCE du type RevocationInfos et d'un CertBundle facultatif. La syntaxe et la sémantique du type RevocationInfos sont décrites au paragraphe 3.2.9. Le

CertBundle DOIT être inclus si des certificats exigés pour valider les informations de révocation n'étaient pas retournés dans le wantBack id-aa-cert-path. Le CertBundle DOIT inclure tous ces certificats, mais il n'y a pas d'exigence d'ordre.

La valeur de CHAÎNE D'OCTETS pour la preuve de l'état de révocation du certificat d'attribut, { id-swb 7 }, contient le type RevInfoWantBack. Le type RevInfoWantBack est une SEQUENCE du type RevocationInfos et d'un CertBundle facultatif. La syntaxe et la sémantique du type RevocationInfos sont décrites au paragraphe 3.2.9. Le CertBundle DOIT être inclus si des certificats exigés pour valider les informations de révocation n'étaient pas retournés dans le wantBack id-swb-aa-cert-path. Le CertBundle DOIT inclure tous ces certificats, mais il n'y a pas d'exigence d'ordre.

La valeur de CHAÎNE D'OCTETS pour retourner tous les chemins, { id-swb 12 }, contient un type ASN.1 de CertBundles, comme défini ci-dessous. La syntaxe et la sémantique du type CertBundle sont décrites au paragraphe 3.2.8. Chaque CertBundle inclut tous les certificats sur un chemin, en commençant par le certificat de fin et en finissant par le certificat produit par l'ancre de confiance.

CertBundles ::= TAILLE DE SÉQUENCE (1..MAX) DE CertBundle

La valeur de CHAÎNE D'OCTETS pour les réponses relayées, { id-swb 9 }, contient un type ASN.1 de SCVPResponses, comme défini ci-dessous. Si le serveur SCVP a utilisé les informations obtenues des autres serveurs SCVP quand il a généré cette réponse, alors SCVPResponses DOIT inclure chacune des réponses SCVP reçues de ces serveurs. Si le serveur SCVP n'a pas utilisé les informations obtenues des autres serveurs SCVP quand il a généré la réponse, alors SCVPResponses DOIT être une séquence vide.

SCVPResponses ::= SEQUENCE DE ContentInfo

La valeur de CHAÎNE D'OCTETS pour la preuve de l'état de révocation du certificat cible du chemin, { id-swb-13 }, contient le type RevInfoWantBack. Le type RevInfoWantBack est une SEQUENCE du type RevocationInfos et d'un CertBundle facultatif. La syntaxe et la sémantique du type RevocationInfos sont décrites au paragraphe 3.2.9. Le CertBundle DOIT être inclus si des certificats exigés pour valider les informations de révocation n'étaient pas retournés dans le wantBack id-swb-pkc-best-cert-path ou id-swb-pkc-all-cert-paths. Le CertBundle DOIT inclure tous ces certificats, mais il n'y a pas d'exigence d'ordre.

La valeur de CHAÎNE D'OCTETS pour la preuve de l'état de révocation des certificats intermédiaires dans le chemin, { id-swb 14 }, contient le type RevInfoWantBack. Le type RevInfoWantBack est une SEQUENCE du type RevocationInfos et d'un CertBundle facultatif. La syntaxe et la sémantique du type RevocationInfos sont décrites au paragraphe 3.2.9. Le CertBundle DOIT être inclus si des certificats exigés pour valider les informations de révocation n'étaient pas retournés dans le wantBack id-swb-pkc-best-cert-path ou id-swb-pkc-all-cert-paths. Le CertBundle DOIT inclure tous ces certificats, mais il n'y a pas d'exigence d'ordre.

4.9.6 validationErrors

L'élément validationErrors DOIT seulement être présent dans les réponses d'échec. Si il est présent, il DOIT contenir un ou plusieurs OID représentant la raison de l'échec de validation (les erreurs de validation pour l'algorithme de validation de base et l'algorithme de validation de nom sont définies aux paragraphes 3.2.4.2.2 et 3.2.4.2.4). L'élément validationErrors DEVRAIT être inclus seulement quand replyStatus est 3, 5, 6, 7, ou 8. Les serveurs SCVP ne sont pas obligés de spécifier toutes les raisons de l'échec de validation. Les clients SCVP NE DOIVENT PAS supposer que les OID inclus dans validationErrors représentent toutes les erreurs de validation pour le chemin de certification.

4.9.7 nextUpdate

L'élément nextUpdate dit l'heure à laquelle le serveur attend qu'un rafraîchissement des informations concernant la validité du certificat devienne disponible. L'élément nextUpdate est particulièrement intéressant si les informations d'état de révocation du certificat ne sont pas disponibles ou si le certificat est suspendu. L'élément nextUpdate représente la date et l'heure en UTC, en utilisant le type GeneralizedTime. Les règles de codage pour GeneralizedTime du paragraphe 3.2.7 DOIVENT être utilisées.

4.9.8 certReplyExtensions

L'élément certReplyExtensions contient les réponses à l'élément queryExtensions dans la demande. L'élément certReplyExtensions utilise le type Extensions défini dans la [RFC3280]. Les identifiants d'objet (OID) dans l'élément

queryExtensions de la demande sont utilisés pour identifier les valeurs de réponse correspondantes. L'élément certReplyExtensions , quand il est présent, contient une séquence d'éléments Extension, dont chacun contient un élément extnID, un élément critical, et un élément extnValue.

L'élément extnID est un identifiant pour l'extension. Il contient l'OID qui nomme l'extension, et il DOIT correspondre à un des OID dans l'élément queryExtensions de la demande.

L'élément critical est un BOOLÉEN, et il DOIT être réglé à FAUX.

L'élément extnValue contient une CHAINE D'OCTETS. La valeur d'extension est dans la CHAINE D'OCTETS. Un type ASN.1 est spécifié pour chaque extension, identifiée par l'identifiant d'objet extnID associé.

4.10 respNonce

L'élément respNonce contient un identifiant pour lier la demande à la réponse.

Si le client inclut une valeur de requestNonce dans la demande et si le serveur génère une réponse non mise en antémémoire spécifique de la demande, alors le serveur DOIT retourner la même valeur dans la réponse.

Si le serveur utilise une réponse mise en antémémoire à la demande, il DOIT alors omettre l'élément respNonce.

Si le serveur retourne une réponse spécifique non mise en antémémoire à une demande sans un nom occasionnel *(nonce)* alors le serveur PEUT inclure un nom occasionnel spécifique du message. Pour les messages signés numériquement, le serveur PEUT utiliser la valeur de l'attribut Résumé de message dans les signedAttrs au sein des SignerInfo de la demande comme valeur dans l'élément respNonce.

L'élément requestNonce utilise le type CHAINE D'OCTETS.

Les mises en œuvre de client conformes DOIVENT être capables de traiter une réponse qui inclut cet élément. Les serveurs conformes DOIVENT prendre en charge respNonce.

4.11 serverContextInfo

L'élément serverContextInfo dans une réponse est un mécanisme pour que le serveur passe des informations opaques de contexte au client. Si le client n'aime pas le chemin de certification retourné, il peut faire une nouvelle interrogation et la passer avec ces informations de contexte.

Le paragraphe 3.2.6 contient des informations sur l'utilisation de cet élément par le client.

Les informations de contexte sont opaques pour le client, mais elles donnent des informations au serveur qui assurent qu'un chemin de certification différent va être retourné (si un autre peut être trouvé). Les informations de contexte pourraient indiquer l'état du serveur, ou elles pourraient contenir une séquence de hachages des chemins de certification qui auraient déjà été retournés au client. Le protocole n'impose aucune structure ou exigence pour cet élément. Cependant, les mises en œuvre devraient revoir la section Considérations sur la sécurité de ce document avant de choisir une structure.

Les serveurs qui sont incapables de retourner des chemins supplémentaires NE DOIVENT PAS inclure l'élément serverContextInfo dans la réponse.

4.12 cvResponseExtensions

Si il est présent, l'élément cvResponseExtensions contient une séquence d'extensions qui étendent la réponse. La présente spécification ne définit aucune extension. La facilité est fournie pour permettre que de futures spécifications étendent SCVP. La syntaxe de Extensions est importée de la [RFC3280]. L'élément cvResponseExtensions, quand il est présent, contient une séquence d'éléments Extension, dont chacun contient un élément extnID, un élément critical, et un élément extnValue.

L'élément extnID est un identifiant de l'extension. Il contient l'identifiant d'objet (OID) qui désigne l'extension.

L'élément critical est un BOOLÉEN. Chaque extension est désignée comme critique (avec une valeur de VRAI) ou non

critique (avec une valeur de FAUX). Un client SCVP DOIT rejeter la réponse si il rencontre une extension critique qu'il ne reconnaît pas ; cependant, une extension non critique PEUT être ignorée si elle n'est pas reconnue.

L'élément extnValue contient une CHAINE D'OCTETS. La valeur d'extension est dans la CHAINE D'OCTETS. Un type ASN.1 est spécifié pour chaque extension, identifiée par l'identifiant d'objet extnID associé.

4.13 requestorText

L'élément requestorText contient un champ text fourni par le client.

Si le client inclut une valeur requestorText dans la demande et si le serveur génère une réponse spécifique non mise en antémémoire à la demande, le serveur DOIT alors retourner la même valeur dans la réponse.

Si le serveur utilise une réponse mise en antémémoire à la demande, il DOIT alors omettre l'élément requestorText.

L'élément requestNonce utilise le type chaîne UTF8.

Les mises en œuvre de client conformes qui prennent en charge l'élément requestorText dans les demandes (voir le paragraphe 3.10) DOIVENT être capables de traiter une réponse qui inclut cet élément. Les serveurs conformes DOIVENT prendre en charge requestorText dans les réponses.

4.14 Validation de réponse SCVP

Il y a deux mécanismes pour la validation des réponses SCVP, une fondée sur la connaissance par le client d'une clé spécifique du serveur SCVP et l'autre fondée sur la validation du certificat correspondant à la clé privée utilisée pour protéger la réponse SCVP.

4.14.1 Validation simple de clé

La méthode de validation simple de clé est lorsque le client SCVP a une politique locale de une ou plusieurs clés de serveur SCVP qui identifient directement l'ensemble de serveurs SCVP valides. Les mécanismes pour la mémorisation des clés de serveur ou des identifiants sont une affaire locale. Par exemple, un client pourrait mémoriser des hachages cryptographiques des clés publiques utilisées pour vérifier les réponses SignedData. Autrement, un client pourrait mémoriser des clés symétriques partagées utilisées pour vérifier les MAC dans les réponses AuthenticatedData.

La validation simple de clé DOIT être utilisée par les clients SCVP qui ne peuvent pas valider les certificats PKIX-1 et font donc des demandes de validation de chemin déléguée au serveur SCVP [RFC3379]. C'est une affaire de politique locale de ces clients d'utiliser SignedData ou AuthenticatedData. La validation simple de clé PEUT être utilisée par les autres clients SCVP pour d'autres raisons.

4.14.2 Validation de certificat de serveur SCVP

C'est une affaire de politique locale de décider quelle politique de validation le client utilise quand il valide les réponses. Quand ils valident des réponses protégées par SCVP, les clients SCVP DEVRAIENT utiliser l'algorithme de validation défini à la Section 6 de la [RFC3280]. Les clients SCVP peuvent imposer des limitations supplémentaires à l'algorithme, comme de limiter le nombre de certificats dans le chemin ou d'établir des contraintes de nom initial, comme spécifié au paragraphe 6.2 de la [RFC3280].

Si le certificat utilisé pour signer les réponses de politique de validation et si les réponses de validation SignedData contiennent l'extension d'usage de clé ([RFC3280], paragraphe 4.2.1.3) il DOIT soit avoir le bit Signature numérique établi, soit le bit Non répudiation établi, soit les deux bits établis.
Si le certificat pour les réponses de validation AuthenticatedData contient l'extension d'usage de clé, il DOIT avoir le bit Accord de clé établi.

Si le certificat utilisé sur une réponse de politique de validation ou une réponse de validation contient l'extension usage de clé étendu ([RFC3280], paragraphe 4.2.1.13) il DOIT contenir l'OID anyExtendedKeyUsage ou l'OID suivant :

IDENTIFIANT D'OBJET id-kp-scvpServer ::= { id-kp 15 }

5. Demande de politique de serveur

Un client SCVP utilise l'élément ValPolRequest pour demander des informations sur les politiques et les informations de configuration d'un serveur SCVP, incluant la liste des politiques de validation prises en charge par le serveur SCVP. Quand une ValPolRequest est encapsulée dans une partie de corps MIME, elle DOIT être portée dans une partie de corps MIMIE application/scvp-vp-request.

La demande consiste en une ValPolRequest encapsulée dans un ContentInfo. Le client ne signe pas la demande.

```
ContentInfo {
  contentType    id-ct-scvp-valPolRequest,          -- (1.2.840.113549.1.9.16.1.12)
  content        ValPolRequest }
```

Le type ValPolRequest a la syntaxe suivante :

```
ValPolRequest ::= SEQUENCE {
  vpRequestVersion      ENTIER DEFAUT 1,
  requestNonce          CHAINE D'OCTETS }
```

Les mises en œuvre conformes de serveur SCVP DOIVENT reconnaître et traiter la demande de politique du serveur. Les clients conformes DEVRAIENT prendre en charge la demande de politique du serveur.

5.1 vpRequestVersion

La syntaxe et la sémantique de vpRequestVersion sont les mêmes que celles de cvRequestVersion, comme décrit au paragraphe 3.1.

5.2 requestNonce

L'élément requestNonce contient un identifiant de demande généré par le client SCVP. Si le serveur retourne une réponse spécifique, il DOIT inclure le requestNonce provenant de la demande dans la réponse, mais le serveur PEUT retourner une réponse mise en antémémoire, qui NE DOIT PAS inclure de requestNonce.

6. Réponse de politique de validation

En réponse à une ValPolRequest, le serveur SCVP fournit une réponse ValPolResponse. La ValPolResponse peut n'être pas unique pour toute ValPolRequest, de sorte qu'elle peut être réutilisée par le serveur en réponse à plusieurs ValPolRequest. La ValPolResponse a aussi une indication de la fréquence à laquelle la ValPolResponse peut être réutilisée. Le serveur DOIT signer la réponse en utilisant son certificat de signature numérique. Quand une ValPolResponse est encapsulée dans une partie de corps MIME, elle DOIT être portée dans une partie de corps MIME application/scvp-vp-response.

La réponse consiste en une ValPolResponse encapsulée dans un SignedData, qui est à son tour encapsulée dans un ContentInfo. C'est-à-dire, le champ EncapsulatedContentInfo de SignedData consiste en un champ eContentType avec une valeur de id-ct-scvp-valPolResponse (1.2.840.113549.1.9.16.1.13) et un champ eContent qui contient une ValPolResponse codée en DER. Le serveur SCVP DOIT inclure son propre certificat dans le champ certificates au sein de SignedData, et le champ signerInfos de SignedData DOIT inclure exactement une SignerInfo. Le SignedData NE DOIT PAS inclure de champ unsignedAttrs.

Le type ValPolResponse a la syntaxe suivante :

```
ValPolResponse ::= SEQUENCE {
  vpResponseVersion       ENTIER,
  maxCVRequestVersion     ENTIER,
  maxVPRequestVersion     ENTIER,
```

```
  serverConfigurationID     ENTIER,
  thisUpdate                GeneralizedTime,
  nextUpdate                GeneralizedTime FACULTATIF,
  supportedChecks           CertChecks,
  supportedWantBacks        WantBack,
  validationPolicies        SEQUENCE DE IDENTIFIANT D'OBJET,
  validationAlgs            SEQUENCE DE IDENTIFIANT D'OBJET,
  authPolicies              SEQUENCE DE AuthPolicy,
  responseTypes             ResponseTypes,
  defaultPolicyValues       RespValidationPolicy,
  revocationInfoTypes       RevocationInfoTypes,
  signatureGeneration       SEQUENCE DE AlgorithmIdentifier,
  signatureVerification     SEQUENCE DE AlgorithmIdentifier,
  hashAlgorithms            TAILLE DE SÉQUENCE (1..MAX) DE IDENTIFIANT D'OBJET,
  serverPublicKeys          SEQUENCE DE KeyAgreePublicKey FACULTATIF,
  clockSkew                 ENTIER DEFAUT 10,
  requestNonce              CHAINE D'OCTETS FACULTATIF }

ResponseTypes  ::= ENUMERATED {
  cached-only               (0),
  non-cached-only           (1),
  cached-and-non-cached     (2) }

RevocationInfoTypes ::= CHAINE DE BITS {
  fullCRLs                  (0),
  deltaCRLs                 (1),
  indirectCRLs              (2),
  oCSPResponses             (3) }
```

Les clients SCVP qui prennent en charge les demandes de politique de validation DOIVENT prendre en charge les réponses de politique de validation. Les serveurs SCVP DOIVENT prendre en charge les réponses de politique de validation.

Les serveurs SCVP DOIVENT prendre en charge les réponses de politique mises en antémémoire et PEUVENT prendre en charge les réponses spécifiques aux demandes de politique.

6.1 vpResponseVersion

La syntaxe et la sémantique de l'élément vpResponseVersion sont les mêmes que celles de cvRequestVersion comme décrit au paragraphe 3.1. La vpResponseVersion utilisée DOIT être la même que celle de vpRequestVersion sauf si le client a utilisé une valeur supérieure aux valeurs que le serveur accepte. Si le client soumet une vpRequestVersion supérieure à la version prise en charge par le serveur, le serveur DOIT retourner une vpResponseVersion en utilisant le plus haut numéro de version qu'il prend en charge comme numéro de version.

6.2 maxCVRequestVersion

L'élément maxCVRequestVersion définit le numéro de version maximum pour les demandes de CV que le serveur supporte.

6.3 maxVPRequestVersion

L'élément maxVPRequestVersion définit le numéro de version maximum pour les demandes de VP que le serveur supporte.

6.4 serverConfigurationID

L'élément serverConfigurationID est un entier qui représente de façon univoque la version de configuration de serveur comme représentée par validationPolicies, validationAlgs, authPolicies, defaultPolicyValues, et clockSkew. Si une de ces

valeurs change, le serveur DOIT créer une nouvelle ValPolResponse avec un nouveau serverConfigurationID. Si la configuration n'a pas changé, alors le serveur peut réutiliser serverConfigurationID sur plusieurs messages ValPolResponse. Cependant, si le serveur revient à une configuration antérieure, il NE DOIT PAS revenir aussi à l'identifiant de configuration, mais DOIT choisir une autre valeur univoque.

6.5 thisUpdate

Cet élément indique la date et heure de signature de cette réponse de politique.

Les valeurs de GeneralizedTime DOIVENT être exprimées en temps moyen de Greenwich (Zulu) et interprétées comme défini au paragraphe 3.2.7.

6.6 nextUpdate et requestNonce

Ces éléments sont utilisés pour indiquer si les réponses de politique sont spécifiques des demandes de politique. Lorsque les réponses de politique sont mises en antémémoire, ces éléments indiquent quand les informations vont être mises à jour. L'élément facultatif nextUpdate indique l'heure à laquelle la prochaine réponse de politique va être publiée. L'élément facultatif requestNonce relie la réponse à une demande spécifique en retournant le nom occasionnel fourni dans la demande.

Si l'élément nextUpdate est omis, cela indique une réponse non mise en antémémoire générée en réponse à une demande spécifique (c'est-à-dire, la ValPolResponse est liée à une demande spécifique). Si cet élément est omis, l'élément requestNonce DOIT être présent et DOIT inclure la valeur de requestNonce provenant de la demande.

Si l'élément nextUpdate est présent, cela indique une réponse mise en antémémoire qui n'est pas liée à une demande spécifique. Un serveur SCVP DOIT périodiquement générer une nouvelle réponse comme défini par l'heure de prochaine mise à jour, mais PEUT utiliser la même ValPolResponse pour répondre à de multiples demandes. Le requestNonce est omis si l'élément nextUpdate est présent.

C'est une affaire de politique locale du serveur de retourner une réponse spécifique mise en antémémoire ou non.

Les valeurs de GeneralizedTime dans nextUpdate DOIVENT être exprimées en temps moyen de Greenwich (Zulu) et interprétées comme défini au paragraphe 3.2.7.

6.7 supportedChecks

L'élément supportedChecks contient une séquence d'identifiants d'objet représentant les vérifications prises en charge par le serveur.

6.8 supportedWantBacks

L'élément supportedWantBacks contient une séquence d'identifiants d'objet représentant les wantBacks pris en charge par le serveur.

6.9 validationPolicies

L'élément validationPolicies contient une séquence d'identifiants d'objet représentant les politiques de validation prises en charge par le serveur. C'est une affaire de politique locale si le serveur souhaite traiter les demandes en utilisant la politique de validation par défaut, et si il ne le fait pas, alors il NE DOIT PAS inclure le id-svp-defaultValPolicy dans cette liste.

6.10 validationAlgs

L'élément validationAlgs contient une séquence d'OID. Chaque OID identifie un algorithme de validation pris en charge par le serveur.

6.11 authPolicies

L'élément authPolicies contient une séquence de références de politiques pour s'authentifier auprès du serveur SCVP.

La référence à la politique d'authentification est un OID sur lequel le client et le serveur se sont accordés qui représente une politique d'authentification. La liste des politiques est destinée à indiquer au client si l'authentification est requise pour certaines demandes et comment.

AuthPolicy ::= IDENTIFIANT D'OBJET

6.12 responseTypes

L'élément responseTypes permet au serveur de publier la gamme de types de réponse qu'il accepte. Cached-only signifie que le serveur va seulement retourner des réponses mises en antémémoire aux demandes. Non-cached-only signifie que le serveur va retourner une réponse spécifique à la demande, c'est-à-dire, contenant le nom occasionnel du demandeur. Les deux signifie que le serveur supporte les deux types de réponse mise en antémémoire et réponse non mise en antémémoire et va retourner une réponse mise en antémémoire ou une réponse non mise en antémémoire, selon la demande.

6.13 revocationInfoTypes

L'élément revocationInfoTypes permet au serveur d'indiquer les sources des informations de révocation qu'il est capable de traiter. Pour chaque bit dans la CHAINE DE BITS RevocationInfoTypes, le serveur DOIT régler le bit à un si il est capable de traiter le type d'informations de révocation correspondant et à zéro si il ne le peut pas.

6.14 defaultPolicyValues

C'est la politique de validation par défaut utilisée par le serveur. Elle contient une RespValidationPolicy, qui est définie au paragraphe 4.5. Tous les éléments FACULTATIFS dans l'élément validationPolicy DOIVENT être remplis. Un serveur va utiliser ces valeurs par défaut quand la demande fait référence à la politique de validation par défaut et que le client n'outrepasse pas les valeurs par défaut en fournissant d'autres valeurs dans la demande.

Cela permet au client d'optimiser la demande en omettant des paramètres qui correspondent aux valeurs par défaut du serveur.

6.15 signatureGeneration

Cette séquence spécifie l'ensemble d'algorithmes de signature numérique pris en charge par un serveur SCVP pour signer les messages CVResponse. Chaque algorithme de signature numérique est spécifié comme un AlgorithmIdentifier, en utilisant les règles de codage associées au champ signatureAlgorithm dans un certificat de clé publique [RFC3280]. Les algorithmes pris en charge sont définis dans les [RFC3279] et [RFC4055], mais d'autres algorithmes de signature peuvent aussi être pris en charge.

En includant un algorithme (par exemple, RSA avec SHA-1) dans cette liste, le serveur déclare qu'il a une clé privée et la clé publique certifiée correspondante pour cet algorithme asymétrique, et qu'il prend en charge l'algorithme de hachage spécifié. La liste est ordonnée ; le premier algorithme de signature numérique est l'algorithme par défaut du serveur. L'algorithme par défaut va être utilisé par le serveur pour protéger les messages signés sauf si le client spécifie un autre algorithme.

Pour les serveurs qui n'ont pas de clé privée en ligne, et ne peuvent pas signer les messages CVResponse, l'élément signatureGeneration est codé comme une séquence vide.

6.16 signatureVerification

Cette séquence spécifie l'ensemble d'algorithmes de signature numérique qui peut être vérifié par ce serveur SCVP. Chaque algorithme de signature numérique est spécifié comme un AlgorithmIdentifier, en utilisant les règles de codage associées à au champ signatureAlgorithm dans un certificat de clé publique [RFC3280]. Les algorithmes pris en charge sont définis dans les [RFC3279] et [RFC4055], mais d'autres algorithmes de signature peuvent aussi être pris en charge.

Pour les serveurs qui ne vérifient pas les signatures sur les messages CVRequest, l'élément signatureVerification est codé comme une séquence vide.

6.17 hashAlgorithms

Cette séquence spécifie l'ensemble d'algorithmes de hachage que le serveur peut utiliser pour hacher les certificats et les demandes. La liste est ordonnée ; le premier algorithme de hachage est l'algorithme par défaut du serveur. L'algorithme par défaut va être utilisé par le serveur pour calculer les hachages inclus dans les réponses, sauf si le client spécifie un autre algorithme. Chaque algorithme de hachage est spécifié comme un identifiant d'objet. La [RFC4055] spécifie les identifiants d'objet pour SHA-1, SHA-224, SHA-256, SHA-384, et SHA-512. D'autres algorithmes de hachage peuvent aussi être pris en charge.

6.18 serverPublicKeys

L'élément serverPublicKeys est une séquence d'une ou plusieurs clés publiques d'accord de clé et paramètres associés. Il est utilisé par les clients qui font des demandes AuthenticatedData au serveur. Chaque élément dans la séquence serverPublicKeys est du type KeyAgreePublicKey :

```
KeyAgreePublicKey ::= SEQUENCE {
  algorithm          AlgorithmIdentifier,
  publicKey          CHAINE DE BITS,
  macAlgorithm       AlgorithmIdentifier,
  kDF                AlgorithmIdentifier FACULTATIF }
```

L'élément KeyAgreePublicKey inclut l'identifiant d'algorithme et la clé publique du serveur. Les serveurs SCVP qui prennent en charge le mode d'accord de clé de AuthenticatedData pour les demandes SCVP DOIVENT prendre en charge serverPublicKeys et l'algorithme d'accord de clé Diffie-Hellman comme spécifié dans la [RFC3279]. Les serveurs SCVP qui prennent en charge serverPublicKeys DOIVENT prendre en charge la clé de groupe (groupe 2) modulaire exponentielle de 1024 bits (MODP, *Modular Prime exponential*) comme défini dans la [RFC4306]. Les serveurs SCVP qui prennent en charge serverPublicKeys PEUVENT prendre en charge d'autres groupes Diffie-Hellman [RFC3526], ainsi que d'autres algorithmes d'accord de clé.

L'élément macAlgorithm spécifie l'algorithme symétrique que le serveur s'attend à ce que le client utilise avec le résultat de l'algorithme d'accord de clé. Une fonction de déduction de clé (KDF, *key derivation function*) qui déduit le matériel de clé symétrique du résultat d'accord de clé, peut être impliquée par le macAlgorithm. Autrement, la KDF peut être explicitement spécifiée en utilisant l'élément facultatif kDF.

6.19 clockSkew

L'élément clockSkew est le nombre de minutes que le serveur va permettre pour le biais d'horloge. La valeur par défaut est 10 minutes.

7. Relais de serveur SCVP

Dans certains environnements de réseau, en particulier ceux qui comportent des pare-feu, un serveur SCVP pourrait n'être pas capable d'obtenir toutes les informations dont il a besoin pour traiter une demande. Cependant, le serveur pourrait être configuré à utiliser les services d'un ou plusieurs autres serveurs SCVP pour satisfaire toutes les demandes. Dans ce cas, le client SCVP ne sait pas que le serveur SCVP initial utilise les services d'autres serveurs SCVP. Le serveur SCVP initial agit comme un client pour un autre serveur SCVP. À la différence du client d'origine, le serveur SCVP est supposé avoir des ressources modérées de calcul et de mémoire. Cette Section décrit les échanges de serveur SCVP à serveur SCVP. Elle n'impose aucune exigence aux clients SCVP qui ne sont pas aussi des serveurs SCVP. De plus, cette Section n'impose aucune exigence aux serveurs SCVP qui ne relayent pas les demandes aux autres serveurs SCVP.

Quand un serveur SCVP relaye une demande à un autre serveur, dans un système de serveur incorrectement configuré, il est possible que la même demande soit relayée à nouveau en retour. Tout serveur SCVP qui relaye des demandes DOIT mettre en œuvre les conventions décrites dans cette section pour détecter et dénouer les boucles.

Quand un serveur SCVP relaye une demande, la demande DOIT inclure l'élément requestorRef. Si la demande à relayer contient déjà un élément requestorRef, alors la demande générée par le serveur DOIT contenir un élément requestorRef construit à partir de cette valeur et un GeneralName supplémentaire qui contient un identifiant du serveur SCVP. Si la demande à relayer ne contient pas d'élément requestorRef, alors la demande générée par le serveur DOIT contenir un élément requestorRef qui inclut un GeneralName qui contient un identifiant du serveur SCVP.

Pour empêcher une fausse détection de boucle, les serveurs devraient utiliser des identifiants qui sont uniques au sein de leur réseau de serveurs SCVP coopérants. Les serveurs SCVP qui prennent en charge le relais DEVRAIENT remplir cet élément avec le nom DNS du serveur ou le nom distinctif dans le certificat du serveur. Les serveurs SCVP PEUVENT choisir d'autres procédures pour générer des identifiants uniques au sein de leur communauté.

Quand un serveur SCVP reçoit une demande qui contient un élément requestorRef, le serveur DOIT vérifier la séquence des noms dans l'élément requestorRef pour voir si il y a son propre identifiant. Si le serveur découvre son propre identifiant dans l'élément requestorRef, il DOIT répondre par une erreur, réglant le statusCode dans l'élément responseStatus à 40.

Quand un serveur SCVP génère une réponse non mise en antémémoire à une demande relayée, le serveur DOIT inclure l'élément requestorRef provenant de la demande dans la réponse.

8. Module ASN.1 SCVP

Cette Section définit la syntaxe pour les paires de demande-réponse SCVP. La sémantique des messages est définie dans les Sections 3, 4, 5, et 6. Le module ASN.1 de SCVP est :

```
SCVP

{ iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0) 21 }

ÉTIQUETTES IMPLICITES DE DÉFINITIONS ::= DÉBUT

IMPORTS

AlgorithmIdentifier, Attribute, Certificate, Extensions,
-- Import UTF8String si exigé par le compilateur
-- UTF8String, -- CertificateList, CertificateSerialNumber
    FROM PKIX1Explicit88 -- RFC 3280
    { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0) 18 }

GeneralNames, GeneralName, KeyUsage, KeyPurposeId
 FROM PKIX1Implicit88 -- RFC 3280
 { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0) 19 }

AttributeCertificate
 FROM PKIXAttributeCertificate -- RFC 3281
 { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0) 12 }

OCSPResponse
 FROM OCSP -- RFC 2560
 { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0) 14 }

ContentInfo
 FROM CryptographicMessageSyntax2004 -- RFC 3852
 { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0) cms-2004(24) } ;

-- Demande de validation de certificat SCVP

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

IDENTIFIANT D'OBJET  id-ct-scvp-certValRequest ::= { id-ct 10 }
```

```
CVRequest ::= SEQUENCE {
  cvRequestVersion    ENTIER DEFAUT 1,
  query               Query,
  requestorRef        [0] GeneralNames FACULTATIF,
  requestNonce        [1] CHAINE D'OCTETS FACULTATIF,
  requestorName       [2] GeneralName FACULTATIF,
  responderName       [3] GeneralName FACULTATIF,
  requestExtensions   [4] Extensions FACULTATIF,
  signatureAlg        [5] AlgorithmIdentifier FACULTATIF,
  hashAlg             [6] IDENTIFIANT D'OBJET FACULTATIF,
  requestorText       [7] UTF8String (SIZE (1..256)) FACULTATIF }

Query ::= SEQUENCE {
  queriedCerts        CertReferences,
  checks              CertChecks,
  wantBack            [1] WantBack FACULTATIF,
  validationPolicy       ValidationPolicy,
  responseFlags        ;  ResponseFlags FACULTATIF,
  serverContextInfo   [2] CHAINE D'OCTETS FACULTATIF,
  validationTime      [3] GeneralizedTime FACULTATIF,
  intermediateCerts   [4] CertBundle FACULTATIF,
  revInfos            [5] RevocationInfos FACULTATIF,
  producedAt          [6] GeneralizedTime FACULTATIF,
  queryExtensions     [7] Extensions FACULTATIF }

CertReferences ::= CHOIX {
  pkcRefs      [0] TAILLE DE SÉQUENCE (1..MAX) DE PKCReference,
  acRefs       [1] TAILLE DE SÉQUENCE (1..MAX) DE ACReference }

CertReference::= CHOIX {
  pkc          PKCReference,
  ac           ACReference }

PKCReference ::= CHOIX {
  cert         [0] Certificate,
  pkcRef       [1] SCVPCertID }

ACReference ::= CHOIX {
  attrCert     [2] AttributeCertificate,
  acRef        [3] SCVPCertID }

SCVPCertID ::= SEQUENCE {
  certHash         CHAINE D'OCTETS,
  issuerSerial     SCVPIssuerSerial,
  hashAlgorithm  AlgorithmIdentifier DEFAUT { algorithm sha-1 } }

SCVPIssuerSerial ::= SEQUENCE {
  issuer           GeneralNames,
  serialNumber   CertificateSerialNumber
}

ValidationPolicy ::= SEQUENCE {
  validationPolRef       ValidationPolRef,
  validationAlg          [0] ValidationAlg FACULTATIF,
  userPolicySet          [1] TAILLE DE SÉQUENCE (1..MAX) DE IDENTIFIANT D'OBJET FACULTATIF,
  inhibitPolicyMapping [2] BOOLEEN FACULTATIF,
  requireExplicitPolicy  [3] BOOLEEN FACULTATIF,
  inhibitAnyPolicy       [4] BOOLEEN FACULTATIF,
  trustAnchors           [5] TrustAnchors FACULTATIF,
  keyUsages              [6] SEQUENCE DE KeyUsage FACULTATIF,
  extendedKeyUsages    [7] SEQUENCE DE KeyPurposeId FACULTATIF,
```

```
  specifiedKeyUsages   [8] SEQUENCE DE KeyPurposeId FACULTATIF }

CertChecks ::= TAILLE DE SÉQUENCE (1..MAX) DE IDENTIFIANT D'OBJET

WantBack ::= TAILLE DE SÉQUENCE (1..MAX) DE IDENTIFIANT D'OBJET

ValidationPolRef ::= SEQUENCE {
  valPolId              IDENTIFIANT D'OBJET,
  valPolParams        ANY DEFINI PAR valPolId FACULTATIF }

ValidationAlg ::= SEQUENCE {
  valAlgId             IDENTIFIANT D'OBJET,
  parameters          ANY DEFINI PAR valAlgId FACULTATIF }

NameValidationAlgParms ::= SEQUENCE {
  nameCompAlgId         IDENTIFIANT D'OBJET,
  validationNames         GeneralNames }

TrustAnchors ::= TAILLE DE SÉQUENCE (1..MAX) DE PKCReference

KeyAgreePublicKey ::= SEQUENCE {
  algorithm          AlgorithmIdentifier,
  publicKey          CHAINE DE BITS,
  macAlgorithm     AlgorithmIdentifier,
  kDF                  AlgorithmIdentifier FACULTATIF }

ResponseFlags ::= SEQUENCE {
  fullRequestInResponse         [0] BOOLEEN DEFAUT FAUX,
  responseValidationPolByRef [1] BOOLEEN DEFAUT VRAI,
  protectResponse                  [2] BOOLEEN DEFAUT VRAI,
  cachedResponse                   [3] BOOLEEN DEFAUT VRAI }

CertBundle ::= TAILLE DE SÉQUENCE (1..MAX) DE Certificate

RevocationInfos ::= TAILLE DE SÉQUENCE (1..MAX) DE RevocationInfo

RevocationInfo ::= CHOIX {
  crl                [0] CertificateList,
  delta-crl          [1] CertificateList,
  ocsp               [2] OCSPResponse,
  autre               [3] OtherRevInfo }

OtherRevInfo ::= SEQUENCE {
  riType                  IDENTIFIANT D'OBJET,
  riValue                  ANY DEFINI PAR riType }

-- Réponse de validation de certificat SCVP

IDENTIFIANT D'OBJET  id-ct-scvp-certValResponse ::= { id-ct 11 }

CVResponse ::= SEQUENCE {
  cvResponseVersion       ENTIER,
  serverConfigurationID    ENTIER,
  producedAt                 GeneralizedTime,
  responseStatus              ResponseStatus,
  respValidationPolicy     [0] RespValidationPolicy FACULTATIF,
  requestRef                  [1] RequestReference FACULTATIF,
  requestorRef                [2] GeneralNames FACULTATIF,
  requestorName               [3] GeneralNames FACULTATIF,
  replyObjects                [4] ReplyObjects FACULTATIF,
  respNonce                   [5] CHAINE D'OCTETS FACULTATIF,
```

```
  serverContextInfo         [6] CHAINE D'OCTETS FACULTATIF,
  cvResponseExtensions   [7] Extensions FACULTATIF,
  requestorText             [8] UTF8String (SIZE (1..256)) FACULTATIF }

ResponseStatus ::= SEQUENCE {
  statusCode            CVStatusCode DEFAUT  okay,
  errorMessage          UTF8String FACULTATIF }

CVStatusCode ::= ENUMERATED {
  okay                                  (0),
  skipUnrecognizedItems                 (1),
  tooBusy                              (10),
  invalidRequest                       (11),
  internalError                        (12),
  badStructure                         (20),
  unsupportedVersion                   (21),
  abortUnrecognizedItems               (22),
  unrecognizedSigKey                   (23),
  badSignatureOrMAC                    (24),
  unableToDecode                       (25),
  notAuthorized                        (26),
  unsupportedChecks                    (27),
  unsupportedWantBacks                 (28),
  unsupportedSignatureOrMAC            (29),
  invalidSignatureOrMAC                (30),
  protectedResponseUnsupported         (31),
  unrecognizedResponderName            (32),
  relayingLoop                         (40),
  unrecognizedValPol                   (50),
  unrecognizedValAlg                   (51),
  fullRequestInResponseUnsupported     (52),
  fullPolResponseUnsupported           (53),
  inhibitPolicyMappingUnsupported      (54),
  requireExplicitPolicyUnsupported     (55),
  inhibitAnyPolicyUnsupported          (56),
  validationTimeUnsupported            (57),
  unrecognizedCritQueryExt             (63),
  unrecognizedCritRequestExt           (64) }

RespValidationPolicy ::= ValidationPolicy

RequestReference ::= CHOIX {
  requestHash   [0] HashValue,                  -- hachage de CVRequest
  fullRequest    [1] CVRequest }

HashValue ::= SEQUENCE {
  algorithm        AlgorithmIdentifier DEFAUT { algorithm sha-1 },
  value            CHAINE D'OCTETS }

IDENTIFIANT D'OBJET sha-1 ::= { iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) 26 }

ReplyObjects ::= TAILLE DE SÉQUENCE (1..MAX) OF CertReply

CertReply ::= SEQUENCE {
  cert                  CertReference,
  replyStatus           ReplyStatus DEFAUT success,
  replyValTime          GeneralizedTime,
  replyChecks           ReplyChecks,
  replyWantBacks        ReplyWantBacks,
  validationErrors      [0] TAILLE DE SÉQUENCE (1..MAX) DE IDENTIFIANT D'OBJET FACULTATIF,
  nextUpdate            [1] GeneralizedTime FACULTATIF,
```

```
  certReplyExtensions  [2] Extensions FACULTATIF }

ReplyStatus ::= ENUMERATED {
  success                 (0),
  malformedPKC            (1),
  malformedAC             (2),
  unavailableValidationTime  (3),
  referenceCertHashFail   (4),
  certPathConstructFail   (5),
  certPathNotValid        (6),
  certPathNotValidNow     (7),
  wantBackUnsatisfied     (8) }

ReplyChecks ::= SEQUENCE DE ReplyCheck

ReplyCheck ::= SEQUENCE {
  check                 IDENTIFIANT D'OBJET,
  status                ENTIER DEFAUT 0 }

ReplyWantBacks ::= SEQUENCE DE ReplyWantBack

ReplyWantBack::= SEQUENCE {
  wb                    IDENTIFIANT D'OBJET,
  value                 CHAINE D'OCTETS }

CertBundles ::= TAILLE DE SÉQUENCE (1..MAX) DE CertBundle

RevInfoWantBack ::= SEQUENCE {
  revocationInfo        RevocationInfos,
  extraCerts            CertBundle FACULTATIF }

SCVPResponses ::= SEQUENCE DE ContentInfo

-- Demande de politique de validation SCVP

IDENTIFIANT D'OBJET id-ct-scvp-valPolRequest    ::= { id-ct 12 }

ValPolRequest ::= SEQUENCE {
  vpRequestVersion      ENTIER DEFAUT 1,
  requestNonce          CHAINE D'OCTETS }

-- Réponse de politique de validation SCVP

IDENTIFIANT D'OBJET id-ct-scvp-valPolResponse ::= { id-ct 13 }

ValPolResponse ::= SEQUENCE {
  vpResponseVersion         ENTIER,
  maxCVRequestVersion       ENTIER,
  maxVPRequestVersion       ENTIER,
  serverConfigurationID     ENTIER,
  thisUpdate                GeneralizedTime,
  nextUpdate                GeneralizedTime FACULTATIF,
  supportedChecks           CertChecks,
  supportedWantBacks        WantBack,
  validationPolicies        SEQUENCE DE IDENTIFIANT D'OBJET,
  validationAlgs            SEQUENCE DE IDENTIFIANT D'OBJET,
  authPolicies              SEQUENCE DE AuthPolicy,
  responseTypes             ResponseTypes,
  defaultPolicyValues       RespValidationPolicy,
  revocationInfoTypes       RevocationInfoTypes,
  signatureGeneration       SEQUENCE DE AlgorithmIdentifier,
```

```
  signatureVerification          SEQUENCE DE AlgorithmIdentifier,
  hashAlgorithms                 TAILLE DE SÉQUENCE (1..MAX) DE IDENTIFIANT D'OBJET,
  serverPublicKeys               SEQUENCE DE KeyAgreePublicKey FACULTATIF,
  clockSkew                      ENTIER DEFAUT 10,
  requestNonce                   CHAINE D'OCTETS FACULTATIF }

ResponseTypes  ::= ENUMERATED {
  cached-only                (0),
  non-cached-only            (1),
  cached-and-non-cached      (2) }

RevocationInfoTypes ::= CHAINE DE BITS {
  fullCRLs                   (0),
  deltaCRLs                  (1),
  indirectCRLs               (2),
  oCSPResponses              (3) }

AuthPolicy ::= IDENTIFIANT D'OBJET

-- Identifiants de vérification SCVP

IDENTIFIANT D'OBJET id-stc ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5)
                                 pkix(7) 17 }
IDENTIFIANT D'OBJET id-stc-build-pkc-path ::= { id-stc 1 }
IDENTIFIANT D'OBJET id-stc-build-valid-pkc-path ::= { id-stc 2 }
IDENTIFIANT D'OBJET id-stc-build-status-checked-pkc-path ::= { id-stc 3 }
IDENTIFIANT D'OBJET id-stc-build-aa-path ::= { id-stc 4 }
IDENTIFIANT D'OBJET id-stc-build-valid-aa-path ::= { id-stc 5 }
IDENTIFIANT D'OBJET id-stc-build-status-checked-aa-path ::= { id-stc 6 }
IDENTIFIANT D'OBJET id-stc-status-check-ac-et-build-status-checked-aa-path ::= { id-stc 7 }

-- Identifiants de WantBack SCVP

IDENTIFIANT D'OBJET id-swb ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5)
                                 pkix(7) 18 }
IDENTIFIANT D'OBJET id-swb-pkc-best-cert-path      ::= { id-swb 1 }
IDENTIFIANT D'OBJET id-swb-pkc-revocation-info ::= { id-swb 2 }
IDENTIFIANT D'OBJET id-swb-pkc-public-key-info ::= { id-swb 4 }
IDENTIFIANT D'OBJET id-swb-aa-cert-path ::= { id-swb 5 }
IDENTIFIANT D'OBJET id-swb-aa-revocation-info ::= { id-swb 6 }
IDENTIFIANT D'OBJET id-swb-ac-revocation-info ::= { id-swb 7 }
IDENTIFIANT D'OBJET id-swb-relayed-responses  ::= { id-swb 9 }
IDENTIFIANT D'OBJET id-swb-pkc-cert ::= { id-swb 10}
IDENTIFIANT D'OBJET id-swb-ac-cert ::= { id-swb 11}
IDENTIFIANT D'OBJET id-swb-pkc-all-cert-paths ::= { id-swb 12}
IDENTIFIANT D'OBJET id-swb-pkc-ee-revocation-info ::= { id-swb 13}
IDENTIFIANT D'OBJET d-swb-pkc-CAs-revocation-info ::= { id-swb 14}

-- Identifiant de politique de validation et d'algorithme SCVP

IDENTIFIANT D'OBJET id-svp ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5)
                                 pkix(7) 19 }

IDENTIFIANT D'OBJET id-svp-defaultValPolicy ::= { id-svp 1 }

-- Identifiant d'algorithme de validation de base SCVP

IDENTIFIANT D'OBJET    id-svp-basicValAlg ::= { id-svp 3 }

-- Erreurs d'algorithme de validation de base SCVP
```

```
IDENTIFIANT D'OBJET id-bvae ::= id-svp-basicValAlg
IDENTIFIANT D'OBJET id-bvae-expired ::= { id-bvae 1 }
IDENTIFIANT D'OBJET id-bvae-not-yet-valid ::= { id-bvae 2 }
IDENTIFIANT D'OBJET id-bvae-wrongTrustAnchor ::= { id-bvae 3 }
IDENTIFIANT D'OBJET id-bvae-noValidCertPath ::= { id-bvae 4 }
IDENTIFIANT D'OBJET id-bvae-revoked ::= { id-bvae 5 }
IDENTIFIANT D'OBJET id-bvae-invalidKeyPurpose ::= { id-bvae 9 }
IDENTIFIANT D'OBJET id-bvae-invalidKeyUsage ::= { id-bvae 10 }
IDENTIFIANT D'OBJET id-bvae-invalidCertPolicy ::= { id-bvae 11 }

-- Identifiant d'algorithme de validation de nom SCVP

IDENTIFIANT D'OBJET id-svp-nameValAlg ::= { id-svp 2 }

-- Algorithme de comparaison de DN d'algorithme de validation de nom SCVP

IDENTIFIANT D'OBJET id-nva-dnCompAlg ::= { id-svp 4 }

-- Erreurs d'algorithme de validation de nom SCVP

IDENTIFIANT D'OBJET id-nvae ::= id-svp-nameValAlg
IDENTIFIANT D'OBJET id-nvae-name-mismatch ::= { id-nvae 1 }
IDENTIFIANT D'OBJET id-nvae-no-name ::= { id-nvae 2 }
IDENTIFIANT D'OBJET id-nvae-unknown-alg ::= { id-nvae 3 }
IDENTIFIANT D'OBJET id-nvae-bad-name ::= { id-nvae 4 }
IDENTIFIANT D'OBJET id-nvae-bad-name-type ::= { id-nvae 5 }
IDENTIFIANT D'OBJET id-nvae-mixed-names ::= { id-nvae 6 }

-- Identifiants d'objet de clé d'usage de clé étendu SCVP

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

IDENTIFIANT D'OBJET id-kp-scvpServer ::= { id-kp 15 }
IDENTIFIANT D'OBJET id-kp-scvpClient ::= { id-kp 16 }

FIN
```

9. Considérations sur la sécurité

Pour les considérations de sécurité spécifiques des formats de message de la syntaxe de message cryptographique, voir la [RFC3852]. Pour les considérations de sécurité spécifiques du processus de validation de chemin de certification PKI, voir la [RFC3280].

Un client qui fait confiance à la réponse d'un serveur pour la validation d'un certificat fait autant confiance à ce serveur qu'à son propre logiciel de validation. Cela signifie que si un attaquant compromet un serveur SCVP de confiance, l'attaquant peut changer le processus de validation pour chaque client qui s'appuie sur ce serveur. Donc, un serveur SCVP doit être protégé au moins aussi bien que les ancres de confiance auxquelles le serveur SCVP fait confiance.

Les clients DOIVENT vérifier que la réponse correspond à leur demande d'origine. Les clients ont besoin de s'assurer que le serveur a effectué les vérifications appropriées pour les certificats corrects sous la politique de validation demandée pour l'heure de validation spécifiée, et que la réponse inclut les wantBack demandés et satisfait les exigences de fraîcheur du client.

Quand la réponse SCVP est utilisée pour déterminer la validité d'un certificat, le client DOIT valider la signature numérique ou le MAC sur la réponse pour s'assurer que le serveur SCVP attendu l'a généré. Si le client ne vérifie pas la signature numérique ou le MAC de la réponse, une attaque par interposition pourrait tromper le client et lui faire croire à des réponses modifiées du serveur ou des réponses à des questions que le client n'a pas posées.

Si le client n'inclut pas un élément requestNonce, ou si le client ne vérifie pas que le requestNonce dans la réponse correspond à la valeur de la demande, un attaquant peut répéter des réponses antérieures provenant du serveur SCVP.

Si le serveur n'exige pas une forme d'autorisation (comme des demandes signées) un attaquant peut amener le serveur à répondre à des demandes arbitraires. De telles réponses peuvent donner à l'attaquant des informations sur les faiblesses du serveur ou sur la programmation des vérifications du serveur. Ces informations peuvent être précieuses pour une attaque future.

Si le serveur utilise l'élément serverContextInfo pour indiquer des états de serveur associés à un demandeur, les mises en œuvre doivent prendre des mesures appropriées contre les attaques de déni de service où un attaquant envoie beaucoup de demandes à la fois pour forcer le serveur à conserver beaucoup d'informations d'état.

SCVP n'inclut pas de mécanismes de protection de la confidentialité. Si la confidentialité est nécessaire, elle peut être réalisée avec un protocole de sécurité de couche inférieure comme TLS [RFC4346].

Si un client SCVP ne fonctionne pas sur un réseau avec une bonne protection physique, il doit s'assurer qu'il y a la protection de l'intégrité sur la paire demande-réponse SCVP. Le client peut s'assurer de l'intégrité en utilisant un transport protégé comme TLS. Il peut s'assurer de l'intégrité en utilisant des MAC ou des signatures numériques pour protéger individuellement les messages de demande et de réponse.

Si un client SCVP remplit le userPolicySet dans une demande avec une valeur autre que anyPolicy, mais n'établit pas le fanion requireExplicitPolicy, le serveur peut retourner une réponse affirmative pour des chemins qui ne satisfont aucune des politiques spécifiées. En général, quand un client remplit le userPolicySet dans une demande avec une valeur autre que anyPolicy, le fanion requireExplicitPolicy devrait aussi être établi. Cela garantit que tous les chemins valides satisfont au moins une des politiques demandées.

Dans SCVP, l'historique de validation d'un certificat retourne l'état connu du certificat à l'instant spécifié dans validationTime. Cela peut être utilisé pour démontrer une bonne gestion, mais ne fournit pas nécessairement les informations les plus complètes. Un certificat peut avoir été révoqué après l'instant spécifié dans validationTime, mais la notice de révocation peut spécifier une date d'invalidité qui précède le validationTime. Le serveur SCVP va fournir une réponse affirmative même si les informations disponibles les plus à jour indiquent que le certificat ne devrait plus être de confiance à ce moment. Les clients SCVP peuvent souhaiter spécifier un validationTime plus tard que l'instant actuel pour atténuer ce risque.

10. Considérations relatives à l'IANA

Les détails des demandes et réponses SCVP sont communiqués en utilisant des identifiants d'objet (OID). Les objets sont définis dans un arc délégué par l'IANA au groupe de travail PKIX. Le présent document inclut aussi quatre enregistrements de type MIME dans l'Appendice A. Aucune autre action de l'IANA n'est nécessaire pour le présent document ou ses mises à jour prévues.

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)

[RFC2560] M. Myers, R. Ankney, A. Malpani, S. Galperin et C. Adams, "Protocole d'état de certificat en ligne d'infrastructure de clé publique X.509 pour l'Internet - OCSP", juin 1999. *(P.S.) (Remplacée par* RFC6960)

[RFC2616] R. Fielding et autres, "Protocole de transfert hypertexte -- HTTP/1.1", juin 1999. *(D.S., MàJ par* 2817, 6585)

[RFC2634] P. Hoffman, éd., "Services de sécurité améliorés pour S/MIME", juin 1999. (*MàJ par* RFC5035) *(P.S.)*

[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)

[RFC3281] S. Farrell et R. Housley, "Profil de certificat d'attribut Internet pour l'autorisation", avril 2002. *(Obsolète, voir* RFC5755)

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

[RFC3850] B. Ramsdell, éd., "Traitement de certificat d'extensions multi-objets/sécurisées de messagerie Internet (S/MIME) version 3.1", juillet 2004. *(P.S.) (Remplacée par* RFC5750)

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

[RFC4055] J. Schaad et autres, "Algorithmes et identifiants supplémentaires pour la cryptographie RSA à utiliser dans le profil de certificat d'infrastructure de clé publique X.509 et de liste de révocation de certificat (CRL) pour l'Internet", juin 2005.

[RFC4306] C. Kaufman, "Protocole d'échange de clés sur Internet (IKEv2)", décembre 2005. *(Obsolète, voir la* RFC5996)

11.2 Références pour information

[RFC3379] D. Pinkas, R. Housley, "Exigences pour le protocole de validation de chemin délégué et de découverte de chemin délégué," septembre 2002. *(Information)*

[RFC3526] T. Kivinen et M. Kojo, "Groupes supplémentaires d'exponentiation modulaire (MODP) Diffie-Hellman pour l'échange de clés Internet (IKE)", mai 2003.

[RFC4346] T. Dierks et E. Rescorla, "Protocole de sécurité de la couche Transport (TLS) version 1.1", avril 2006. *(Remplace* RFC2246 *; Remplacée par* RFC5246 *; MàJ par* RFC4366, 4680, 4681, 5746, 6176, 7465, 7507, 7919)

12. Remerciements

Le débat animé au sein du groupe de travail PKIX a eu un impact significatif sur ce protocole. Des remerciements particuliers sont dus aux personnes suivantes pour leurs contributions à ce document et qui l'ont largement amélioré : Paul Hoffman, Phillip Hallam-Baker, Mike Myers, Frank Balluffi, Ameya Talwalkar, John Thielens, Peter Sylvester, Yuriy Dzambasow, Sean P. Turner, Wen-Cheng Wang, Francis Dupont, Dave Engberg, Faisal Maqsood.

Merci aussi au président du groupe de travail Steve Kent pour son aide et son soutien.

Appendice A. Enregistrements de types de prise en charge MIME

Quatre types d'enregistrement MIME sont fournis dans cet appendice.

A.1 application/scvp-cv-request

Pour : ietf-types@iana.org
Sujet : enregistrement du type MIME de prise en charge application/scvp-cv-request
Nom du type MIME de prise en charge : application
Nom de sous-type MIME : scvp-cv-request
Paramètres exigés : aucun
Paramètres facultatifs : aucun
Considérations de codage : binaire

Considérations de sécurité : porte une demande d'informations. Cette demande peut facultativement être protégée cryptographiquement.
Considérations d'interopérabilité : aucune
Spécification publiée : RFC 5055
Applications qui utilisent ce type de prise en charge : les clients SCVP qui envoient des demandes de validation de certificat
Informations supplémentaires :
Numéro magique : aucun
Extension de fichier : .SCQ
Code de type de fichier Macintosh : aucun
Adresse de la personne et de messagerie à contacter pour plus d'informations : Ambarish Malpani <ambarish@yahoo.com>
Usage prévu : COMMUN
Restrictions d'usage : ce type de prise en charge peut être utilisé avec tout protocole qui peut transporter des objets signés numériquement.
Auteur : Ambarish Malpani <ambarish@yahoo.com>
Contrôleur des changements : IESG

A.2 application/scvp-cv-response

Pour : ietf-types@iana.org
Sujet : enregistrement du type MIME de prise en charge application/scvp-cv-response
Nom du type MIME de prise en charge : application
Nom de sous-type MIME : scvp-cv-response
Paramètres exigés : aucun
Paramètres facultatifs : aucun
Considérations de codage : binaire
Considérations de sécurité : le client peut exiger que cette réponse soit protégée cryptographiquement, ou peut choisir d'utiliser un mécanisme de transport sûr. Les réponses de DPD peuvent être non protégées, mais le client valide les informations fournies dans la demande.
Considérations d'interopérabilité : aucune
Spécification publiée : RFC 5055
Applications qui utilisent ce type de prise en charge : les serveurs SCVP qui répondent aux demandes de validation de certificat
Informations supplémentaires :
Numéro magique : aucun
Extension de fichier : .SCQ
Code de type de fichier Macintosh : aucun
Adresse de la personne et de messagerie à contacter pour plus d'informations : Ambarish Malpani <ambarish@yahoo.com>
Usage prévu : COMMUN
Restrictions d'usage : ce type de prise en charge peut être utilisé avec tout protocole qui peut transporter des objets signés numériquement.
Auteur : Ambarish Malpani <ambarish@yahoo.com>
Contrôleur des changements : IESG

A.3 application/scvp-vp-request

Pour : ietf-types@iana.org
Sujet : enregistrement du type MIME de prise en charge application/scvp-vp-request
Nom du type MIME de prise en charge : application
Nom de sous-type MIME : scvp-vp-request
Paramètres exigés : aucun
Paramètres facultatifs : aucun
Considérations de codage : binaire
Considérations de sécurité : porte une demande d'information.
Considérations d'interopérabilité : aucune
Spécification publiée : RFC 5055
Applications qui utilisent ce type de prise en charge : les clients SCVP qui envoient des demandes de politique de validation.
Informations supplémentaires :

Numéro magique : aucun
Extension de fichier : .SCQ
Code de type de fichier Macintosh : aucun
Adresse de la personne et de messagerie à contacter pour plus d'informations : Ambarish Malpani <ambarish@yahoo.com>
Usage prévu : COMMUN
Restrictions d'usage : ce type de prise en charge peut être utilisé avec tout protocole qui peut transporter des objets signés numériquement.
Auteur : Ambarish Malpani <ambarish@yahoo.com>
Contrôleur des changements : IESG

A.4 application/scvp-vp-response

Pour : ietf-types@iana.org
Sujet : enregistrement du type MIME de prise en charge application/scvp-vp-response
Nom du type MIME de prise en charge : application
Nom de sous-type MIME : scvp-vp-response
Paramètres exigés : aucun
Paramètres facultatifs : aucun
Considérations de codage : binaire
Considérations de sécurité : aucune
Considérations d'interopérabilité : aucune
Spécification publiée : RFC 5055
Applications qui utilisent ce type de prise en charge : les serveurs SCVP qui répondent aux demandes de politique de validation.
Informations supplémentaires :
Numéro magique : aucun
Extension de fichier : .SPP
Code de type de fichier Macintosh : aucun
Adresse de la personne et de messagerie à contacter pour plus d'informations : Ambarish Malpani <ambarish@yahoo.com>
Usage prévu : COMMUN
Restrictions d'usage : ce type de prise en charge peut être utilisé avec tout protocole qui peut transporter des objets signés numériquement.
Auteur : Ambarish Malpani <ambarish@yahoo.com>
Contrôleur des changements : IESG

Appendice B. SCVP sur HTTP

Cet Appendice décrit les conventions de formatage et de transport pour la demande et la réponse SCVP quand elles sont portées par HTTP.

Pour que les clients et les serveurs SCVP qui utilisent HTTP interopèrent, les règles suivantes s'appliquent :
- Les clients DOIVENT utiliser la méthode POST pour soumettre leurs demandes.
- Les serveurs DOIVENT utiliser le code de réponse 200 pour les réponses de succès.
- Les clients PEUVENT tenter d'envoyer des demandes HTTPS en utilisant TLS 1.0 ou plus récent, bien que les serveurs ne soient pas obligés de prendre en charge TLS.
- Les serveurs NE DOIVENT PAS supposer que le client prend en charge un type d'authentification HTTP tel que des mouchards, l'authentification de base, ou l'authentification par résumé.
- Les clients et les serveurs sont supposés suivre les autres règles et restrictions de la [RFC2616]. Noter que certaines de ces règles sont pour des méthodes HTTP autres que POST ; en clair, seules les règles qui s'appliquent à POST sont pertinentes pour cette spécification.

B.1 Demande SCVP

Une demande SCVP qui utilise la méthode POST est construite comme suit :

L'en-tête Content-Type DOIT avoir la valeur "application/scvp-cv-request".

Le corps du message est la valeur binaire du codage en DER de la CVRequest, enveloppée dans un corps de CMS comme

décrit à la Section 3.

B.2 Réponse SCVP

Une réponse SCVP fondée sur HTTP est composée des en-têtes HTTP appropriés, suivis par la valeur binaire du codage en BER de la CVResponse, enveloppée dans un corps de CMS comme décrit à la Section 4.

L'en-tête Content-Type DOIT avoir la valeur "application/scvp-cv-response".

B.3 Demande de politique SCVP

Une demande SCVP qui utilise la méthode POST est construite comme suit :

L'en-tête Content-Type DOIT avoir la valeur "application/scvp-vp-request".

Le corps du message est la valeur binaire du codage en BER de la ValPolRequest, enveloppée dans un corps de CMS comme décrit à la Section 5.

B.4 Réponse de politique SCVP

Une réponse de politique SCVP fondée sur HTTP est composée des en-têtes HTTP appropriés, suivis par la valeur binaire du codage en DER de la ValPolResponse, enveloppée dans un corps de CMS comme décrit à la Section 6. L'en-tête Content-Type DOIT avoir la valeur "application/scvp-vp-response".

Adresse des auteurs

Trevor Freeman
Microsoft Corporation,
One Microsoft Way
Redmond, WA 98052
USA
mél : trevorf@microsoft.com

Russell Housley
Vigil Security, LLC
918 Spring Knoll Drive
Herndon, VA 20170
USA
mél : housley@vigilsec.com

Ambarish Malpani
Malpani Consulting Services
mél : ambarish@yahoo.com

David Cooper
National Institute of Standards et Technology
100 Bureau Drive, Mail Stop 8930
Gaithersburg, MD 20899-8930
mél : david.cooper@nist.gov

Tim Polk
National Institute of Standards et Technology
100 Bureau Drive, Mail Stop 8930
Gaithersburg, MD 20899-8930
USA
mél : wpolk@nist.gov

Déclaration complète de droits de reproduction

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.