rfc5755

Profil de certificat d'attribut Internet pour l'autorisation

Introduction

Les certificats à clé publique X.509 (PKCs) lien une identité et une clé publique. Un certificat d'attribut (AC) est une structure similaire à PKC ; la principale différence est que AC ne contient pas de clé publique. Un AC peut contenir des attributs qui spécifie l'appartenance à un groupe, un rôle, habilitation, ou d'autres informations d'autorisation associés avec le titulaire de l'AC.

La syntaxe pour AC est définie dans Recommendation X.509, rendant le terme "X.509 Certificate' ambigus.

Certaines personnes confondent constamment PKCs et ACs. Un analogie peut rendre la distinction plus claire. An PKC peut être considéré comme un passeport : il identifie le titulaire, tend à durer longtemps, et ne devrait pas être trivial à obtenir. Un AC est plus comme un visa d'entrée : il est typiquement utilisé par une autorité différente et ne dure pas très longtemps.

Les informations d'autorisation peuvent être placés dans une extension PKC ou placé dans un certificat d'attribut séparé. Le placement des informations d'autorisation dans les PKCs est généralement indésirable pour 2 raisons. D'abord, les informations d'autorisation n'ont pas la même durée de vie que la liaison de l'identité et de la clé publique. Ensuite, le fournisseur PKC n'a généralement pas autorité pour les informations d'autorisation.

Pour ces raisons, il est souvent meilleur de séparer les informations d'autorisation des PKC. Les informations d'autorisation doivent également être liées a une identité. Un AC fournis cette liaison ; c'est simplement une identité signé ou certifiée numériquement et un jeu d'attributs.

Un AC peut être utilisé avec divers services de sécurité, incluant le contrôle d'accès, authentification de l'origine des données, et la non-répudiation.

les PKCs peuvent fournir une identité à des fonctions de contrôle d'accès. Cependant, dans de nombreux contextes, l'identité n'est pas le critère qui est utilisé pour les décisions de contrôle d'accès, mais plutôt le rôle ou l'appartenance à un groupe. De tels schémas de contrôles d'accès sont appelé RBAC.

En créant des décisions de contrôle d'accès basé sur un AC, une fonction de décision de contrôle d'accès peut nécessiter de s'assurer que le titulaire de l'AC approprié est l'entité qui a demandé l'accès. Une manière de lier la demande ou l'identité et l'AC peut être accomplie par l'inclusion d'une référence à un PKC dans l'AC et l'utilisation de la clé privée correspondante au PKC pour l'authentification dans la demande d'accès.

Les AC peuvent également être utilisés dans le contexte de service d'authentification de l'origine des données et de non-répudiation. Dans ces contextes, les attributs contenus dans l'AC fournis des informations additionnels sur l'identité du signataire. Cet information peut être utilisée pour s'assurer que l'identité est autorisée à signer la donnée. Ce genre de vérification dépend soit du contexte dans lequel la donnée est échangée soit sur la donnée qui a été signée numériquement.

Délégation de chemin d'AC

Le standard X.509 définis l'autorisation comme le "transport de privilège d'une entité qui détient ce privilège, à une autre identité.". Un AC est un mécanisme d'autorisation.

Une séquence ordonnée d'AC pourrait être utilisée pour vérifier l'authenticité des privilège du déclarant. De cette manière, les chaînes et chemins d'AC pourraient être employés pour déléguer l'autorisation.

Vu que l'administration est le traitement associé avec de telles chaînes AC est complexe et que l'utilisation des AC dans l'Internet aujourd'hui est limitée, il est recommandé que les implémentations de cette spécification n'utilisent pas les chaînes d'AC. D'autres futures spécifications peuvent adresser l'utilisation de chaînes d'AC. Cette spécification s'occupe de cas simples, où une autorité gère tous les AC pour un jeu d'attributs particuliers. Cependant, cette simplification n'exclus par l'utilisation de plusieurs autorités différentes, chacune d'entre elles gérant un jeu d'attributs différents. Par exemple, l'appartenance à un groupe peut être inclus dans un AC fournis par une autorité, et les habilitations de sécurité peuvent être inclus dans un autre AC par une autre autorité.

Cela signifie que les implémentations conformes doivent seulement être capable de traiter un simple AC à la fois. Le traitement de plus d'un AC, un après l'autre, peut être nécessaire. Noter cependant, que la validation d'un AC peut nécessiter une validation d'une chaîne de PKCs.

Distribution de certificat d'attributs

Comme discuté plus haut, les AC fournissent un mécanisme pour fournir de manière sécurisé des informations d'autorisation pour, par exemple, des fonctions de décision de contrôle. Cependant, il y a un nombre de chemins de communication possible pour les AC.

Dans certains environnements, il est possible pour un client de "pousser" un AC à un serveur. Cela signifie qu'il n'y a pas de nouvelle connexion entre le client et le serveur. Ce la signifie également qu'aucune charge de recherche n'est imposée sur les serveurs, ce qui améliore les performances et le vérificateur d'AC est seulement présenté avec ce que l'on a besoin de connaître. Le modèle "push" est préférable spécialement dans les cas inter-domaines où les droits des client devraient être assignés dans le domaine du client.

Dans d'autres cas, il est préférable pour un client de simplement s'authentifier auprès du serveur et de lui demander ("pull") l'AC du client depuis un fournisseur d'AC. Un bénéfice majeur est du modèle "pull" est qu'il peut être implémenté sans changements sur le client ou dans le protocole. Le modèle "pull" est préférable spécialement pour les cas inter-domaines où les droits du client devraient être assignés dans le domaine du serveur.

Il y a un certain nombre d'échanges possibles impliquant 3 entités : Le client, le serveur, et le fournisseur d'AC. En plus, un service d'annuaire ou d'autres annuaire pour la récupération d'AC peuvent être supportés.

La Figure 1 montre un vue abstraite des échanges qui peuvent se produire. Ce profile ne spécifie pas de protocole pour ces échanges :

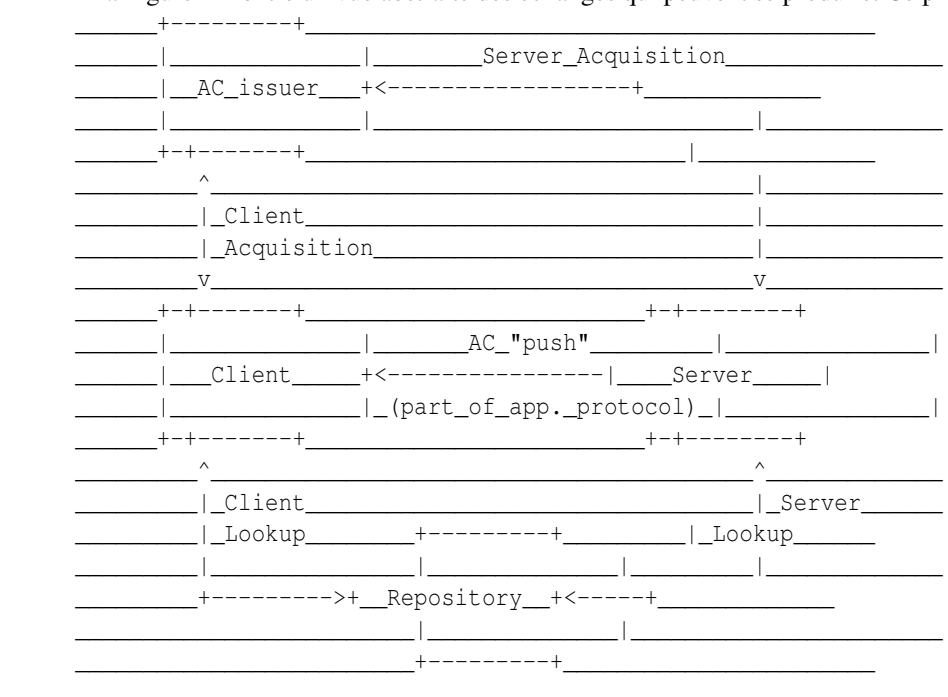

Figure_1:_AC_Exchanges

Terminologie

Par simplicité, on utilise les termes client et serveur dans cette spécification. Cela n'est pas prévu pour indiquer que les AC sont seulement utilisé dans des environnements client/serveur. Par exemple, les AC peuvent être utilisé dans un contexte S/MIME v3.2, où le MUA serait à la fois client et serveur dans le sens des termes utilisés ici.

AA Attribute Autority, l'entité qui fournis l'AC, synonyme dans cette spécification avec "fournisseur de CA"

AC Attribute Certificate

AC user Une entité qui parse ou traite un AC

AC verifier Une entité qui vérifie la validité d'un AC et utilise le résultat

AC issuer L'entité qui signe l'AC : synonyme de AA

Client L'entité qui demande l'action pour laquelle les vérifications d'autorisation doivent êtes faites

Proxying Dans cette spécification, le proxying est utilisé pour signifier la situation où un serveur d'application agit comme application cliente à la demande de l'utilisateur.

PKC Public Key Certificates - utilise les certificats de type ASN.1 définis dans X.509 et profilé dans la rfc5280.

Server L'entité qui nécessite qu'une vérification d'autorisation soit faite.

Pré-requis

Ce profile AC à les pré-requis suivants :

1. Support pour les AC à durée de vie courte et longue. Une période de validité courte typique peut se mesurer en heures, en mois pour les PKC. Les périodes de validité courtes permettent aux AC d'être utiles sans mécanisme de révocation.
2. Les fournisseurs d'AC devraient être capable de définir leur propres types d'attributs à utiliser dans les domaines fermés.
3. Certains types d'attributs standard, qui peuvent être contenus dans les AC devraient être définis. Par exemple "access identity", "group", "role", "clearance", "audit identity", et "charging identity".
4. Les type d'attributs standard devraient être définis d'une manière qui permette à un AC verifier de distinguer l'utilisation de mêmes attributs dans des domaines différents. Par exemple, "Administrators group" comme définis par "Baltimore" et "Administrators group" définis par "SPYRUS" devraient être facilement distincts.
5. Il devrait être possible de cibler un AC à un ou un petit nombre de serveurs. Cela signifie qu'un serveur non-ciblé sera rejeté par les décisions d'autorisation d'AC.
6. Les AC devaient être définis pour qu'il puissent être soit "pushed" par le client au serveur, ou "pulled" par le serveur depuis un dépôt ou un service réseaux, incluant un fournisseur d'AC online.

Profile de certificat d'attribut

Les AC peuvent être utilisés dans un large éventail d'applications et d'environnements, couvrant un large spectre de besoins opérationnels et d'assurances. Le but de ce document est d'établir une base commune pour des applications génériques nécessitant une grande interopérabilité et des besoins spéciaux limités. En particulier, l'accent sera mis sur le support de l'utilisation de certificats d'attributs pour les messages électroniques, IPsec, et les applications www.

Cette section présente un profile pour les AC qui va favoriser l'interopérabilité. Cette section définis également certaines extensions pour la communauté internet.

Alors que les documents ISO/IEC/ITU utilisent la version 1993 (ou ultérieur) de ASN.1, ce document utilise la syntaxe ASN.1 1988, comme pour les PKCs.

Définition de certificat d'attribut X.509

X.509 contient la définition d'un AC donné ci-dessous. Tous les types qui ne sont pas définis dans ce document peuvent être trouvés dans la rfc 5280.

```
AttributeCertificate ::= SEQUENCE {
 acinfo AttributeCertificateInfo,
 signatureAlgorithm AlgorithmIdentifier,
 signatureValue BIT STRING
}

AttributeCertificateInfo ::= SEQUENCE {
 version AttCertVersion, - version is v2
 holder Holder,
 issuer AttCertIssuer,
 signature AlgorithmIdentifier,
 serialNumber CertificateSerialNumber,
 attrCertValidityPeriod AttCertValidityPeriod,
 attributes SEQUENCE OF Attribute,
 issuerUniqueID UniqueIdentifier OPTIONAL,
 extensions Extensions OPTIONAL
}

AttCertVersion ::= INTEGER { v2(1) }

Holder ::= SEQUENCE {
 baseCertificateID [0] IssuerSerial OPTIONAL,
  - le issuer et serial number du titulaire du certificat à clé publique
 entityName [1] GeneralNames OPTIONAL,
  - le nom du demandeur ou du rôle
 objectDigestInfo [2] ObjectDigestInfo OPTIONAL
  - Utilisé pour authentifier directement le titulaire, par exemple, un exécutable.
}

ObjectDigestInfo ::= SEQUENCE {
 digestedObjectType ENUMERATED {
  publicKey (0),
  publicKeyCert (1),
  otherObjectTypes (2) },
 - otherObjectTypes ne doit pas être utilisé dans ce profile
 otherObjectTypeID OBJECT IDENTIFIER OPTIONAL,
 digestAlgorithm AlgorithmIdentifier,
 objectDigest BIT STRING
}

AttCertIssuer ::= CHOICE {
 v1Form GeneralNames, - ne doit pas être utilisé dans ce profile
 v2Form [0] V2Form - v2 only
}

V2Form ::= SEQUENCE {
 issuerName GeneralNames OPTIONAL,
 baseCertificateID [0] IssuerSerial OPTIONAL,
 objectDigestInfo [1] ObjectDigestInfo OPTIONAL
```

```
  - issuerName doit être présent dans ce profile
  - baseCertificateID et objectDigestInfo ne doivent pas être présents dans ce profile
}

IssuerSerial ::= SEQUENCE {
 issuer GeneralNames,
 serial CertificateSerialNumber,
 issuerUID UniqueIdentifier OPTIONAL
}

AttCertValidityPeriod ::= SEQUENCE {
 notBeforeTime GeneralizedTime,
 notAfterTime GeneralizedTime
}
```

Bien que la syntaxe d'attribut soit définis dans la rfc 5280, on répète la définition ici par commodité :

```
Attribute ::= SEQUENCE {
 type AttributeType,
 values SET OF AttributeValue
  - Au moins une valeur est requise
}

AttributeType ::= OBJECT IDENTIFIER

AttributeValue ::= ANY DEFINED BY AttributeType
```

Les implémenteurs devraient noter que l'encodage DER des valeurs de jeu nécessitent d'ordonner l'encodage des valeurs. Bien que cette question se pose avec les dn, et doit être géré avec l'implémentation rfc 5280, il est plus important dans ce contexte, vu que les valeurs multiples sont plus courantes dans les AC.

Profile des champs standards

GeneralName offre une grande flexibilité. Pour parvenir à l'interopérabilité, en dépit de cette souplesse, ce profile impose des contraintes à l'utilisation de GeneralName.

Les implémentations doivent être capable de supporter dNSName, directoryName, uniformResourceIdentifier, et iPAddress. C'est compatible avec les pre-requis de GeneralName dans la rfc 5280. Les implémentations doivraient également supporter SRVName, et ne doivent pas utiliser x400Address, ediPartyName et registeredID.

Les implémentations peuvent utiliser otherName pour transmettre les formes de nom définis dans les standards Internet. Par exemple, le format kerberos des noms peut être encodé dans otherName, en utilisant l'OID du nom principal kerberos v5 et une séquence de domaine et du PrincipalName.

Version

Le champ version doit avoir la valeur v2. Le champ version est présent dans l'encodage DER.

Note : cette version n'est pas compatible avec les précédentes définition des certificats d'attribut du standard X.509-1997, mais est compatible avec la définition v2 X.509-2000.

Holder

Le champ Holder est une séquence permettant 2 syntaxes optionnelles différentes : baseCertificateID, entityName, et objectDigestInfo. Où seul une option est présente, la signification du champ Holder est claire.

Cependant, quand plus d'une option est utilisée, il y a une confusion possible, laquelle est normative, laquelle est une allusion, etc. Vu que la position correcte n'est pas claire dans X.509-2000, cette spécification recommande que seule une de ces options soit utilisée dans un AC donné.

Pour tout environnement où l'AC est passée dans un message authentifié ou une session et où l'authentification est basée sur l'utilisation d'un PKC X.509, le champ Holder devrait être le baseCertificateID.

Avec l'option baseCertificateID, le serialNumber et issuer du PKC du titulaire doit être identique au champ Holder de l'AC. Le fournisseur PKC doit avoir un dn non vide qui doit être présent comme simple valeure de la construction holder.baseCertificateID.issuer dans le champ directoryName. Le champ holder.baseCertificateID.issuerUID de l'AC doit seulement être utilisé si le PKC du titulaire contient un champ issuerUniqueID. Si le champs holder.baseCertificateID.issuerUID de l'AC et le champ issuerUniqueID du PKC sont présents, la même valeur doit être présente dans ces 2 champs. Donc, baseCertificateID est seulement utilisable avec les profiles PKC qui mandatent que le champ issuer du PKC contient une valeur dn non-vide.

Note : un dn vide est un dn où la séquence de dn relatifs à une longueur de 0. Dans un encodage DER, il a la valeur '3000'H.

Si le champ Holder utilise l'option entityName et que l'authentification sous-jacent est basée sur un PKC, entityName doit être le même que le champ subject du PKC ou une des valeurs de subjectAltName du PKC (si présent). Noter que la rfc 5280 mandate que l'extension subjectAltName soit présente si le sujet du PKC est un dn vide.

Dans tous les cas où le champ Holder utilise l'option entityName, seul un nom devrait être présent.

Les implémentations se conformant à ce profile ne sont pas obligés de supporter l'utilisation du champ objectDigest.

Tout protocole se conformant à ce profile devrait spécifier quel option de titulaire de l'AC doit être utilisé et comment cela s'intègre avec les schémas d'authentification supportés définis dans ce protocole.

Issuer

Les AC se conformant à ce profile doivent utiliser le choix v2Form, qui doit contenir un et un seul GeneralName dans le issuerName, qui doit contenir un dn non-vide dans le champ directoryName. Cela signifie que tous les fournisseurs d'AC doivent avoir des noms distincts non-vide. Les AC se conformant à ce profile doivent omettre les champs baseCertificateID et objectDigestInfo.

Une des raisons d'utiliser v2Form contenant seulement un issuerName et que cela le fournisseur n'a pas à connaître quel PKC l'AC verifier va utiliser. Utiliser le champ baseCertificateID pour référencer le fournisseur d'AC signifie que l'AC verifier doit faire confiance au PKC que le fournisseur d'AC choisis pour lui à la création de l'AC.

Signature

Contient l'identifiant d'algorithme utilisé pour valider la signature AC. Doit être un des algorithmes de signature définis dans les rfc3279, rfc4055, rfc5480, et rfc5756, ou définis dans une mise à jour approuvée par l'ietf de ces rfc. Les implémentations conformes doivent honorer toutes les déclarations MUST/SHOULD/MAY des algorithmes de signature.

SerialNumber

Pour tout AC conforme, la paire issuer/serialNumber doit former une combinaison unique, même si les AC ont une durée de vie très courte. Les fournisseurs d'AC doivent s'assure que le serialNumber est un nombre positif, qui est, le bit de signe dans l'encodage DER de la valeur integer doit être 0. Cela peut être fait en ajoutant un '00'H gauche si nécessaire. Cela supprime une ambiguïté dans le mappage entre une chaîne d'octets et une valeur entière.

En donnant un timing et une unicité requise, les numéros de série peuvent contenir des entiers longs. Les utilisateurs d'AC doivent être capable de manipuler les valeurs de serialNumber supérieurs à 4 octets. Les AC conformes ne doivent pas contenir de valeurs serialNumber supérieurs à 20 octets.

Il n'y a pas de pré-requis pour que le numéro de série utilisé par un fournisseur d'AC suive un ordre particulier. En particulier, ils nécessitent de ne pas être incrémentés monotoniquement avec le temp. Chaque fournisseur d'AC doit s'assurer que chaque AC qu'il fournis contient un numéro de série unique.

Période de validité

Le champ attrCertValidityPeriod spécifie la période pour laquelle le fournisseur d'AC certifie que le lien entre le titulaire et les champs d'attributs sont valides.

Le type GeneralizedTime est un type ASN.1 standard pour représenter le temps. Ce champ peut optionnellement inclure une représentation du temps différentiel entre la zone de temps local et GMT.

Pour ce profile, les valeurs GeneralizedTime doivent être exprimés en UTC et doivent inclure les secondes (ex : YYYYMMDDHHMMSSZ), même quand le nombre de seconde est 0. Les valeurs GeneralizedTime ne doivent pas inclure les fractions de secondes.

Les utilisateurs d'AC doivent être capable de manipuler un AC que, au moment du traitement, a des parties de sa période de validité ou toute sa période de validité dans le passé ou dans le future. C'est valide pour certaines applications, tels que les backups.

Attributes

Le champ attributes donne des informations sur le titulaire de l'AC. Quand l'AC est utilisé pour l'autorisation, il contient souvent un jeu de privilèges.

Le champ attributes contient une séquence d'attributs. chaque attribut contient le type de l'attribut et un jeu de valeurs. Pour un AC donné, chaque identifiant d'object AttributeType dans la séquence doit être unique, mais peut être multi-valué.

Les utilisateurs d'AC doivent être capable de manipuler plusieurs valeurs pour tous les types d'attributs. Un AC doit contenir au moins un attribut, c'est à dire une séquence d'Attributes de longueur non-zéro.

Issuer Unique Identifier

Ce champ ne doit pas être utilisé à moins qu'il est également utilisé dans le PKC du fournisseur d'AC, auquel cas il doit être utilisé. Noter que la rfc5280 indique que ce champ ne devrait pas être utilisé par les autorités de certification conformes, mais ces applications devraient être capable de gérer les PKCs contenant ce champ.

Extensions

Le champ extensions donne généralement des informations sur l'AC, en opposé aux informations sur le titulaire de l'AC.

Un AC qui n'a pas d'extensions est conforme avec ce profile. Cependant les sections suivantes définissent des extensions qui peuvent être utilisé avec ce profile, et indique s'ils sont à marquer critiques. Si une autre extension critique est utilisé, l'AC n'est pas conforme à ce profile. Cependant, si une autre extension non-critique est utilisée, l'AC est conforme à ce profile.

Les extensions définie pour les AC fournissent des méthodes pour associer les attributs additionnels avec les titulaires. Ce profile permet aux communautés de définir des extensions privées pour les informations unique à ces communautés. Chaque extension dans un AC peut être désigné comme critique ou non-critique. Un system utilisant les AC doit rejeter un AC s'il rencontre une extension critique qu'il ne reconnaît pas ; cependant, une extension non-critique peut être ignorée s'il ne la reconnaît pas.

Audit Identity

Dans certaines circonstances, il est requis (par ex : protection des données) que chemins d'audit ne contiennent pas d'enregistrement qui identifie directement les individus. Cette circonstance peut rendre le champ Holder inutilisable dans les chemins d'audit.

Pour permettre de tels cas, un AC peut contenir une extension d'identité d'audit. Idéalement, il devrait être impossible de dériver l'identité du titulaire depuis la valeur de l'identité d'audit sans la coopération du fournisseur d'AC.

La valeur de l'identité d'audit, avec le issuer/serial, devrait être utilisé dans un but d'audit/logging. Si la valeur de l'identité d'audit est bien choisis, un serveur/administrateur de service peut utiliser les traces d'audit pour tracer le comportement du titulaire de l'AC sans être capable d'identifier ce titulaire.

Le serveur/administrateur de service en combinaison avec le fournisseur d'AC doit être capable d'identifier le titulaire de l'AC dans les cas où des problèmes sont détectés. Cela signifie que le fournisseur d'AC doit être capable de déterminer l'identité actuelle du titulaire de l'AC depuis l'identité d'audit.

Bien évidemment, l'audit pourrait être basée sur la paire issuer/serial ; cependant, cette méthode ne permet pas de tracer le même titulaire avec plusieurs AC. Donc, une identité d'audit n'est utile que si elle dure plus longtemps que la durée de vie typique d'un AC. L'audit pourrait également être basée sur le issuer/serial du PKC du titulaire de l'AC ; cependant, cela permet souvent au serveur/administrateur de service d'identifier le titulaire de l'AC.

Un AC verifier peut sinon utiliser holder ou d'autres valeur identifiant pour un but d'audit, cette extension doit être critique quand elle est utilisée.

Les protocoles qui utilisent les AC vont souvent exposer l'identité du titulaire. Dans de tels cas, une identité d'audit opaque n'utilise pas d'AC anonyme ; il s'assure simplement que les traces d'audit ne contiennent pas d'information d'identification.

La valeur d'une identité d'audit doit être plus longue que 0 octet. La valeur d'une identité d'audit ne doit pas être plus longue que 20 octets.

```
name id-pe-ac-auditIdentity
OID { id-pe 4 }
syntax OCTET STRING
criticality MUST be TRUE
```

AC Targeting

Pour cibler un AC, l'extension d'information de cible, importé depuis X.509-2000, peut être utilisé pour spécifier un nombre de

serveurs/services. Le but est que l'AC devrait seulement être utilisable sur les serveurs/services spécifiés. Un AC verifier (honnête) qui n'est pas parmi les serveurs/services nommés doit rejeter l'AC.

Si cette extension n'est pas présente, l'AC n'est pas ciblée et peut être accepté par n'importe quel serveur. Dans ce profile, les informations de cible consiste simplement d'une liste de cibles nommés, ou de groupes.

La syntaxe suivante est utilisée pour représenter les informations de cible :

```
Targets ::= SEQUENCE OF Target

Target ::= CHOICE {
 targetName [0] GeneralName,
 targetGroup [1] GeneralName,
 targetCert [2] TargetCert
}

TargetCert ::= SEQUENCE {
 targetCertificate IssuerSerial,
 targetName GeneralName OPTIONAL,
 certDigestInfo ObjectDigestInfo OPTIONAL
}
```

Le choix targetCert dans la structure Target est seulement présente pour permettre de futures compatibilités avec X.509-2000 et ne doit pas être utilisé.

Les vérifications de cible réussissent si le serveur courant (le destinataire) est un des champs targetName dans la séquence Targets, ou si le serveur courant est un membre d'un des champs targetGroup dans la sequence Targets. Dans ce cas, le serveur courant est dit "correspondre" à l'extension de ciblage.

La manière de déterminer si la cible est dans un targetGroup n'est pas déterminés ici. Il est assumé que la cible donnée connaît les noms des targetGroups pour lequel il appartient ou peut déterminer ses membres. Par exemple, le targetGroup spécifie un domaine DNS, et l'AC verifier connaît le domaine DNS auquel il appartient. Pour un autre exemple, targetGroup spécifie "PRINTERS" et l'AC verifier sait s'il est ou non une imprimante ou un serveur d'impression.

Note : X.509-2000 définis la syntaxe d'extension comme une séquence de target. Les implémentations de fournisseur d'AC conformes doivent seulement produire un élément Targets. Les utilisateurs d'AC doivent être capable d'accepter une sequence de Targets. Si plus d'un élément Targets est trouvé dans l'AC, l'extension doit être traitée comme si tous les éléments Target avaient été trouvés dans un élément Targets.

```
name id-ce-targetInformation
OID { id-ce 55 }
syntax SEQUENCE OF Targets
criticality MUST be TRUE
```

Authority Key Identifier

L'extension authorityKeyIdentifier, comme profilé dans la rfc 5280, peut être utilisée pour assister l'AC verifier dans la vérification de la signature de l'AC. La description de la rfc 5280 devrait être lue comme si CA signifiait AC issuer. Comme avec les PKC, cette extension devrait être incluse dans les AC.

Note : Un AC, où le champ issuer utilisant le choix baseCertificateID, ne nécessite pas l'extension authorityKeyIdentifier, vu qu'il est explicitement lié à la clé dans le certificat référé. Cependant, l'AC doit utiliser la v2Form avec le choix issuerName, cette duplication ne se produit pas.

```
name id-ce-authorityKeyIdentifier
```

```
OID { id-ce 35 }
syntax AuthorityKeyIdentifier
criticality MUST be FALSE
```

Authority Information Access

l'extension authorityInfoAccess, comme définis dans la rfc 5280, peut être utilisée pour assister l'AC verifier dans la vérification de statut de révocation de l'AC. Le support pour id-ad-caIssuers accessMethod est optionnel par ce profile vu que les chaînes d'AC ne sont pas attendus.

L'accessMethod suivante est utilisée pour indiquer que la vérification de statut de révocation est fournie pour cet AC, en utilisant OCSP :

```
id-ad-ocsp OBJECT IDENTIFIER ::= { id-ad 1 }
```

accessLocation doit contenir une URI, et l'URI doit contenir une URL HTTP qui spécifie l'emplacement d'un répondeur OCSP. Le fournisseur d'AC doit, bien sûr, maintenir un répondeur OCSP à cet emplacement.

```
name id-ce-authorityInfoAccess
OID { id-pe 1 }
syntax AuthorityInfoAccessSyntax
criticality MUST be FALSE
```

CRL Distribution Points

l'extension crlDistributionPoints, comme définis dans la rfc 5280, peut être utilisée pour assister l'AC verifier dans la vérification du statut de révocation de l'AC.

Si l'extension crlDistributionPoints est présent, un seul point de distribution doit être présent. l'extension crlDistributionPoints doit utiliser l'option DistributionPointName, qui doit contenir un nom complet, qui doit contenir une forme nom simple. Ce nom doit contenir soit un nom distinct soit une URI. L'URI doit être soit une URL HTTP soit une URL LDAP.

```
name id-ce-cRLDistributionPoints
OID { id-ce 31 }
syntax CRLDistributionPoints
criticality MUST be FALSE
```

No Revocation Available

L'extension noRevAvail, définis dans X.509-2000, permet à un fournisseur d'AC d'indiquer qu'aucune information de révocation n'est disponible pour cet AC.

Cette extension doit être non-critique. Un AC vérifier qui ne comprend par cette extension doit être capable de trouver une liste de révocation du fournisseur d'AC, mais la liste de révocation n'inclura jamais l'AC.

```
name id-ce-noRevAvail
OID { id-ce 56 }
syntax NULL (i.e., '0500'H is the DER encoding)
criticality MUST be FALSE
```

Types d'attributs

Certains attributs définis ci-dessous utilisent le type IetfAttrSyntax, également définis ci-dessous. Les raisons d'utiliser ce type sont :

1. Il permet une séparation entre le fournisseur d'AC et l'autorité de stratégie d'attributs. C'est utile pour les situations où une seule autorité de stratégie (par exemple une organisation) alloue des valeurs d'attributs, mais où plusieurs fournisseurs d'AC sont déployés pour les raisons de performances, ou autre.
2. Les syntaxes permises pour les valeurs sont restreintes à des chaînes d'octets, identifiant d'objet, et UTF8String, qui réduit significativement la complexité associée avec les syntaxes plus générales. Tous les attributs multi-valués utilisant cette syntaxe sont restreints pour que chaque valeur utilise le même choix de syntaxe de valeur. Par exemple, le fournisseur d'AC ne doit pas utiliser une valeur avec un oid et une autre valeur avec une chaîne.

```
IetfAttrSyntax ::= SEQUENCE {
 policyAuthority [0] GeneralNames OPTIONAL,
 values SEQUENCE OF CHOICE {
  octets OCTET STRING,
  oid OBJECT IDENTIFIER,
  string UTF8String
 }
}
```

Dans la description ci-dessous, chaque type d'attribut est soit marqué "Multiple Allowed" ou "One Attribute value only ; multiple values within the IetfAttrSyntax". Cela réfère au jeu d'AttributeValues ; le AttributeType ne se produit qu'une seul fois.

Service Authentication Information

L'attribut SvceAuthInfo identifie le titulaire de l'AC au serveur/service par un nom, et l'attribut peut inclure des information optionnelles d'authentification spécifiques au service. Typiquement, il contient une paire nom/mot de passe.

Cet attribut fournis des informations qui peuvent être présentés par l'AC verifier pour être interprété et authentifié par une application séparée dans le système cible. Noter que c'est une utilisation différente de ce qui est prévu pour l'attribut accessIdentity.

Ce type d'attribut est typiquement chiffré quand le champ authInfo contient des information sensibles, tel qu'un mot de passe.

```
name id-aca-authenticationInfo
OID { id-aca 1 }
syntax SvceAuthInfo
values Multiple allowed

SvceAuthInfo ::= SEQUENCE {
 service GeneralName,
 ident GeneralName,
 authInfo OCTET STRING OPTIONAL
}
```

Access Identity

L'attribut accessIdentity identifie le fournisseur d'AC au serveur/service. Pour cet attribut le champ authInfo ne doit pas être présent.

Cet attribut est prévu pour fournir des informations sur le titulaire d'AC, qui peut être utilisé par l'AC verifier (ou un grand système dans lequel l'AC verifier est un composant) pour autoriser les actions du titulaire dans le système de l'AC verifier. Noter que c'est une utilisation différente que l'attribut svceAuthInfo.

```
name id-aca-accessIdentity
OID { id-aca 2 }
syntax SvceAuthInfo
values Multiple allowed
```

Charging Identity

L'attribute chargingIdentity identifie le titulaire de l'AC dans un but de charge. En général, l'identité de charge sera différent des autres identités du titulaire. Par exemple, la compagnie du titulaire peut être chargé pour un service.

```
name id-aca-chargingIdentity
OID { id-aca 3 }
syntax IetfAttrSyntax
values One Attribute value only; multiple values within the IetfAttrSyntax
```

Group

L'attribut group gère les informations d'appartenance du titulaire à des groupes.

```
name id-aca-group
OID { id-aca 4 }
syntax IetfAttrSyntax
values One Attribute value only; multiple values within the IetfAttrSyntax
```

Role

L'attribut role, spécifié dans X.509-2000, gère les informations sur l'allocation de rôle du titulaire de l'AC. La syntaxe utilisée pour cet attribut est :

```
RoleSyntax ::= SEQUENCE {
 roleAuthority [0] GeneralNames OPTIONAL,
 roleName [1] GeneralName
}
```

Le champ roleAuthority peut être utilisé pour spécifier l'autorité de délivrance de certificat de spécification de rôle. Il n'y a pas de pré-requis pour qu'un certificat de spécification de rôle existe nécessairement pour le roleAuthority. Cela diffère de X.500-2000, où le champ roleAuthority est assumé au nom du fournisseur du certificat de spécification de rôle. Par exemple, pour distinguer le rôle administrateur comme définis par Baltimore de ce qui est définis par SPYRUS, on pourrait placer la valeur "urn :administrator" dans le champ roleName et la valeur "Baltimore" ou "SPYRUS" dans le champ roleAuthority.

Le champ roleName doit être présent, et roleName doit utiliser le choix uniformResourceIdentifier de GeneralName :

```
name id-at-role
OID { id-at 72 }
syntax RoleSyntax
values Multiple allowed
```

Clearance

L'attribut clearance, spécifié dans X.501-1993, gère les informations de clairance (associé avec les labels de sécurité).

Le champ policyId est utilisé pour identifier la stratégie de sécurité pour lequel la clairance est liée. policyId indique les sémantiques des champs classList et securityCategories.

Cette spécification inclus le champ classList exactement comme spécifié dans X.501-1993. Des valeurs additionnelles de classification de sécurité, et leur position dans la hiérarchie de classification, peuvent être définis par une stratégie de sécurité au niveau local ou par accord bilatéral. La hiérarchie de classification de sécurité de base est, dans l'ordre ascendant : unmarked, unclassified, restricted, confidential, secret, et top-secret.

Une organisation peut développer ses propres stratégies de sécurité qui définissent les valeurs de classification de sécurité et leur signification. Cependant, les positions 0 à 5 de la chaîne de bits sont réservés pour la hiérarchie de classification de sécurité de base.

Si présent, le champ securityCategory fournis d'autres information d'autorisation. La stratégie de sécurité identifié par le champ policyId indique les syntaxes qui sont permises dans le jeu securityCategories. Un identifiant d'objet identifie chaque syntaxe permise. Quand une de ces syntaxe est présente dans le jeu securityCategories, l'identifiant d'objet associé avec cette syntaxe est réalisé dans le champ securityCategory.type.

L'identifiant d'objet pour l'attribut clearance de la rfc3281 est :

```
id-at-clearance OBJECT IDENTIFIER ::= {
 joint-iso-ccitt(2) ds(5) module(1) selected-attribute-types(5)
 clearance (55) }
```

La syntaxe a été corrigée depuis la rfc3281, pour restaurer la conformité avec X.509-1997 :

```
Clearance ::= SEQUENCE {
 policyId OBJECT IDENTIFIER,
 classList ClassList DEFAULT {unclassified},
 securityCategories SET OF SecurityCategory OPTIONAL
}
```

L'identifiant d'objet pour l'attribut clearance de X.509-1997 est :

```
id-at-clearance OBJECT IDENTIFIER ::= {
 joint-iso-ccitt(2) ds(5) attributeType(4) clearance (55) }
```

La syntaxe associée est la suivante :

```
Clearance ::= SEQUENCE {
 policyId OBJECT IDENTIFIER,
 classList ClassList DEFAULT {unclassified},
 securityCategories SET OF SecurityCategory OPTIONAL
}
```

Les implémentations doivent supporter l'attribut clearance comme définis dans X.501-1997. Les implémentations devraient supporter le décodage de la syntaxe clearance de la rfc3281. Les implémentations ne doivent pas générer d'attribut clearance comme définis dans la rfc3281.

```
ClassList ::= BIT STRING {
 unmarked (0),
 unclassified (1),
 restricted (2),
 confidential (3),
 secret (4),
 topSecret (5)
}
```

```
SecurityCategory ::= SEQUENCE {
 type [0] OBJECT IDENTIFIER,
 value [1] EXPLICIT ANY DEFINED BY type
}
name { id-at-clearance }
OID { joint-iso-ccitt(2) ds(5) attribute-type (4) clearance (55) }
syntax Clearance - imported from [X.501-1997]
values Multiple allowed
```

Profile du PKC du AC Issuer

Le PKC du fournisseur doit être conforme à la rfc5280, et l'extension keyUsage dans le PKC ne doit pas explicitement indiquer que la clé publique du fournisseur d'AC ne peut pas être utilisée pour valider une signature numérique. Pour éviter toute confusion dans les numéros de série et les révocations, un fournisseur d'AC ne doit pas être en même temps un PKC Issuer. Donc le PKC du fournisseur d'AC ne doit pas avoir une extension basicConstraints avec cA à TRUE.

Validation de certificat d'attribut

Cette section décrit un jeu de règles de base que tout AC valide doit satisfaire. Certaines vérifications additionnelles sont également décrites, que l'AC verifier peut choisir d'implémenter. Pour être valide, un AC doit satisfaire tous les points suivant :

1. Là où le titulaire utilise un PKC pour s'authentifier auprès de l'AC verifier, le PKC du titulaire de l'AC doit être trouvé, et tout le chemin de certification de ce PKC doit être vérifié en accord avec la rfc5280. Comme mentionné dans la section Considération de Sécurité, si un autre schémas de sécurité est utilisé, les AC verifier doivent être très attentifs au mappage des identités.
2. La signature de l'AC doit être cryptographiquement correcte, et tout le chemin de certification du PKC du fournisseur d'AC doit être vérifié en accord avec la rfc5280.
3. Le PKC du fournisseur d'AC doit également être conforme au profile spécifié dans la section précédente.
4. Le fournisseur d'AC doit être directement considéré comme un fournisseur d'AC.
5. Le temps pour lequel l'AC est évalué doit être dans la validité de l'AC. Si l'évaluation est égal à soit notBeforeTime soit notAfterTime, la vérification réussit. Noter que dans certaines applications, l'évaluation du temps peut ne pas être la même que l'heure courante.
6. La vérification de l'AC targeting doit passe comme spécifié dans la section AC Targeting.
7. Si l'AC contient une extension critique non-supportée, l'AC doit être rejetée.

Le support pour une extension dans ce contexte signifie :

1. L'AC verifier doit être capable de lire la valeur de l'extension
2. Là où la valeur de l'extension implique de rejeter l'AC, l'AC verifier doit rejeter l'AC.

Vérifications additionnelles :

1. L'AC peut être rejetée sur la base d'une configuration de l'AC verifier. Par exemple, un AC verifier peut être configuré pour rejeter les AC qui contiennent ou qui n'ont pas certains attributs.
2. Si l'AC verifier fournis une interface qui permet aux applications de requêter le contenu de l'AC, alors l'AC verifier peut filtrer les atributs de l'AC en fonction de sa configuration. Par exemple, un AC verifier pourrait être configuré pour ne pas retourner certains attributs pour certains serveurs.

Révocation

Dans de nombreux environnements, la période de validité d'un AC est inférieure au temps requis pour fournir et distribuer les informations de révocation. Les AC à durée de vie courte ne nécessitent pas de support de révocation. Cependant, les AC à durée de vie longue et les environnements où les AC permettent les transaction à forte valeur, le support de la révocation peut être requise.

2 schémas de révocation sont définis, et le fournisseur d'AC devrait élire celui qui est le plus adapté à l'environnement dans lequel l'AC sera employé.

Schéma "Never revoke" : Les AC peuvent être marqués pour que les parties comprennent qu'aucune information de statut de révocation ne sera rendu disponible. L'extension NoRevAvail doit être présent dans l'AC pour indiquer l'utilisation de ce schéma. Si cette extension n'est pas présente, le fournisseur d'AC statut qu'une vérification du statut de révocation est suppportée, et certaines méthodes de révocation doivent être fournis pour permettre aux AC verifiers d'établir le statut de révocation de l'AC.

Schéma "Pointer in AC" : Les AC peuvent "pointer" sur la source d'information de statut de révocation, en utilisant soit l'extension authorityInfoAccess ou l'extension crlDistributionPoints dans l'AC.

Pour les utilisateurs d'AC, le schéma "never revoke" doit être supporté, et le "pointer in AC" devrait être supporté. Si seul le premier est supporté, tous les AC qui ne contiennent pas l'extension noRevAvail doivent être rejetés.

Un AC ne doit pas contenir à la fois l'extension noRevAvail et un "pointer in AC".

Un AC verifier peut utiliser toute information de statut de révocation de l'AC.

Fonctionnalités optionnelles

Cette section spécifie les fonctionnalités qui peuvent être implémentées. La conformité avec ce profile n'impose pas de supporter ces fonctionnalités ; cependant, si ces fonctionnalités sont supportées, elles doivent le faire tel que décris ici.

Chiffrement d'attribut

Les attributs d'AC peuvent nécessiter d'être chiffrés si l'AC est transporté dans le protocole d'application en clair ou contient des informations sensibles (par ex : username/password).

Quand un jeu d'attributs doit être chiffré dans un AC, le CMS EnvelopedData est utilisé pour transporter le texte chiffré et les informations par destinataire associés.

Ce type de chiffrement d'attribut est ciblé. Avant que l'AC soit signé, les attributs sont chiffrés pour un jeu de destinataires prédéterminés.

Dans l'EnvelopedData, encapsulatedContentInfo identifie le type de contenu transporté dans le texte chiffré. Dans ce cas, le champ contentType de encapsulatedContentInfo doit contenir id-ct-attrCertEncAttrs, qui a la valeur suivante :

```
attrCertEncAttrs OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
id-smime(16) id-ct(1) 14 }
```

Le texte chiffré est inclus dans l'AC comme valeur d'un attribut encAttrs. Seul un attribut encAttrs peut être présent dans un AC ; cependant, l'attribut encAttrs peut être multi-valué, et chacune de ses valeurs contient un EnvelopedData indépendant.

Chaque valeur peut contenir un jeu d'attributs (chaque pouvant être multi-valué) chiffré pour un jeu de déstinataires prédéterminés.

Le texte en clair qui est chiffré a le type :

```
ACClearAttrs ::= SEQUENCE {
```

```
 acIssuer GeneralName,
 acSerial INTEGER,
 attrs SEQUENCE OF Attribute
}
```

L'encodé DER de la structure ACClearAttrs est utilisée comme champ encryptedContent de EnvelopedData. L'encodé DER doit être embarqué dans une chaîne d'octets.

Les champs acIssuer et acSerial sont présents pour empêcher le vol du texte chiffré. Quand un AC verifier a déchiffré un attribut chiffré avec succès, il doit ensuite vérifier que les champs acIssuer et acSerial contiennent les même valeur. Cela empêche un fournisseur d'AC malicieux de copier le texte chiffré d'un autre AC.

La procédure pour un fournisseur d'accès pour chiffrer des attributs est illustré par les étapes suivantes (tout autre procédure qui donne le même résultat peut être utilisé) :

1. Identifier les jeux d'attributs qui doivent être chiffrés pour chaque jeu de destinataires.
2. Pour chaque jeu d'attribut qui doit être chiffré :
   - **2.1.** Créer une structure EnvelopedData pour les données pour ce jeu de destinataires
   - **2.2.** Encoder ContentInfo contenant le EnvelopedData comme valeur de l'attribut encAttrs
   - **2.3.** S'Assurer que les attributs en texte clair ne sont pas présents dans l'AC à signer
3. Ajouter encAtrs (avec ses multiples valeurs) à l'AC.

Noter qu'il peut y avoir plus d'un attribut de même type (le même identifiant d'objet) après de déchiffrement. C'est à dire, un AC peut contenir le même type d'attribut à la fois en clair et chiffré (et de nombreuses fois si différents destinataires sont associés avec plus d'un EnvelopedData). Par exemple, un AC pourrait contenir un attribut clearance en texte claire indiquant que le titulaire est autorisé à SECRET, et , en plus, un attribut clearance chiffré dont la valeur est une clairance supérieure qui ne doit pas être connus par tout le monde. Une approche peut choisir de fusionner les valeurs d'attributs après déchiffrement pour rétablir la contrainte "once only".

```
name id-aca-encAttrs
OID { id-aca 6}
syntax ContentInfo
values Multiple Allowed
```

Si un AC contient des attributs apparemment chiffrés pour l'AC verifier, un échec du processus de déchiffrement doit impliquer le rejet de l'AC.

Proxying

Quand un serveur agit comme un client pour un autre serveur de la part du titulaire, le serveur peut nécessiter proxyfier un AC. Un tel proxy peut être fait sous le contrôle du fournisseur d'AC, pour que tous les AC ne soient pas proxyfiables et qu'un AC proxyfié puisse l'être de façon ciblée. Le support pour les chaînes de proxy (avec plus d'un serveur intermédiaire) peut également être requis. Noter que cela n'implique pas une chaîne d'AC.

Pour ce pré-requis, on définis une autre extension, ProxyInfo, similaire à l'extension targeting.

Quand cette extension est présente, l'AC verifier doit vérifier que l'entité depuis lequel l'AC a été reçus était autorisé à l'envoyer et que l'AC est autorisé à être utilisé par l'AC verifier.

Les informations de proxying est une liste dans laquelle chaque élément est une liste d'informations de ciblage. Si le verifieur et l'émetteur de l'AC sont nommés dans la même liste de proxy, l'AC peut alors être accepté (la règle exacte est donnée plus bas).

L'effet est que le titulaire de l'AC peut envoyer l'AC à une cible valide qui peut ainsi seulement proxyfier aux cibles qui sont dans une des même listes que lui.

La structure de donnée suivante est utilisée pour représenter les information de targeting/proxying :

```
ProxyInfo ::= SEQUENCE OF Targets
```

Les Targets sont expliqués dans la section AC Targeting. Comme dans le cas du targeting, le choix targetCert ne doit pas être utilisé.

Une vérification de proxy réussie si une des conditions suivantes est rencontrée :

1. L'identité de l'émetteur, comme établit par le service d'authentification sous-jacent, correspond au champ Holder de l'AC, et le serveur courant correspond à un nom dans les jeux proxy.
2. L'identité de l'émetteur, comme établit par le service d'authentification sous-jacent, correspond à un des jeux proxy (appelons le jeu A), et le serveur courant est un des champs targetName dans le jeu A, ou le serveur courant est un membre d'un des champs targetGroup dans le jeu A.

Quand un AC est proxyfié plus d'une fois, un nombre de cibles seront sur le chemin du client original, qui est normalement, mais pas toujours, le titulaire de l'AC. Dans de tels cas, la prévention du vol d'AC nécessite que l'AC verifier vérifie toujours que toutes les cibles dans le chemin sont dans le jeu proxy. Il est de la responsabilité du protocole utilisant d'AC de s'assurer qu'une liste de cibles de confiance dans le chemin soit disponible à l'AC verifier.

```
name id-pe-ac-proxying
OID { id-pe 10 }
syntax ProxyInfo
criticality MUST be TRU
```

Utilisation de ObjectDigestInfo

Dans certains environnements, il peut être requis que l'AC ne soit pas lié soit à une identité (via entityName) ou à un PKC (via baseCertificateID). Le choix objectDigestInfo dans le champ Holder permet de le gérer.

Si le titulaire est identifié avec le champ objectDigestInfo, alors le champ version de l'AC doit contenir v2 (l'entier 1).

L'idée est de lier l'AC à un objet en plaçant un hash de cet objet dans le champ Holder de l'AC. Par exemple, cela permet la production d'AC qui sont liés aux clés publiques au lieu de leur noms. Cela permet également la production d'AC qui contiennent des privilèges associés avec un objet exécutable telle qu'une classe java au travers d'une clé publique ou un PKC. Les AC conformes ne doivent pas utiliser de valeur otherObjectTypes pour le digestedObjectType.

Pour lier un AC à une clé publique, le hash doit être calculé sur la présentation de cette clé publique, qui pourrait être présente dans un PKC, spécifiquement, l'entrée pour l'algorithme de hash doit être l'encodé DER d'une représentation SubjectPublicKeyInfo de la clé.

Note : cela inclus l'AlgorithmIdentifier aussi bien que la chaîne de bit. Les règles donnée dans les algorithmes PKIX (rfc3279, rfc4055, rfc5480, et rfc5756) pour encoder les clés doivent être suivies. Dans ce cas, le digestedObjectType doit être publicKey et le champ otherObjectTypeID ne doit pas être présent.

Noter que si la valeur de la clé publique utilisé en entrée de la fonction de hashage a été extraite d'un PKC, il est possible que le SubjectPublicKeyInfo de ce PKC ne soit pas la valeur qui devrait être hashé. Cela peut se produire si les Dss-parms de DSA sont hérités comme décris dans la section 2.3.2 de la rfc3279. L'entrée correcte pour le hashage dans ce contexte inclus la valeur des paramètres hérités du PKC de la CA, et donc peut différer du SubjectPublicKeyInfo présent dans le PKC.

Les implémentations qui supportent cette fonctionnalité doivent être capable de gérer les représentations de clé publique pour les algorithmes spécifiés dans la section 2.3 de la rfc3279.

Pour lier un AC à un PKC via un hash, le hash doit être calculé sur l'encodé DER de tout le PKC, incluant la valeur de signature. Dans ce cas, digestedObjectType doit être publicKeyCert et otherObjectTypeID ne doit pas être présent.

Contrôles AA

Durant la validation d'AC, un partie de confiance doit répondre à la question : est-ce que ce fournisseur d'AC est valide pour fournir des ACs avec cet attribut ? L'extension AAControls est prévu pour être utilisé dans les CA et les fournisseurs d'AC

```
id-pe-aaControls OBJECT IDENTIFIER ::= { id-pe 6 }

AAControls ::= SEQUENCE {
 pathLenConstraint INTEGER (0..MAX) OPTIONAL,
 permittedAttrs [0] AttrSpec OPTIONAL,
 excludedAttrs [1] AttrSpec OPTIONAL,
 permitUnSpecified BOOLEAN DEFAULT TRUE
}

AttrSpec::= SEQUENCE OF OBJECT IDENTIFIER
```

L'extension AAControls est utilisé comme suit :

PathLenConstraint si présent,, est interprété comme dans la rfc5280. Il restreint la distance permise entre le AA CA (une CA directement validé pour inclure les AAControls dans ses PKCs), et le fournisseur d'AC.

permittedAttrs spécifie une liste de types d'attributs que tout fournisseur d'AC sous ce AA CA est autorisé à inclure dans les AC. Si ce champ n'est pas présent, cela signifie qu'aucun type d'attribut n'est explicitement permis.

excludedAttrs spécifie une liste de types d'attributs qu'aucun fournisseur d'AC sous ce AA CA n'est autorisé à inclure dans les AC. Si ce champ n'est pas présent, cela signifie qu'aucun type d'attribut n'est explicitement interdis.

permitUnSpecified spécifie comment manipuler les types d'attributs qui ne sont pas présents dans les champs permittedAttrs ou excludedAttrs. TRUE (par défaut) signifie qu'un type d'attribut non-spécifié est autorisé dans les AC, et FALSE, qu'il ne l'est pas.

- Quand AAControls est utilisé, les vérifications additionnelles suivantes sur la chaîne de PKC d'AA doit tous réussir pour que l'AC soit valide :

  1. Certaines CA dans le chemin de certification de l'AC doivent être directement trustés pour fournir des PKCs qui précède le fournisseur d'AC dans le chemin de certification ; appelons cette CA le "AA CA".

  2. Tous les PKC dans le chemin depuis le AA CA, incluant le PKC du fournisseur d'AC, doivent contenir l'extension AAControls ; cependant, le PKC du AA CA n'a pas besoin de contenir cette extension.

  3. Seul les attributs dans l'AC qui sont permis, en accord avec toutes les valeurs d'extension AAControls dans tous les PKC depuis le AA CA jusqu'au fournisseur d'AC, peuvent être utilisé pour les décisions d'autorisation ; tous les autres attributs doivent être ignorés. Cette vérification doit être appliquée à la liste des attributs suivant le déchiffrement d'attributs, et le type id-aca-encAtrs doit être également vérifié.

```
name id-pe-aaControls
OID { id-pe 6 }
syntax AAControls
criticality MAY be TRUE
```

Considérations de sécurité

La protection offerte pour les clés privées est un facteur critique pour maintenir la sécurité. Si les fournisseurs d'AC ne parviennent pas à protéger leur clé privée, un attaquant pour les utiliser, potentiellement pour générer de faux AC ou statut de révocation. Si le système est compromis, tous les AC fournis par le fournisseur d'AC doivent être révoqués. La reconstruction sera problématique, donc les fournisseurs

d'AC doivent implémenter une combinaison de techniques fortes (ex : modules cryptographique inviolable) et de procédures de gestion appropriés (ex : séparation des fonctions) pour éviter un tel incident.

La perte d'une clé privée d'un fournisseur d'AC peut également être problématique. Le fournisseur d'AC ne sera plus capable de produire de status de révocation ou de renouvellement d'AC. Les fournisseur d'AC doivent maintenir une sauvegarde sécurisée de leur clé de signatue. La sécurité des procédures de sauvegarde de clé est un facteur critique.

La disponibilité et la "fraîcheur" du status de révocation va affecter le degré d'assurance qui serai placé dans un AC long terme. Bien que les AC long-terme expirent naturellement, des évènements peuvent se produire durant sa durée de vie. Si le status de révocation est non-timé ou non-disponible, l'assurance associée avec le lien entre le titulaire de l'AC et les attributs est clairement réduit.

La liaison entrée un titulaire d'AC et les attributs ne peut pas être plus forte que l'implémentation cryptographique et les algorithmes utilisé pour générer la signature. Des clé courtes ou des algorithmes faibles limitent l'utilité d'un AC.

Les applications qui sont inconsistant lors de la comparaison des noms peut engendrer l'acceptation de cibles ou proxy invalides, ou le rejet de certains valides. Les séries X.500 de spécifications définissent des règles pour comparer les noms distincts. Ces règles nécessitent la comparaison de chaînes sans regarder la casse, le jeu de caractères, les espace blancs multi-caractères, ou les espaces blanc de début et de fin. Cette spécification relaxe ces pré-requis, nécessitant une comparaison binaire au minimum.

Les fournisseurs d'AC doivent encoder le nom distinct dans le champ holder.entityName de l'AC identiquement au nom distinct dans le PKC du titulaire. Si des encodages différents sont utilisé, les implémentations de cette spécifiaciton peuvent échouer dans la reconnaissance de l'AC et du PKC appartenant à la même entité.

Si un certificat d'attribut est lié au PKC du titulaire un utilisant le composant baseCertificateID du champ Holder et la PKI utilisée inclue un faux CA avec le même nom de fournisseur spécifié dans le composant baseCertificateID, ce faux CA pourrait fournir un PKC à un paire malicieux, en utilisant le même nom de fournisseur et numéro de série que le PKC du titulaire. Ainsi, le paire malicieux pourrait utiliser ce PKC en conjonction avec l'AC. Ce scénario devrait être évité en gérant et en configurant proprement la PKI pour qu'il ne puisse pas y avoir 2 CA avec le même nom.

Une autre alternative est de lier les AC aux PKC en utilisant le type publicKeyCert dans le champ ObjectDigestInfo. Les AC verifier doivent établir (en utilisant d'autres moyens) que les collisions potentiels ne peuvent pas se produire, par exemple, le Certificate Practice Statements (CPSs) des CA impliqués peuvent préciser qu'une collision de nom ne puisse se produire.

Les implémenteurs doivent s'assurer qu'en suivant la validation d'un AC, seuls les attributs dont le fournisseur est autorisé à fournir sont utilisés dans les décisions d'autorisation. Les autres attributs, qui peuvent être présents doivent être ignorés. L'extension PKC AAControls est optionnel, et les informations de configuration peuvent être une alternative. Cela devient très important si un AC verifier trust plus d'un fournisseur d'AC.

Il est souvent requis de mapper l'authentification fournie par un protocole de sécurité particulier (ex : TLS, S/MIME) et l'identité du titulaire de l'AC. Si l'authentification utilise les PKC, alors le mapping est simple. Cependant, il est envisagé que les AC sont également utilisés dans des environnement où le titulaire peut être authentifié en utilisant d'autres moyens. Les implémenteurs devraient prendre en charge le mapping d'identité d'authentification qui ne proviennent pas de certificats à clé publique.

Appendix A - Object Identifiers

Cet appendix liste les nouveaux identifiant d'objet qui sont définis dans cette spécification. Certains sont requis uniquement pour le support de fonctionnalités optionnels et ne sont pas obligatoire pour se conformer à ce profile.

Les OID suivant sont importés depuis les rfc de profile PKIX :

```
id-pkix OBJECT IDENTIFIER ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5)
mechanisms(5) pkix(7) }
id-mod OBJECT IDENTIFIER ::= { id-pkix 0 }
id-pe OBJECT IDENTIFIER ::= { id-pkix 1 }
id-ad OBJECT IDENTIFIER ::= { id-pkix 48 }
```

```
id-at OBJECT IDENTIFIER ::= { joint-iso-ccitt(2) ds(5) 4 }
id-ce OBJECT IDENTIFIER ::= { joint-iso-ccitt(2) ds(5) 29 }
```

Le nouvel OID de mode ASN.1 suivant est définis :

```
id-mod-attribute-cert OBJECT IDENTIFIER ::= { id-mod 12 }
```

Les OID d'extension d'AC suivant sont définis :

```
id-pe-ac-auditIdentity OBJECT IDENTIFIER ::= { id-pe 4 }
id-pe-ac-proxying OBJECT IDENTIFIER ::= { id-pe 10 }
id-ce-targetInformation OBJECT IDENTIFIER ::= { id-ce 55 }
```

Les OID d'extension PKC suivants sont définis :

```
id-pe-aaControls OBJECT IDENTIFIER ::= { id-pe 6 }
```

Les OID d'attributs suivant sont définis :

```
id-aca OBJECT IDENTIFIER ::= { id-pkix 10 }
id-aca-authenticationInfo OBJECT IDENTIFIER ::= { id-aca 1 }
id-aca-accessIdentity OBJECT IDENTIFIER ::= { id-aca 2 }
id-aca-chargingIdentity OBJECT IDENTIFIER ::= { id-aca 3 }
id-aca-group OBJECT IDENTIFIER ::= { id-aca 4 }
id-aca-encAttrs OBJECT IDENTIFIER ::= { id-aca 6 }
id-at-role OBJECT IDENTIFIER ::= { id-at 72 }
id-at-clearance OBJECT IDENTIFIER ::= {
 joint-iso-ccitt(2) ds(5) attributeType(4) clearance (55) }
id-at-clearance OBJECT IDENTIFIER ::= {
 joint-iso-ccitt(2) ds(5) module(1) selected-attribute-types(5) clearance (55) }
```

Appendix B - Module ASN.1

Cet appendix décrit les objects utilisé par les composant de PKI conformes dans une syntaxe ASN.1-like, un hybrid des syntaxe 1988 et 1993 d'ASN.1.

```
PKIXAttributeCertificate-2008 { iso(1) identified-organization(3)
 dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
 id-mod-attribute-cert-v2(61) }

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

- EXPORTS ALL -

IMPORTS

- Les OID des modules IMPORTés peuvent changer si les rfc de profile PKIX changent les extensions de
certificat PKIX.

Attribute, AlgorithmIdentifier, CertificateSerialNumber,
Extensions, UniqueIdentifier, id-pkix, id-pe, id-kp, id-ad, id-at
 FROM PKIX1Explicit88
  { iso(1) identified-organization(3) dod(6) internet(1)
   security(5) mechanisms(5) pkix(7) id-mod(0)
   id-pkix1-explicit-88(18) }
```

```
GeneralName, GeneralNames, id-ce, AuthorityKeyIdentifier,
 AuthorityInfoAccessSyntax, CRLDistributionPoint
  FROM PKIX1Implicit88
  { iso(1) identified-organization(3) dod(6) internet(1)
   security(5) mechanisms(5) pkix(7) id-mod(0)
   id-pkix1-implicit-88(19) }

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

;

id-pe-ac-auditIdentity OBJECT IDENTIFIER ::= { id-pe 4 }

id-pe-aaControls OBJECT IDENTIFIER ::= { id-pe 6 }

id-pe-ac-proxying OBJECT IDENTIFIER ::= { id-pe 10 }

id-ce-targetInformation OBJECT IDENTIFIER ::= { id-ce 55 }

id-aca OBJECT IDENTIFIER ::= { id-pkix 10 }

id-aca-authenticationInfo OBJECT IDENTIFIER ::= { id-aca 1 }

id-aca-accessIdentity OBJECT IDENTIFIER ::= { id-aca 2 }

id-aca-chargingIdentity OBJECT IDENTIFIER ::= { id-aca 3 }

id-aca-group OBJECT IDENTIFIER ::= { id-aca 4 }

- { id-aca 5 } is reserved

id-aca-encAttrs OBJECT IDENTIFIER ::= { id-aca 6 }

id-at-role OBJECT IDENTIFIER ::= { id-at 72}

id-at-clearance OBJECT IDENTIFIER ::= {
 joint-iso-ccitt(2) ds(5) attributeType(4) clearance (55) }

- Décommenter la déclaration suivante et commenter la ligne au dessus pour utiliser
- l'attribut id-at-clearance comme définis dans la rfc3281.

- id-at-clearance OBJECT IDENTIFIER ::= {
- joint-iso-ccitt(2) ds(5) module(1) selected-attribute-types(5)
- clearance (55) }

- Uncomment this if using a 1988 level ASN.1 compiler

- UTF8String ::= [UNIVERSAL 12] IMPLICIT OCTET STRING

AttributeCertificate ::= SEQUENCE {
 acinfo AttributeCertificateInfo,
 signatureAlgorithm AlgorithmIdentifier,
 signatureValue BIT STRING
}

AttributeCertificateInfo ::= SEQUENCE {
```

```
 version AttCertVersion, - version is v2
 holder Holder,
 issuer AttCertIssuer,
 signature AlgorithmIdentifier,
 serialNumber CertificateSerialNumber,
 attrCertValidityPeriod AttCertValidityPeriod,
 attributes SEQUENCE OF Attribute,
 issuerUniqueID UniqueIdentifier OPTIONAL,
 extensions Extensions OPTIONAL
}

AttCertVersion ::= INTEGER { v2(1) }

Holder ::= SEQUENCE {
 baseCertificateID [0] IssuerSerial OPTIONAL,
  - the issuer and serial number of
  - the holder's Public Key Certificate
 entityName [1] GeneralNames OPTIONAL,
  - the name of the claimant or role
 objectDigestInfo [2] ObjectDigestInfo OPTIONAL
  - used to directly authenticate the
  - holder, for example, an executable
}

ObjectDigestInfo ::= SEQUENCE {
 digestedObjectType ENUMERATED {
  publicKey (0),
  publicKeyCert (1),
  otherObjectTypes (2) },
 - otherObjectTypes MUST NOT
 - MUST NOT be used in this profile
 otherObjectTypeID OBJECT IDENTIFIER OPTIONAL,
 digestAlgorithm AlgorithmIdentifier,
 objectDigest BIT STRING
}

AttCertIssuer ::= CHOICE {
 v1Form GeneralNames, - MUST NOT be used in this profile
 v2Form [0] V2Form - v2 only
}

V2Form ::= SEQUENCE {
 issuerName GeneralNames OPTIONAL,
 baseCertificateID [0] IssuerSerial OPTIONAL,
 objectDigestInfo [1] ObjectDigestInfo OPTIONAL
  - issuerName MUST be present in this profile
  - baseCertificateID and objectDigestInfo MUST
  - NOT be present in this profile
}

IssuerSerial ::= SEQUENCE {
 issuer GeneralNames,
 serial CertificateSerialNumber,
 issuerUID UniqueIdentifier OPTIONAL
}

AttCertValidityPeriod ::= SEQUENCE {
 notBeforeTime GeneralizedTime,
 notAfterTime GeneralizedTime
```

```
}

Targets ::= SEQUENCE OF Target

Target ::= CHOICE {
 targetName [0] GeneralName,
 targetGroup [1] GeneralName,
 targetCert [2] TargetCert
}

TargetCert ::= SEQUENCE {
 targetCertificate IssuerSerial,
 targetName GeneralName OPTIONAL,
 certDigestInfo ObjectDigestInfo OPTIONAL
}

IetfAttrSyntax ::= SEQUENCE {
 policyAuthority [0] GeneralNames OPTIONAL,
 values SEQUENCE OF CHOICE {
  octets OCTET STRING,
  oid OBJECT IDENTIFIER,
  string UTF8String
 }
}

SvceAuthInfo ::= SEQUENCE {
 service GeneralName,
 ident GeneralName,
 authInfo OCTET STRING OPTIONAL
}

RoleSyntax ::= SEQUENCE {
 roleAuthority [0] GeneralNames OPTIONAL,
 roleName [1] GeneralName
}

Clearance ::= SEQUENCE {
 policyId OBJECT IDENTIFIER,
 classList ClassList DEFAULT {unclassified},
 securityCategories SET OF SecurityCategory OPTIONAL
}

- Uncomment the following lines to support deprecated clearance
- syntax and comment out previous Clearance.

- Clearance ::= SEQUENCE {
- policyId [0] OBJECT IDENTIFIER,
- classList [1] ClassList DEFAULT {unclassified},
- securityCategories [2] SET OF SecurityCategory OPTIONAL
- }

ClassList ::= BIT STRING {
 unmarked (0),
 unclassified (1),
 restricted (2),
 confidential (3),
 secret (4),
 topSecret (5)
}
```

```
SecurityCategory ::= SEQUENCE {
 type [0] OBJECT IDENTIFIER,
 value [1] EXPLICIT ANY DEFINED BY type
}

- Note that in [RFC3281] the syntax for SecurityCategory was
- as follows:
-
- SecurityCategory ::= SEQUENCE {
- type [0] IMPLICIT OBJECT IDENTIFIER,
- value [1] ANY DEFINED BY type
- }
-
- The removal of the IMPLICIT from the type line and the
- addition of the EXPLICIT to the value line result in
- no changes to the encoding.

AAControls ::= SEQUENCE {
 pathLenConstraint INTEGER (0..MAX) OPTIONAL,
 permittedAttrs [0] AttrSpec OPTIONAL,
 excludedAttrs [1] AttrSpec OPTIONAL,
 permitUnSpecified BOOLEAN DEFAULT TRUE
}

AttrSpec ::= SEQUENCE OF OBJECT IDENTIFIER

ACClearAttrs ::= SEQUENCE {
 acIssuer GeneralName,
 acSerial INTEGER,
 attrs SEQUENCE OF Attribute
}

ProxyInfo ::= SEQUENCE OF Targets

END
```