Groupe de travail Réseau
Request pour Comments : 4534
Catégorie : Sur la voie de la normalisation
Traduction Claude Brière de L'Isle

A. Colegrove, SPARTA, Inc.
H. Harney, SPARTA, Inc.

juin 2006

Jeton de politique de sécurité de groupe, v1

Statut du présent mémoire

Le présent document spécifie un protocole de l'Internet en cours de normalisation pour la communauté de l'Internet, et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "Protocoles officiels de l'Internet" (STD 1) pour voir l'état de normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé

Le jeton de politique de sécurité de groupe est une structure utilisée pour spécifier la politique de sécurité et les paramètres configurables pour un groupe cryptographique, comme un groupe de diffusion groupée sécurisé. Parce que la sécurité d'un groupe est composée de la totalité de multiples services, mécanismes, et attributs de sécurité sur toute l'infrastructure de communications, une représentation authentifiable des caractéristiques qui doivent être prises en charge sur tout le système est nécessaire pour assurer une sécurité cohérente. Le présent document spécifie la structure d'un tel jeton.

Table des matières

1. Introduction

L'architecture de groupe de diffusion groupée [RFC3740] définit l'infrastructure de sécurité pour prendre en charge des communications de groupe sûres. Le jeton de politique suppose cette architecture dans sa définition. Il définit les paramètres de sécurité applicables pour une association de groupe sûre.

Le jeton de politique est une construction de données vérifiable signée par le propriétaire du groupe, entité qui a l'autorisation de créer la politique de sécurité. Les contrôleurs de groupe dans un groupe vont utiliser le jeton de politique pour s'assurer que les mécanismes utilisés pour sécuriser le groupe sont corrects et pour mettre en application les règles de contrôle d'accès pour les nouveaux membres. Les membres du groupe, qui peuvent contribuer par des données au groupe ou accéder à des données provenant du groupe, vont utiliser le jeton de politique pour s'assurer que le groupe est possédé par une autorité de confiance. Aussi, les membres peuvent vouloir vérifier que les règles de contrôle d'accès sont adéquates pour protéger les données que le membre soumet au groupe.

Le jeton de politique est spécifié en ASN.1 [X.208] et doit être codé en DER [X.660]. Cette caractéristique de spécification permet au jeton d'importer facilement des définitions de groupe qui s'étendent sur différentes applications et divers environnements. L'ASN.1 permet au jeton de spécifier des branches qui peuvent être utilisées par tout protocole de sécurité de diffusion groupée. Tout groupe peut utiliser cette structure de jeton de politique pour spécifier l'utilisation de plusieurs protocoles pour sécuriser le groupe.

On a pris soin dans cette spécification de fournir un niveau central de spécificité de jeton qui va faciliter l'extensibilité et la flexibilité des mécanismes de soutien. Cela a été fait avec la construction abstraite suivante :

```
Mechanism ::= SEQUENCE {
     mechanismIdentifier  IDENTIFIANT D'OBJET,
     mechanismParameters CHAINE D'OCTETS
   }
```

Cette construction va permettre l'utilisation des mécanismes de groupe spécifiés dans d'autres documents avec le jeton de politique.

Le jeton de politique est structuré pour refléter les couches de l'architecture MSEC [RFC4046] pour une association de sécurité de groupe. Chacune des couches architecturales est identifiée et reçoit une branche dans le jeton "cœur". Cela permet un haut degré de souplesse pour les spécifications de protocole futures à chaque couche architecturale sans qu'il soit besoin de changer le jeton de politique "cœur", qui peut alors agir comme un seul point de référence pour définir des groupes sûrs en utilisant toute combinaison de protocoles pour tous les environnements.

2. Création et réception de jeton

Au moment de la création du groupe ou chaque fois que la politique du groupe est mise à jour, le propriétaire du groupe (GO) va créer un nouveau jeton de politique.

Pour s'assurer de l'authenticité de la politique spécifiée, le jeton DOIT être signé par le GO. Le jeton signé DOIT être conforme au type SignedData *(données signées)* de la syntaxe de message cryptographique (CMS) [RFC3852].

Le contenu des SignedData est le jeton lui-même. Il est représenté avec l'identifiant d'objet ContentType de

IDENTIFIANT D'OBJET id-ct-msec-token ::= {1.3.6.1.5.5.12.1.1}

La valeur sid de CMS de SignerInfo, qui identifie la clé publique nécessaire pour valider la signature, DOIT être celle du GO.

Le champ signedAttrs DOIT être présent. En plus des champs minimums exigés de signedAttrs, l'attribut signing-time *(heure de signature)* DOIT être présent.

À réception d'un jeton de politique, le receveur DOIT vérifier que :
- le GO, identifié par le sid dans SignerInfo, est l'entité attendue ;
- la valeur de "signing-time" est plus récente que la valeur de "signing-time" vue dans un jeton de politique reçu

précédemment pour ce groupe, ou le jeton de politique est le premier vu par le receveur pour ce groupe ;
- le traitement de la signature la valide en accord avec la RFC 3852 ;
- les mécanismes de sécurité et de communication spécifiés (ou au moins un mécanisme de chaque choix) sont pris en charge et sont conformes à la politique locale du receveur.

3. Jeton de politique

La structure du jeton de politique est la suivante :

```
Jeton ::= SEQUENCE {
     tokenInfo     TokenID,
     registration  SEQUENCE DE Registration,
     rekey         SEQUENCE DE GroupMngmtProtocol,
     data          SEQUENCE DE DataProtocol
    }
```

tokenInfo fournit des informations sur l'instance de jeton de politique (PT, *Policy Token*).

registration fournit une liste des politiques et mécanismes acceptables d'enregistrement et désenregistrement qui peuvent être utilisés pour gérer les adhésions et départs d'un groupe, initiés par les membres. Une séquence NULL indique que le groupe ne prend pas en charge l'enregistrement et le désenregistrement de membres. Un membre DOIT être capable de prendre en charge au moins un ensemble de mécanismes d'enregistrement afin de se joindre au groupe. Lorsque plusieurs mécanismes sont présents, un membre PEUT utiliser l'une quelconque des méthodes citées. La liste est ordonnée en termes de préférences du propriétaire du groupe. Un membre DOIT choisir le plus fort mécanisme cité que la politique locale prend en charge.

rekey fournit les protocoles de changement de clé qui vont être utilisés dans la gestion du groupe. Le membre DOIT être capable d'accepter un des types de messages de changement de clé cités. La liste est ordonnée en termes de préférences du propriétaire du groupe. Un membre DOIT choisir le plus fort mécanisme cité que la politique locale prend en charge.

data fournit les applications utilisées dans les communications entre membres du groupe. Quand plusieurs applications sont fournies, l'ordre de la liste implique l'ordre d'encapsulation des données. Un membre DOIT être capable de prendre en charge toutes les applications citées et si des choix de mécanismes sont fournis par application, le membre DOIT prendre en charge au moins un de ces mécanismes.

Pour les champs registration, rekey, et data, les mises en œuvre qui rencontrent des identifiants de protocole inconnus DOIVENT traiter cela en douceur en fournissant des indicateurs qu'un protocole inconnu est dans la séquence des protocoles permis. Si le protocole inconnu est le seul protocole admissible dans la séquence, alors la mise en œuvre ne peut pas prendre en charge ce champ, et le membre ne peut pas rejoindre le groupe. C'est une affaire de politique locale de décider si une jonction est permise quand un protocole inconnu existe parmi les protocoles admissibles connus.

Les protocoles en plus de registration, rekey, et données NE DEVRAIENT PAS être ajoutés aux versions suivantes de ce jeton sauf changement de l'architecture MSEC [RFC4046].

Chaque champ de données du PT est spécifié plus en détails dans les paragraphes qui suivent.

3.1 Identifiants de jetons

tokenInfo identifie explicitement une version du jeton de politique pour un groupe particulier. Il est défini par :

```
TokenID ::= SEQUENCE {
     tokenDefVersion ENTIER (1),
     groupName       CHAINE D'OCTETS,
     edition         ENTIER FACULTATIF
    }
```

tokenDefVersion est la version de la spécification du jeton de politique de groupe. La présente spécification (v1) est représentée par un (1). Les changements à la structure du jeton de politique de sécurité de groupe exigeront une mise à

jour de ce champ.

groupName est l'identifiant du groupe et DOIT être unique par rapport au propriétaire du groupe (GO).

edition est un ENTIER facultatif qui indique le numéro de séquence du PT. Si edition est présent, les entités du groupe DOIVENT n'accepter un PT que quand la valeur est supérieure à la dernière valeur vue dans un PT valide pour ce groupe.

Le type LifeDate est aussi défini pour fournir des méthodes standard d'indiquer les horodatages et intervalles dans les jetons.

```
LifeDate ::= CHOIX {
     gt         GeneralizedTime,
     utc        UTCTime,
     interval   ENTIER
    }
```

3.2 Politique d'enregistrement

L'association de sécurité (SA, *security association*) d'enregistrement est définie dans l'architecture MSEC [RFC4046]. Durant l'enregistrement, un membre prospectif du groupe et le contrôleur de groupe vont interagir pour donner au membre du groupe l'accès aux clés et aux informations dont il a besoin pour se joindre au groupe et participer à la SA de données du groupe.

La partie désenregistrement permet à un membre actuel du groupe de notifier au serveur de clés/contrôleur de groupe (GC/KS, *Group Controller/Key Server*) qu'il ne va plus participer à la SA de données.

```
Registration ::= SEQUENCE {
     register       GroupMngmtProtocol,
     de-register    GroupMngmtProtocol
    }
```

Les protocoles pour l'enregistrement et le désenregistrement sont chacun spécifiés par :

```
GroupMngmtProtocol ::= CHOIX {
     aucun                      NULL,
     protocole pris en charge
    }

Protocol ::= SEQUENCE {
     protocol        IDENTIFIANT D'OBJET,
     protocolInfo    CHAINE D'OCTETS
    }
```

Par exemple, le protocole d'enregistrement de "register" pourrait être spécifié comme protocole de gestion de clés d'association de groupe sécurisé (GSAKMP, *Group Secure Association Key Management Protocol*) [RFC4535]. L'IDENTIFIANT D'OBJET TBS va être suivi par les paramètres utilisés dans l'enregistrement GSAKMP comme spécifié dans l'Appendice B.1.

3.3 Politique de changement de clés

La SA Rekey est définie dans l'architecture MSEC [RFC4046]. Durant le changement de clé d'un groupe, plusieurs changements peuvent être faits :
- rafraîchir/changer les clés de protection du groupe,
- mettre à jour le jeton de politique,
- changer les membres du groupe.

Durant le changement de clé, la participation au groupe peut être modifiée, les clés de protection du trafic de groupe peuvent être rafraîchies, et le jeton de politique peut être mis à jour.

Ce champ est aussi spécifié comme une séquence de protocoles qui vont être utilisés par le GC/KS.

3.4 Politique de données de groupe

La SA de données est le consommateur ultime des clés de groupe. Le champ de données va indiquer les clés et les mécanismes qui sont à utiliser dans les communications entre les membres du groupe. Plusieurs protocoles pourraient utiliser les clés de groupe, allant des simples applications de sécurité qui ont seulement besoin d'une clé pour le chiffrement et/ou la protection de l'intégrité à des protocoles de sécurité configurables plus complexes comme IPsec et le protocole sûr de transport en temps réel (SRTP, *Secure Real-time Transport Protocol*) [RFC3711]. Le séquençage des mécanismes de la SA de données est de "l'intérieur" à "l'extérieur". C'est à dire que la première SA de données définie dans un jeton de politique doit agir sur les données brutes. Toute SA de données spécifiée après cela va être appliquée à son tour.

DataProtocol ::= Protocol

4. Considérations sur la sécurité

Le présent document spécifie la structure pour un jeton de politique de groupe. À ce titre, la structure reçue par une entité de groupe doit être d'une authenticité vérifiable. Ce jeton de politique utilise la CMS pour appliquer l'authentification par des signatures numériques. La sécurité de ce schéma s'appuie sur une mise en œuvre sûre de la CMS, du choix d'un mécanisme de signature de force appropriée pour le groupe qui utilise le jeton de politique, et de clés sûres, de force suffisante. De plus, il s'appuie sur la connaissance d'un propriétaire de groupe bien connu comme racine de l'application de la politique.

Enfin, alors que le propriétaire du groupe peut faire la liste des divers mécanismes pour des fonctions variées, le groupe n'est pas plus fort que le plus faible mécanisme accepté. À ce titre, le propriétaire du groupe est chargé de ne fournir que des mécanismes de sécurité acceptables.

5. Considérations relatives à l'IANA

Les identifiants d'objet suivants ont été alloués :

- IDENTIFIANT D'OBJET id-ct-msec-token ::= 1.3.6.1.5.5.12.1.1
- IDENTIFIANT D'OBJET id-securitySuiteOne ::= 1.3.6.1.5.5.12.2.1
- IDENTIFIANT D'OBJET id-GSAKMPv1RegistrationProtocol ::= 1.3.6.1.5.5.12.3.1
- IDENTIFIANT D'OBJET id-GSAKMPv1DeRegistrationProtocol ::= 1.3.6.1.5.5.12.3.2
- IDENTIFIANT D'OBJET id-GSAKMPv1Rekey ::= 1.3.6.1.5.5.12.3.3
- IDENTIFIANT D'OBJET id-rekeyNone ::= 1.3.6.1.5.5.12.4.1
- IDENTIFIANT D'OBJET id-rekeyMethodGSAKMPLKH ::= 1.3.6.1.5.5.12.4.2
- IDENTIFIANT D'OBJET id-reliabilityNone ::= 1.3.6.1.5.5.12.5.1
- IDENTIFIANT D'OBJET id-reliabilityResend ::= 1.3.6.1.5.5.12.5.2
- IDENTIFIANT D'OBJET id-reliabilityPost ::= 1.3.6.1.5.5.12.5.3
- IDENTIFIANT D'OBJET id-subGCKSSchemeNone ::= 1.3.6.1.5.5.12.6.1
- IDENTIFIANT D'OBJET id-subGCKSSchemeAutonomous ::= 1.3.6.1.5.5.12.6.2
- IDENTIFIANT D'OBJET id-genericDataSA ::= 1.3.6.1.5.5.12.7.1

Le jeton de politique de sécurité de groupe peut être étendu par une spécification. Les extensions sous la forme d'objets peuvent être enregistrés par l'IANA. Les extensions qui exigent des changements de la structure du protocole vont exiger une mise à jour du champ tokenDefVersion de l'identifiant de jeton (TokenID) (voir au paragraphe 3.1).

6. Références

6.1 Références normatives

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

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

[RFC4046] M. Baugher et autres, "Architecture de gestion de clé de groupe de diffusion groupée sécurisée (MSEC)", avril 2005. *(Info.)*

[RFC4535] H. Harney et autres, "GSAKMP : protocole de gestion de clés d'association de groupe sécurisé", juin 2006. *(P.S.)*

[X.208] Recommandation UIT-T X.208, "Spécification de la notation de syntaxe abstraite n° 1 (ASN.1)", 1988.

[X.660] Recommandation UIT-T X.660, "Technologie de l'information – Règles de codage ASN.1 : Spécification des règles de codage de base (BER), des règles de codage canoniques (CER), et des règles de codage distinctives (DER)", 1997.

6.2 Références pour information

[HCLM00] Harney, H., Colegrove, A., Lough, P., et U. Meth, "GSAKMP Token Specification", Travail en cours, février 2003.

[HCM01] H. Harney, A. Colegrove, P. McDaniel, "Principles of Policy in Secure Groups", Proceedings of Network and Distributed Systems Security 2001 Internet Society, San Diego, CA, février 2001.

[HHMCD01] Hardjono, T., Harney, H., McDaniel, P., Colegrove, A., et P. Dinsmore, "Group Security Policy Token: Definition and Payloads", Travail en cours, août 2003.

[RFC3711] M. Baugher et autres, "Protocole de transport sécurisé en temps réel (SRTP)", mars 2004. *(P.S.)*

[RFC3740] T. Hardjono et B. Weis, "Architecture de sécurité de groupe de diffusion groupée", mars 2004. *(Information)*

7. Remerciements

Les personnes suivantes méritent notre reconnaissance et des remerciements pour leurs contributions, qui ont largement amélioré la présente spécification : Uri Meth, dont la connaissance de GSAKMP et des jetons a été très appréciée ainsi que son aide pour la soumission du présent document ; Peter Lough, Thomas Hardjono, Patrick McDaniel, et Pete Dinsmore pour leur travail sur les versions antérieures des jetons de politique ; George Gross pour son apport pour avoir un jeton de politique bien spécifié et extensible ; et Rod Fleischer pour avoir saisi les problèmes de mise en œuvre.

Les travaux techniques suivants ont influencé la conception du jeton de politique de sécurité de groupe : [HCLM00], [HCM01], et [HHMCD01]

Appendice A. Module ASN.1 du cœur de jeton de politique

PolicyToken {1.3.6.1.5.5.12.0.1}

ÉTIQUETTES IMPLICITES DE DÉFINITIONS ::=

DÉBUT

```
Token ::= SEQUENCE {
    tokenInfo     TokenID,
    registration   SEQUENCE DE Registration,
    rekey          SEQUENCE DE GroupMngmtProtocol,
    data           SEQUENCE DE DataProtocol
 }

-------------------------------------------------------------
-- Identifiant de jeton

TokenID ::= SEQUENCE {
    tokenDefVersion ENTIER (1),                  -- Jeton de politique de sécurité de groupe v1
    groupName         CHAINE D'OCTETS,
    edition            ENTIER FACULTATIF
}

LifeDate ::= CHOIX {
    gt          GeneralizedTime,
    utc         UTCTime,
    interval   ENTIER
}

-------------------------------------------------------------
-- Enregistrement

Registration ::= SEQUENCE {
    register     GroupMngmtProtocol,
    de-register GroupMngmtProtocol
}

-------------------------------------------------------------
-- GroupMngmtProtocol

GroupMngmtProtocol ::= CHOIX {
    aucun     NULL,
    Protocole pris en charge
  }

Protocol ::= SEQUENCE {
    protocol       IDENTIFIANT D'OBJET,
    protocolInfo  CHAINE D'OCTETS
}

  -------------------------------------------------------------
-- DataProtocol

DataProtocol ::= Protocol

  -------------------------------------------------------------

FIN
```

Appendice B. Politique de base GSAKMPv1

Cet appendice fournit les structures de données nécessaires pour quand les échanges GSAKMP sont utilisés comme GroupMngmtProtocol pour l'enregistrement, le désenregistrement, et/ou les SA de changement de clé. La présente spécification de politique de base de GSAKMP suppose une certaine familiarité avec GSAKMP.

B.1 Politique d'enregistrement de GSAKMPv1

Quand GSAKMP est utilisé dans le protocole de gestion de groupe pour l'enregistrement, l'identifiant d'objet suivant est utilisé dans le jeton principal.

IDENTIFIANT D'OBJET id-GSAKMPv1RegistrationProtocol ::= {1.3.6.1.5.5.12.3.1}

La politique d'enregistrement pour GSAKMP fournit
1) des informations sur les autorisations pour les rôles du groupe,
2) des informations de contrôle d'accès pour les membres du groupe,
3) les mécanismes utilisés dans le processus d'enregistrement,
4) des informations sur le transport que l'échange d'enregistrement GSAKMP va utiliser.

```
GSAKMPv1RegistrationInfo ::= SEQUENCE {
     joinAuthorization   JoinAuthorization,
     joinAccessControl  SEQUENCE DE AccessControl,
     joinMechanisms     JoinMechanisms,
     transport          Transport
}
```

B.1.1 Autorisation

joinAuthorization fournit des informations sur qui est autorisé à être un contrôleur de groupe/serveur de clés (GC/KS, *Group Controller Key Server*) et un sous GC/KS. Il peut aussi indiquer si il y a des limitations sur ceux qui peuvent envoyer des données dans un groupe.

```
JoinAuthorization ::= SEQUENCE {
     gCKS        GCKSName,
     subGCKS   SEQUENCE DE GCKSName FACULTATIF,
     senders      SenderAuthorization
}
```

Les informations d'autorisation sont sous la forme d'une liste de contrôle d'accès qui indique le nom de l'entité et les informations d'autorité de certification acceptables pour le certificat de l'entité.

```
GCKSName ::= SEQUENCE DE UserCAPair

UserCAPair ::= SEQUENCE {
     groupEntity   GSAKMPID,
     cA              CertAuth
}
```

groupEntity est défini par type et valeur. Les types sont indiqués par des entiers qui correspondent aux types d'identification GSAKMP. Quand une portion d'un type de nom défini est remplie avec un "*", cela indique un caractère générique, représentant tout choix valide pour un champ. Cela permet la spécification d'une règle d'autorisation qui est un ensemble de noms en rapports.

```
GSAKMPID ::= SEQUENCE {
     typeValue  ENTIER,
     typeData    CHAINE D'OCTETS
}
```

L'autorité de certificat est identifiée par l'identifiant de clé X.509 [RFC3280].
CertAuth ::= KeyIdentifier

Les envoyeurs au sein d'un groupe peuvent être tous (n'indiquant pas de restriction d'envoi) ou peuvent être une liste explicite des membres autorisés à envoyer des données.

```
SenderAuthorization ::= CHOIX {
    tous     [0] NULL,
    limité   [1] SEQUENCE EXPLICITE DE UserCAPair
```

```
}
```

B.1.2 Contrôle d'accès

joinAccessControl fournit des informations sur qui est autorisé à être membre du groupe. La liste de contrôle d'accès est mise en œuvre comme un ensemble de permissions que le membre doit satisfaire et d'une liste de règles de noms et l'autorité de certification que chacun doit satisfaire. De plus, une liste des exclusions de la liste peut être fournie.

```
AccessControl ::= SEQUENCE {
   permissions      [1] SEQUENCE EXPLICITE DE Permission FACULTATIF,
   accessRule       [2] SEQUENCE EXPLICITE DE UserCAPair,
   exclusionsRule  [3] SEQUENCE EXPLICITE DE UserCAPair FACULTATIF
 }
```

Les permissions initialement disponibles sont un ensemble abstrait de niveaux numériques qui peuvent être interprétés à l'intérieur d'une communauté.

```
Permission ::= CHOIX {
      simplePermission [1] SimplePermission
 }

SimplePermission ::= ENUMERATION {
      un(1),
      deux(2),
      trois(3),
      quatre(4),
      cinq(5),
      six(6),
      sept(7),
      huit(8),
      neuf(9)
    }
```

B.1.3 Mécanismes de jonction

Les choix de mécanisme GSAKMP admissibles pour un groupe particulier sont spécifiés dans joinMechanisms. Tout ensemble de JoinMechanism est acceptable du point de vue de la politique.

```
JoinMechanisms ::=  SEQUENCE DE JoinMechanism
```

Chaque ensemble de mécanismes utilisé dans l'enregistrement GSAKMP peut être spécifié comme un ensemble défini explicitement ou comme une suite de sécurité pré-définie.

```
JoinMechanism ::= CHOIX {
      alaCarte [0] Mechanisms,
      suite      [1] SecuritySuite
}
```

B.1.3.1 Ensemble "alaCarte"

Dans un ensemble défini explicitement -- ou "alaCarte" -- , un mécanisme est défini pour la signature, l'algorithme d'échange de clés, l'algorithme d'enveloppement de clés, le type des données d'accusé de réception, et les données de configuration pour le réglage des fins de temporisation.

```
Mechanisms ::=  SEQUENCE {
      signatureDef   SigDef,
      kEAlg          KEAlg,
      keyWrap        KeyWrap,
      ackData        AckData,
```

```
    opInfo      OpInfo
  }
```

La définition de signature exige la spécification de l'algorithme de signature pour la signature du message. L'ENTIER qui définit le choix correspond au type de signature GSAKMP.

```
SigDef ::= SEQUENCE {
   sigAlgorithmID   ENTIER,
   hashAlgorithmID ENTIER
}
```

L'ENTIER qui correspond à l'algorithme de hachage *(hashAlgorithm)* va se transposer en la valeur de type de hachage de nom occasionnel GSAKMP. Cet algorithme est utilisé pour calculer le nom occasionnel combiné.

L'algorithme d'échange de clés exige un entier pour définir le type de création de clé GSAKMP et peut exiger des données supplémentaires par type.

```
KEAlg ::= SEQUENCE {
     keyExchangeAlgorithmID    ENTIER,
     keyExchangeAlgorithmData CHAINE D'OCTETS FACULTATIF
 }
```

keyWrap est l'algorithme qui est utilisé pour envelopper la ou les clés de groupe et le jeton de politique (si il est inclus). L'entier correspond au type de chiffrement GSAKMP.

```
KeyWrap ::= ENTIER
```

Des données peuvent éventuellement être retournées dans un message ACK/échec de téléchargement de clé GSAKMP. Le type de données requises par un groupe est spécifié par AckData. Aucun champ de ce type n'est actuellement pris en charge ni exigé.

```
AckData ::= CHOIX {
   aucun [0] NUL
 }
```

OpInfo fournit des données de configuration pour l'opération d'enregistrement GSAKMP. timeOut indique la durée écoulée avant qu'un message envoyé soit considéré être égaré ou perdu. Il est spécifié comme le type d'horodatage LifeDate, défini précédemment dans le jeton central. terse informe un GC/KS de si le groupe devrait opérer en mode concis (VRAI) ou verbeux (FAUX). Le champ facultatif timestamp indique si on utilise pour la protection contre la répétition un horodatage (VRAI) ou un nom occasionnel (FAUX). Si le champ est absent, l'utilisation de noms occasionnels est le mode par défaut pour l'enregistrement GSAKMP.

```
OpInfo ::= SEQUENCE {
  timeOut     LifeDate,
  terse       BOOLEEN,
  timestamp   BOOLEEN FACULTATIF
}
```

B.1.3.2 Suite de sécurité

Si le choix du mécanisme pour la jonction est un problème de sécurité prédéfini, il est alors identifié par un IDENTIFIANT D'OBJET (OID). D'autres suites de sécurité peuvent être définies ailleurs par la spécification et l'enregistrement d'un OID.

```
SecuritySuite ::= IDENTIFIANT D'OBJET
```

L'OID pour la suite de sécurité 1, définie dans la spécification GSAKMPv1 est :

```
IDENTIFIANT D'OBJET id-securitySuiteOne ::= {1.3.6.1.5.5.12.2.1}
```

B.1.4 Transport

transport indique quel protocole GSAKMP devrait utiliser. Le choix de udpRTJtcpOther indique que le message GSAKMP Demande d'adhésion est porté par UDP et que tous les autres messages d'établissement de groupe sont portés par TCP.

```
Transport ::= CHOIX {
     tcp                 [0] NULL,
     udp                 [1] NULL,
     udpRTJtcpOther  [2] NULL
 }
```

B.2 Module ASN.1 d'enregistrement GSAKMPv1

```
GSAKMPv1RegistrationSA {1.3.6.1.5.5.12.0.2}

ÉTIQUETTES IMPLICITES DE DÉFINITIONS ::=

DÉBUT
  EXPORTE
      GCKSName;

  IMPORTE
    LifeDate
      DE PolicyToken {1.3.6.1.5.5.12.0.1}

    KeyIdentifier
      DE PKIX1Implicit88 { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-
                           mod(0) id-pkix1-implicit(19) };

IDENTIFIANT D'OBJET id-GSAKMPv1RegistrationProtocol ::= {1.3.6.1.5.5.12.7}

GSAKMPv1RegistrationInfo ::= SEQUENCE {
    joinAuthorization       JoinAuthorization,
    joinAccessControl       SEQUENCE DE AccessControl,
    joinMechanisms          JoinMechanisms,
    transport               Transport
   }

JoinAuthorization ::= SEQUENCE {
    gCKS        GCKSName,
    subGCKS   SEQUENCE DE GCKSName FACULTATIF,
    senders     SenderAuthorization
   }

GCKSName ::= SEQUENCE DE UserCAPair

   UserCAPair ::= SEQUENCE {
    groupEntity   GSAKMPID,
    cA            CertAuth
   }

CertAuth ::= KeyIdentifier

SenderAuthorization ::= CHOIX {
    tous      [0] NULL,
    limité    [1] SEQUENCE EXPLICITE DE UserCAPair
   }

AccessControl ::= SEQUENCE {
    permissions    [1] SEQUENCE EXPLICITE DE Permission FACULTATIF,
    accessRule     [2] SEQUENCE EXPLICITE DE UserCAPair,
```

```
    exclusionsRule [3] SEQUENCE EXPLICITE DE UserCAPair FACULTATIF
}

Permission ::= CHOIX {
    simplePermission [1] SimplePermission
}

SimplePermission ::= ENUMERE {
    un(1),
    deux(2),
    trois(3),
    quatre(4),
    cinq(5),
    six(6),
    sept(7),
    huit(8),
    neuf(9)
  }

GSAKMPID ::= SEQUENCE {
    typeValue   ENTIER,
    typeData    CHAINE D'OCTETS
  }

JoinMechanisms ::=  SEQUENCE DE JoinMechanism

JoinMechanism ::= CHOIX {
    alaCarte  [0] Mechanisms,
    suite       [1] SecuritySuite
 }

Mechanisms ::=  SEQUENCE {
    signatureDef  SigDef,
    kEAlg          KEAlg,
    keyWrap       KeyWrap,
    ackData       AckData,
    opInfo         OpInfo
 }

SecuritySuite ::= IDENTIFIANT D'OBJET

-- SUITE DE SÉCURITÉ UN --
IDENTIFIANT D'OBJET id-securitySuiteOne ::= {1.3.6.1.5.5.12.2.1}

SigDef ::= SEQUENCE {
    sigAlgorithmID   ENTIER,
    hashAlgorithmID ENTIER
 }

KEAlg ::= SEQUENCE {
    keyExchangeAlgorithmID    ENTIER,
    keyExchangeAlgorithmData CHAINE D'OCTETS FACULTATIF
 }

KeyWrap ::= ENTIER

AckData ::= CHOIX {
    aucun [0] NUL
  }

OpInfo ::= SEQUENCE {
```

```
    timeOut    LifeDate,
    terse      BOOLEEN,
    timestamp BOOLEEN FACULTATIF
}

Transport ::= CHOIX {
    tcp                [0] NUL,
    udp                [1] NUL,
    udpRTJtcpOther [2] NUL
}

FIN
```

B.3 Politique de désenregistrement de GSAKMPv1

Le désenregistrement GSAKMP fournit une méthode pour notifier à un (S-)GC/KS qu'un membre a besoin de quitter un groupe. Quand GSAKMP est le protocole de désenregistrement pour le groupe, l'identifiant d'objet suivant est utilisé dans le jeton cœur.

IDENTIFIANT D'OBJET id-GSAKMPv1DeRegistrationProtocol ::= {1.3.6.1.5.5.12.3.2}

La politique de désenregistrement fournit les mécanismes nécessaires pour les messages d'échange de désenregistrement, comme l'indication de si l'échange est à faire en mode concis (VRAI) ou en mode verbeux (FAUX) et du transport utilisé pour les messages de désenregistrement GSAKMP.

```
GSAKMPv1DeRegistrationInfo ::= SEQUENCE {
     leaveMechanisms  SEQUENCE DE LeaveMechanisms,
     terse                 BOOLEEN,
     transport             Transport
}
```

La politique qui dicte les mécanismes nécessaires pour l'échange de désenregistrement est définie par leaveMechanisms. Ce champ est spécifié comme :

```
LeaveMechanisms ::= SEQUENCE {
     sigAlgorithm     ENTIER,
     hashAlgorithm  ENTIER,
     cA                    KeyIdentifier
}
```

L'ENTIER correspondant à sigAlgorithm va se transposer en une valeur de type de signature GSAKMP. Cet algorithme va être utilisé pour la signature du message.

L'ENTIER qui correspond à hashAlgorithm va se transposer en une valeur de type de hachage de nom occasionnel GSAKMP. Cet algorithme est utilisé pour calculer le nom occasionnel combiné.

cA représente un point de confiance que doit certifier le certificat du signataire. Il est identifié par le type KeyIdentifier [RFC3280] de l'infrastructure de clé publique pour certificats X.509 (PKIX, *Public Key Infrastructure for X.509 Certificates*).

transport indique le transport attendu pour les messages de désenregistrement GSAKMP. Initialement, UDP ou TCP vont être la politique pour un groupe.

```
Transport ::= CHOIX {
     tcp [0] NUL,
     udp [1] NUL
}
```

B.4 Module ASN.1 de désenregistrement GSAKMPv1

```
GSAKMPv1DeRegistrationSA {1.3.6.1.5.5.12.0.3}

ÉTIQUETTES IMPLICITES DE DÉFINITIONS ::=

DÉBUT

IMPORTE
   KeyIdentifier
     DE PKIX1Implicit88 { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-
                          mod(0) id-pkix1-implicit(19) };

IDENTIFIANT D'OBJET id-GSAKMPv1DeRegistrationProtocol ::= {1.3.6.1.5.5.12.3.2}

GSAKMPv1DeRegistrationInfo ::= SEQUENCE {
   leaveMechanisms  SEQUENCEDE LeaveMechanisms,
   transport            Transport
}

LeaveMechanisms ::= SEQUENCE {
   sigAlgorithm     ENTIER,
   hashAlgorithm   ENTIER,
   cA                  KeyIdentifier
}

Transport ::= CHOIX {
   tcp  [0] NUL
   udp [1] NUL
}

FIN
```

B.5 Politique de changement de clés de GSAKMPv1

Quand GSAKMP est utilisé comme protocole de changement de clé pour le groupe, l'identifiant d'objet suivant devrait être utilisé dans le jeton cœur comme protocole de changement de clé :

```
IDENTIFIANT D'OBJET id-GSAKMPv1Rekey ::= {1.3.6.1.5.5.12.0.4}
```

La politique de changement de clé de GSAKMP fournit les informations d'autorisation, les mécanismes pour les messages de changement de clé de GSAKMP, des indicateurs des définitions d'événement de changement de clé pour quand le GC/KS devrait envoyer un message de changement de clé, le protocole ou méthode que l'événement de changement de clé va utiliser, l'intervalle de changement de clé qui va permettre à un membre de reconnaître une défaillance du processus de changement de clé, un indicateur de fiabilité qui définit la méthode que le changement de clé va utiliser pour augmenter la probabilité d'une livraison du changement de clé (si il y en a un) et finalement une indication de la façon dont les GC/KS subordonnés vont traiter le changement de clés. Cette politique décrit aussi les méthodes spécifiques de la politique de changement de clé "None" et "GSAKMP LKH REKEY".

```
GSAKMPv1RekeyInfo ::= SEQUENCE {
     authorization     RekeyAuthorization,
     mechanism        RekeyMechanisms,
     rekeyEventDef  RekeyEventDef,
     rekeyMethod     RekeyMethod,
     rekeyInterval     LifeDate,
     reliability          Reliability,
     subGCKSInfo    SubGCKSInfo
}
```

B.5.1 Autorisation de changement de clé

RekeyAuthorization ::= GCKSName

B.5.2. Mécanismes de changement de clé

La politique qui dicte les mécanismes nécessaires pour le traitement des message de changement de clé est définie par RekeyMechanisms. Ce champ est spécifié comme :

```
RekeyMechanisms ::= SEQUENCE {
    sigAlgorithm    ENTIER,
    hashAlgorithm  ENTIER
}
```

L'ENTIER qui correspond à sigAlgorithm va se transposer en une valeur de type de signature GSAKMP. Cet algorithme va être utilisé pour la signature du message.

L'ENTIER qui correspond à hashAlgorithm va se transposer en la valeur du type de hachage du nom occasionnel GSAKMP. Cet algorithme est utilisé pour calculer le nom occasionnel combiné.

B.5.3 Définition d'événement de changement de clé

La définition d'événement de changement de clé fournit des informations au GC/KS sur les exigences du système pour l'envoi des messages de changement de clé. Cela permet la définition de l'événement de changement de clé dans le temps ainsi que les caractéristiques qui dépendent de l'événement (par exemple un certain nombre de notifications de désenregistrement) ou une combinaison des deux (par exemple, après x désenregistrements ou 24 heures, quel que soit celui qui arrive en premier).

```
RekeyEventDef ::= CHOIX {
    aucun           [0]  NUL,                   -- ne jamais changer de clé
    timeOnly        [1]  LifeDate,              -- changer de clé toutes les x unités
    event           [2]  ENTIER,                - changer de clé après x événements
    timeAndEvent  [3]  TimeAndEvent
}
```

LifeDate spécifie la durée maximum pendant laquelle un groupe devrait exister entre les changements de clé. Cela n'exige pas de synchronisation d'horloge car c'est utilisé par rapport à une horloge locale (l'horloge d'un GC/KS pour l'envoi des messages de changement de clé ou d'un membre pour déterminer si un message a été manqué).

L'ENTIER qui correspond à l'événement est un indicateur du nombre d'événements qu'un groupe devrait subir avant l'envoi d'un message de changement de clé. Cela définit les événements entre les changements de clé. Un exemple d'événement pertinent est la notification de désenregistrement.

TimeAndEvent est défini comme un couple des politiques de LifeDate et d'Entier.

```
TimeAndEvent ::= SEQUENCE {
    temps       LifeDate,                   -- changer de clé après x unités de temps OU
    événement  ENTIER                       -- après x événements.
   }
```

B.5.4 Méthodes de changement de clé

La méthode de changement de clé définit la politique de réalisation du changement de clé. Ce champ est spécifié comme :

```
RekeyMethod ::= SEQUENCE {
    rekeyMethodType  IDENTIFIANT D'OBJET,
    rekeyMethodInfo    CHAINE D'OCTETS
}
```

rekeyMethodType va définir la méthode de changement de clé à utiliser par le groupe.

rekeyMethodInfo va fournir aux GM les informations dont ils ont besoin pour opérer dans le mode de changement de clé correct.

B.5.4.1 Méthode NONE de changement de clé

Un groupe défini comme fonctionnant sans protocole de changement de clé est pris en charge par le type de méthode de changement de clé *(rekeyMethodType)* NONE. Il n'y a pas de RekeyMethodNoneInfo associées à cette option.

IDENTIFIANT D'OBJET id-rekeyNone ::= {1.3.6.1.5.5.12.4.1}

RekeyMethodNoneInfo ::= NUL

B.5.4.2 Méthode de changement de clé GSAKMP LKH

La spécification du protocole GSAKMP a défini une interprétation du protocole de hiérarchie de clé logique (LKH, *Logical Key Hierarchy*) comme méthode de changement de clé. Cette méthode est prise en charge par les valeurs suivantes :

IDENTIFIANT D'OBJET id-rekeyMethodGSAKMPLKH ::= {1.3.6.1.5.5.12.4.2}

RekeyMethodGSAKMPLKHInfo ::= ENTIER

La méthode LKH de GSAKMP exige une valeur de type gsakmp pour identifier l'algorithme de chiffrement utilisé pour envelopper les clés. Cette valeur se transpose en le type de chiffrement GSAKMP.

B.5.5 Intervalle de changement de clé

L'intervalle de changement de clé définit le délai maximum que devrait voir le GM entre des changements de clé valides. Cela fournit un moyen pour s'assurer que le GM est synchronisé, du point de vue de la gestion de clé, avec le reste du groupe. Il est défini comme un horodatage.

B.5.6 Fiabilité de changement de clé

Le message de changement de clé dans le protocole GSAKMP est un seul message poussé. Il y a des soucis de fiabilité avec de tels messages sans accusé de réception (c'est-à-dire, d'échange de messages). La politique de fiabilité définit le mécanisme utilisé pour traiter ces soucis.

```
Reliability ::= SEQUENCE {
     reliabilityMechanism    IDENTIFIANT D'OBJET,
     reliabilityMechContent CHAINE D'OCTETS
 }
```

Le mécanisme de fiabilité est défini par un IDENTIFIANT D'OBJET et les informations nécessaires pour faire fonctionner ce mécanisme sont définies par reliabilityMechContent *(contenu de mécanisme de fiabilité)* et c'est une CHAINE D'OCTETS (comme précédemment).

B.5.6.1 Mécanisme None de fiabilité de changement de clé

Dans les réseaux de fiabilité adéquate, il peut n'être pas nécessaire d'utiliser un mécanisme d'amélioration de la fiabilité du message de changement de clé. Pour ces réseaux le mécanisme de fiabilité NONE est approprié.

IDENTIFIANT D'OBJET id-reliabilityNone ::= {1.3.6.1.5.5.12.5.1}

ReliabilityContentNone ::= NULL

B.5.6.2 Mécanisme Resend de fiabilité de changement de clé

Dans les réseaux d'une fiabilité inconnue ou discutable, il peut être nécessaire d'utiliser un mécanisme pour améliorer la fiabilité du message de changement de clé. Pour ces réseaux, le mécanisme de fiabilité RESEND est potentiellement approprié. Ce mécanisme fait que le GC/KS envoie de façon répétée le même message.

IDENTIFIANT D'OBJET id-reliabilityResend ::= {1.3.6.1.5.5.12.5.2}

ReliabilityResendInfo ::= ENTIER

La valeur d'ENTIER dans ReliabilityResendInfo indique le nombre de fois que le message devrait être envoyé.

B.5.6.3 Mécanisme Post de fiabilité de changement de clé

Un autre mécanisme de fiabilité est d'envoyer le message de changement de clé sur un service qui va le rendre généralement disponible. C'est la méthode reliabilityPost :

IDENTIFIANT D'OBJET id-reliabilityPost ::= {1.3.6.1.5.5.12.5.3}

ReliabilityContentPost ::= IA5String

La chaîne IA5String associée à ReliabilityPost est l'identifiant du site d'envoi et du message de changement de clé.

B.5.7 Politique de fonctionnement réparti

La politique qui dicte les relations entre le GC/KS et les S-GC/KS pour les opérations réparties est définie par SubGCKSInfo. Il est défini comme un couple de subGCKSScheme et des informations relatives à ce schéma dans sGCKSContent.

```
SubGCKSInfo ::= SEQUENCE {
     subGCKSScheme  IDENTIFIANT D'OBJET,
     sGCKSContent      CHAINE D'OCTETS
 }
```

B.5.7.1 Pas de fonctionnement réparti

Si le groupe ne va pas utiliser de S-GC/KS, ce schéma va être SGCKSSchemeNone.

IDENTIFIANT D'OBJET id-subGCKSSchemeNone ::= {1.3.6.1.5.5.12.6.1}

SGCKSNoneContent ::= NULL

B.5.7.2 Mode réparti autonome

Si le groupe va utiliser des S-GC/KS comme défini dans la spécification GSAKMP comme mode Autonome, alors ce schéma va être SGCKSAutonomous.

IDENTIFIANT D'OBJET id-subGCKSSchemeAutonomous ::= {1.3.6.1.5.5.12.6.2}

```
SGCKSAutonomous ::= SEQUENCE {
     authSubs  GCKSName,
     domain     CHAINE D'OCTETS FACULTATIF
    }
```

Les informations de politique nécessaires pour le mode autonome sont une liste de S-GC/KS autorisés et les restrictions sur qui elles peuvent desservir. Le champ domaine qui représente ces restrictions est NUL pour cette version.

B.6 Module ASN.1 de politique de changement de clé de GSAKMPv1

```
GSAKMPv1RekeySA {1.3.6.1.5.5.12.0.4}

ÉTIQUETTES IMPLICITES DE DÉFINITIONS ::=

DÉBUT

IMPORTE
      GCKSName
        DE GSAKMPv1RegistrationSA  {1.3.6.1.5.5.12.0.2}
      LifeDate
        DE PolicyToken  {1.3.6.1.5.5.12.0.1};

IDENTIFIANT D'OBJET id-GSAKMPv1Rekey ::= {1.3.6.1.5.5.12.0.4}

GSAKMPv1RekeyInfo ::= SEQUENCE {
    authorization      RekeyAuthorization,
    mechanism          RekeyMechanisms,
    rekeyEventDef   RekeyEventDef,          -- dit au GCKS quand changer de clés
    rekeyMethod        RekeyMethod,
    rekeyInterval      LifeDate,            -- le membre sait quand se joindre
    reliability        Reliability,         -- quel mécanisme va être utilisé pour augmenter la probabilité de
                                            livraison du changement de clés
    subGCKSInfo     SubGCKSInfo             -- de quels subordonnés le GCKS a besoin
 }

RekeyAuthorization ::= GCKSName

RekeyMechanisms ::= SEQUENCE {
    sigAlgorithm     ENTIER,
    hashAlgorithm    ENTIER
 }

RekeyEventDef ::= CHOIX {
    none             [0] NULL,              -- ne change jamais de clés
    timeOnly         [1] EXPLICIT LifeDate, -- change de clés toutes les x unités
    event            [2] ENTIER,            -- change de clés après x événements
    timeAndEvent [3] TimeAndEvent
 }

TimeAndEvent ::= SEQUENCE {
    time   LifeDate,                        -- change de clés après x unités de temps OU x événements
    event  ENTIER   --
 }

RekeyMethod ::= SEQUENCE {
    rekeyMethodType  IDENTIFIANT D'OBJET,
    rekeyMethodInfo    CHAINE D'OCTETS
 }

-- MÉTHODE DE CHANGEMENT DE CLÉ NONE --

IDENTIFIANT D'OBJET  id-rekeyNone ::= {1.3.6.1.5.5.12.4.1}

RekeyMethodNoneInfo ::= NULL

-- MÉTHODE DE CHANGEMENT DE CLÉ GSAKMP LKH --

 IDENTIFIANT D'OBJET id-rekeyMethodGSAKMPLKH::= {1.3.6.1.5.5.12.4.2}
```

```
RekeyMethodGSAKMPLKHInfo ::= ENTIER     -- valeur de type gsakmp pour le mécanisme d'enveloppement

Reliability ::= SEQUENCE {
  reliabilityMechanism     IDENTIFIANT D'OBJET,
  reliabilityMechContent  CHAINE D'OCTETS
 }

-- MÉCANISME DE FIABILITÉ NONE --

IDENTIFIANT D'OBJET id-reliabilityNone ::= {1.3.6.1.5.5.12.5.1}

ReliabilityContentNone ::= NULL

-- MÉCANISME DE FIABILITÉ RESEND --

IDENTIFIANT D'OBJET id-reliabilityResend ::= {1.3.6.1.5.5.12.5.2}

ReliabilityResendInfo ::= ENTIER              -- nombre de fois que le message Rekey devrait être envoyé

-- MÉCANISME DE FIABILITÉ POST --

IDENTIFIANT D'OBJET  id-reliabilityPost ::= {1.3.6.1.5.5.12.5.3}

ReliabilityContentPost ::= IA5String

SubGCKSInfo ::= SEQUENCE {
  subGCKSScheme  IDENTIFIANT D'OBJET,
  sGCKSContent      CHAINE D'OCTETS
 }

IDENTIFIANT D'OBJET  id-subGCKSSchemeNone ::= {1.3.6.1.5.5.12.6.1}

SGCKSNoneContent ::= NULL

IDENTIFIANT D'OBJET id-subGCKSSchemeAutonomous ::= {1.3.6.1.5.5.12.6.2}

SGCKSAutonomous ::= SEQUENCE {
  authSubs  GCKSName,
  domain    CHAINE D'OCTETS FACULTATIF
 }

FIN
```

Appendice C. Politique d'application de sécurité des données

La SA Données fournit les structures de données nécessaires pour la protection des données échangées entre les membres du groupe. Cet appendice définit les structures de données nécessaires pour une simple application générique de sécurité utilisant des mécanismes fixes de sécurité. Une telle SA de données exige seulement que les clés livrées par les protocoles d'enregistrement et de changement de clés soient transposées dans le service qui les utilise.

C.1 Politique générique des données

La politique générique de données a l'identifiant suivant :

IDENTIFIANT D'OBJET id-genericDataSA :: = {1.3.6.1.5.5.12.7.1}

Si un mécanisme d'authentification est utilisé au sein de l'application de sécurité, l'identifiant de clé (kMKeyID) utilisé dans le protocole de gestion de clé est donné, ainsi qu'une date facultative d'expiration de clé. De même, si un mécanisme de

chiffrement est utilisé au sein de l'application de sécurité, l'identifiant de chiffrement de clé est donné, ainsi qu'une date facultative d'expiration de clé (keyExpirationDate).

```
GenericDataSAInfo ::= SEQUENCE {
   authentication [0] EXPLICIT KeyInfo FACULTATIF,
   encryption      [1] EXPLICIT KeyInfo FACULTATIF
 }

KeyInfo ::= SEQUENCE{
  kMKeyID              CHAINE D'OCTETS,
  keyExpirationDate   LifeDate FACULTATIF
 }
```

C.2 Module ASN.1 de politique générique des données

```
GenericDataSA {1.3.6.1.5.5.12.0.5}

ÉTIQUETTES IMPLICITES DE DÉFINITIONS ::=

DÉBUT

-- APPLICATION DE DONNÉES : Générique
-- Cette spécification de jeton est pour les applications de données avec des mécanismes fixes de sécurité. De telles
applications de données ont seulement besoin d'une transposition des étiquettes d'identification de clé de protocole de
gestion en service de sécurité.

IMPORTE
   LifeDate DE PolicyToken {1.3.6.1.5.5.12.0.1}

KeyIdentifier DE PKIX1Implicit88 { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5)
                                   pkix(7) id-mod(0) id-pkix1-implicit(19) };

IDENTIFIANT D'OBJET id-genericDataSA ::= {1.3.6.1.5.5.12.7.1}

GenericDataSAInfo ::= SEQUENCE {
    authentication [0] EXPLICITE KeyInfo FACULTATIF,
    encryption      [1] EXPLICITE KeyInfo FACULTATIF
 }

KeyInfo ::= SEQUENCE{
    kMKeyID              CHAINE D'OCTETS,
    keyExpirationDate  LifeDate FACULTATIF
}

FIN
```

Adresse des auteurs

Andrea Colegrove
SPARTA, Inc.
7110 Samuel Morse Drive
Columbia, MD 21046
USA
téléphone : (443) 430-8014
mél : acc@sparta.com

Hugh Harney
SPARTA, Inc.
7110 Samuel Morse Drive
Columbia, MD 21046
USA
téléphone : (443) 430-8032
mél : hh@sparta.com

Déclaration complète de droits de reproduction

Remerciement
Le financement de la fonction d'édition des RFC est fourni par l'activité de soutien administatif (IASA) de l'IETF.