Groupe de travail Réseau
Request for Comments : 5275
Catégorie : Sur la voie de la normalisation

S. Turner, IECA
juin 2008
Traduction Claude Brière de L'Isle

Gestion et distribution de clés symétriques avec la CMS

Statut du présent mémoire

Le présent document spécifie un protocole de l'Internet sur la voie de la 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 présent document décrit un mécanisme pour gérer (c'est-à-dire, établir, distribuer, et changer) les clés utilisées avec les algorithmes de chiffrement symétriques. Il définit aussi un mécanisme pour organiser les utilisateurs en groupes pour prendre en charge la distribution du contenu chiffré en utilisant des algorithmes de chiffrement symétriques. Le mécanisme utilise le protocole de syntaxe de message cryptographique (CMS, *Cryptographic Message Syntax*) et le protocole de gestion de certificat sur la CMS (CMC, *Certificate Management over CMS*) pour gérer les clés symétriques. Tout membre du groupe peut ensuite utiliser ces clés partagées distribuées pour déchiffrer d'autres objets chiffrés selon la CMS avec la clé symétrique. Ce mécanisme a été développé pour la prise en charge par les agents de liste de messagerie (MLA, *Mail List Agent*) des extensions de messagerie Internet multi objets sécurisée (S/MIME, *Secure/Multipurpose Internet Mail Extensions*).

Table des matières

1. Introduction

Avec la croissance constante des communications électroniques sécurisées (par exemple, S/MIME [RFC3851]), les utilisateurs exigent un mécanisme pour distribuer des données chiffrées à plusieurs receveurs (c'est-à-dire, un groupe d'utilisateurs). Il y a essentiellement deux façons de chiffrer les données pour les receveurs : en utilisant des algorithmes asymétriques avec des certificats de clé publique (PKC, *Public Key Certificate*) ou des algorithmes symétriques avec des clés symétriques.

Avec les algorithmes asymétriques, le générateur forme une clé de chiffrement de contenu (CEK, *Content-Encryption Key*) déterminée par le générateur, et chiffre le contenu, en utilisant un algorithme symétrique. Ensuite, en utilisant un algorithme asymétrique et la PKC du receveur, le générateur génère des informations par receveur qui soit (a) chiffrent la CEK pour un receveur particulier (CHOIX ktri RecipientInfo) soit (b) transfèrent suffisamment de paramètres pour permettre à un receveur particulier de générer indépendamment la même KEK (CHOIX kari RecipientInfo). Si le groupe est grand, le traitement des informations par receveur peut prendre assez longtemps, sans mentionner le temps requis pour collecter et valider les PKC pour chaque receveur. Chaque receveur identifie ses informations et utilise la clé privée associée à la clé publique de sa PKC pour déchiffrer la CEK et donc obtenir l'accès au contenu chiffré.

Avec les algorithmes symétriques, le processus de génération est légèrement différent. Au lieu d'utiliser des PKC, le générateur utilise une clé de chiffrement de clé (KEK, *Key-Encryption Key*) secrète distribuée précédemment pour chiffrer la CEK (CHOIX kekri RecipientInfo). Une seule copie de la CEK chiffrée est nécessaire parce que tous les receveurs ont déjà la KEK partagée nécessaire pour déchiffrer la CEK et donc obtenir l'accès au contenu chiffré.

Les techniques pour protéger la KEK partagée sortent du domaine d'application de ce document. Seuls les membres de la liste et le gestionnaire de la clé devraient avoir la KEK afin de maintenir la confidentialité. Le contrôle d'accès aux informations protégées par la KEK est déterminé par l'entité qui chiffre les informations, car tous les membres du groupe y ont accès. Si l'entité qui effectue le chiffrement veut s'assurer qu'un sous ensemble du groupe n'obtienne pas l'accès aux informations, une KEK différente devrait être utilisée (partagée seulement par ce plus petit groupe) ou des algorithmes asymétriques devraient être utilisés.

1.1 Conventions utilisées dans ce document

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" en majuscules dans ce document sont à interpréter comme décrit dans le BCP 14, [RFC2119].

1.2 Applicabilité à la messagerie électronique

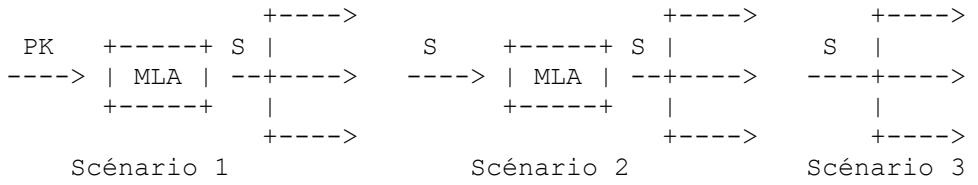
Le public principalement visé par ce mécanisme de distribution est la messagerie électronique. Les listes de distribution, parfois appelées des listes de diffusion, servent à la distribution des messages aux receveurs abonnés à la liste de messagerie. Il y a deux modèles pour l'utilisation d'une liste de messagerie. Si l'origine est un membre de la liste de messagerie, il envoie les messages chiffrés avec la KEK partagée à la liste de messagerie (par exemple, listserv ou majordomo) et le message est distribué aux membres de la liste de messagerie. Si l'origine n'est pas un membre de la liste de messagerie (il n'a pas la KEK partagée) le générateur envoie le message (chiffré pour l'agent de liste de messagerie MLA, *Mail List Agent*)) au MLA, et ensuite le MLA utilise la KEK partagée pour chiffrer le message pour les membres. Dans l'un ou l'autre cas, les receveurs de la liste de messagerie utilisent la KEK précédemment distribuée-partagée pour déchiffrer le message.

1.3 Applicabilité aux répertoires

Les objets peuvent aussi être distribués via des serveurs de répertoire (par exemple, du protocole léger d'accès à un répertoire (LDAP, *Lightweight Directory Access Protocol*) des agents de système de répertoire X.500 (DSA), des serveurs fondés sur la Toile). Si un objet est mémorisé dans un répertoire chiffré avec un algorithme de clé symétrique, quiconque a la KEK partagée et l'accès à cet objet peut déchiffrer cet objet. L'objet chiffré et la KEK chiffrée partagée peuvent être mémorisés dans le répertoire.

1.4 Utilisation de la clé de groupe

Le présent document a été écrit en visant trois scénarios spécifiques : deux qui prennent des agents de liste de messagerie et un pour la distribution générale de message. Le scénario 1 décrit l'envoi par le générateur d'une clé publique (PK) protégée à un MLA qui utilise la ou les KEK partagées pour redistribuer le message aux membres de la liste. Le scénario 2 décrit l'envoi par le générateur d'un message protégé par une KEK partagée à un MLA qui redistribue ensuite le message aux membres de la liste (le MLA ajoute seulement des receveurs supplémentaires). La clé utilisée par le générateur pourrait être une clé partagée soit par tous les receveurs, soit juste par le membre et le MLA. Noter que si le générateur utilise une clé partagée seulement avec le MLA, celui-ci va devoir déchiffrer le message et le rechiffrer pour les receveurs de la liste. Les scénario 3 montre un générateur qui envoie un message protégé par une KEK partagée à un groupe de receveurs sans MLA intermédiaire.



2. Architecture

La Figure 1 décrit l'architecture de prise en charge de la distribution de clés symétriques. L'agent de liste de groupe (GLA, *Group List Agent*) prend en charge deux fonctions distinctes avec deux agents différents :

- L'agent de gestion de clé (KMA, *Key Management Agent*) qui est chargé de générer les KEK partagées.
- L'agent de gestion de groupe (GMA, *Group Management Agent*) qui est chargé de gérer la liste de groupe (GL, *Group List*) à laquelle les KEK partagées sont distribuées.

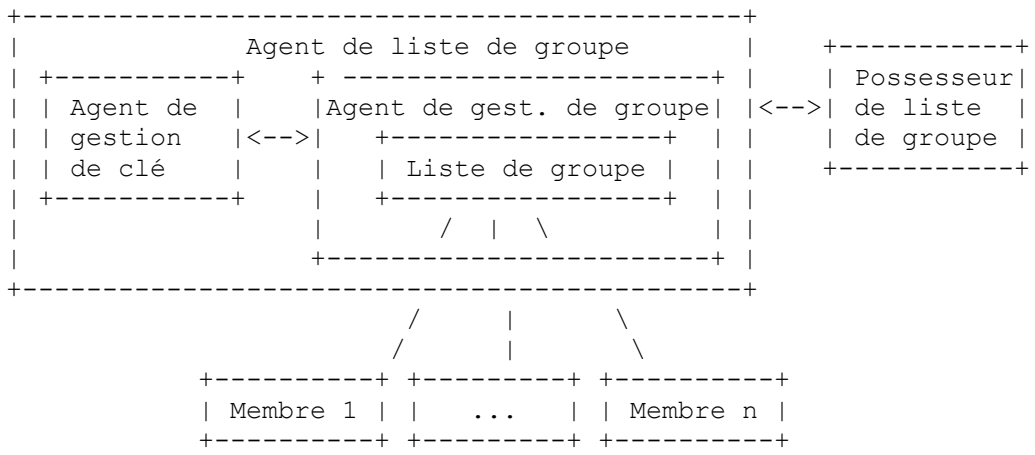


Figure 1 : Architecture de distribution de clés

Un GLA peut prendre en charge plusieurs KMA. Un GLA prend en général seulement en charge un GMA, mais le GMA peut prendre en charge plusieurs GL. Plusieurs KMA peuvent prendre en charge un GMA de la même façon que les GLA prennent en charge plusieurs KMA. Allouer un KMA particulier à une GL sort du domaine d'application de ce document.

La modélisation de mises en œuvre réelles de GL montre qu'il y a des GL très restrictives, où un humain détermine les membres de la GL, et des GL très ouvertes, où il n'y a pas de restriction sur l'appartenance à la GL. Pour prendre en charge ce spectre, le mécanisme décrit ici prend en charge les GL gérées (c'est-à-dire, où un contrôle d'accès est appliqué) et non gérées (c'est-à-dire, où aucun contrôle d'accès n'est appliqué). Le mécanisme de contrôle d'accès pour les listes gérées sort du domaine d'application de ce document. Note : si la distribution pour la liste est effectuée par une entité autre que le générateur (par exemple, un MLA qui distribue un message électronique) cette entité peut aussi appliquer les règles de contrôle d'accès.

Dans l'un et l'autre cas, la GL doit initialement être construite par une entité qu'on appelle ici le possesseur de la liste de groupe (GLO, *Group List Owner*). Il peut y avoir plusieurs entités qui "possèdent" la GL et à qui il est permis de faire des changements aux propriétés ou aux membres de la GL. Le GLO détermine si la GL va être gérée ou non gérée et est la

seule entité qui peut supprimer la GL. Le ou les GLO peuvent ou non être des membres de la GL. Le ou les GLO peuvent aussi établir des listes qui sont closes, où le GLO détermine seul les membres de la GL.

Bien que la Figure 1 décrive le GLA comme englobant les deux fonctions de KMA et de GMA, les deux fonctions pourraient être prises en charge par la même entité ou elles pourraient être prises en charge par deux entités différentes. Si deux entités sont utilisées, elles pourraient être situées sur une ou deux plateformes. Il y a cependant une relation étroite entre les fonction de KMA et de GMA. Si le GMA mémorise toutes les informations relevant des GL et si le KMA génère simplement les clés, un GMA corrompu pourrait causer des dégâts. Pour se protéger contre un GMA corrompu, le KMA serait forcé de faire une double vérification des demandes qu'il reçoit pour s'assurer que le GMA ne les a pas altérées. Cette duplication des vérifications fait peser une tache sur la fonctionnalité des deux composants ensemble. Pour cette raison, les interactions entre le KMA et le GMA sortent du domaine d'application de ce document.

Des mécanismes propriétaires peuvent être utilisés pour séparer les fonctions en renforçant la relation de confiance entre les deux entités. À partir de ce point, la distinction entre les deux agents ne sera plus discutée ; le terme de GLA va être utilisé pour traiter des deux fonctions. On devrait noter qu'un GLA corrompu peut toujours causer des dégâts.

3. Interactions de protocole

Il y a des mécanismes (par exemple, listserv et majordomo) pour gérer les Gl ; cependant, le présent document ne traite pas de la sécurisation de ces mécanismes, car ils ne sont pas normalisés. Il définit plutôt les interactions de protocole, comme décrites dans la Figure 2, utilisées par les membres de la GL, GLA, et GLO pour gérée les GL et distribuer les KEK partagées. Les interactions ont été divisées en messages d'administration et messages de distribution. Les messages administratifs sont les messages de demande et réponse nécessaires pour établir la GL, supprimer la GL, ajouter des membres à la GL, supprimer des membres de la GL, demander un changement de clé de groupe, ajouter des possesseurs à la GL, retirer des possesseurs de la GL, indiquer la compromission d'une clé de groupe, rafraîchir une clé de groupe, interroger le GLA, et mettre à jour les certificats de clés des membres et des possesseurs. Les messages de distribution sont les messages qui distribuent les KEK partagées. Les paragraphes qui suivent décrivent l'ASN.1 pour les messages d'administration et de distribution. La Section 4 décrit comment utiliser les messages d'administration, et la Section 5 décrit comment utiliser les messages de distribution.

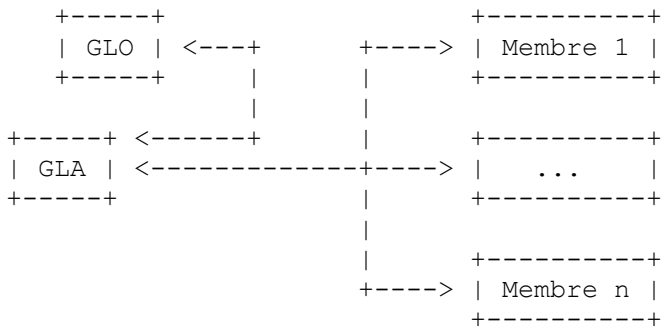


Figure 2 : Interactions de protocole

3.1 Attributs de commandes

Pour éviter de créer un protocole entièrement nouveau, le protocole de gestion de certificat sur la CMS (CMC) a été choisi comme fondement de ce protocole. La principale raison de ce choix était l'aspect de mise en couches fourni par CMC où un ou plusieurs attributs de commande sont inclus dans le message, protégés par la CMS, pour demander ou répondre à une action désirée. La structure de CMC PKIData est utilisée pour les demandes, et la structure de CMC PKIResponse est utilisée pour les réponses. Les types de contenu PKIData et PKIResponse sont alors encapsulés dans les SignedData ou EnvelopedData de la CMS, ou une combinaison des deux (voir au paragraphe 3.2). Les attributs de commande définis dans ce document sont les suivants :

Attribut de commande	**OID**	**Syntaxe**	
glUseKEK	id-skd 1	GLUseKEK	*(la GL utilise une KEK)*
glDelete	id-skd 2	GeneralName	*(nom général)*
glAddMember	id-skd 3	GLAddMember	*(ajoute un membre à la GL)*

glDeleteMember	id-skd 4	GLDeleteMember	*(suppime un mebre de la GL)*
glRekey	id-skd 5	GLRekey	*(changement de clé de la GL)*
glAddOwner	id-skd 6	GLOwnerAdministration	*(ajoute un membre de la GL)*
glRemoveOwner	id-skd 7	GLOwnerAdministration	*(supprime un membre de la GL)*
glkCompromise	id-skd 8	GeneralName	*(clé de GL compromise)*
glkRefresh	id-skd 9	GLKRefresh	*(rafraîchissement de clé de GL)*
glaQueryRequest	id-skd 11	GLAQueryRequest	*(demande d'interrogation du GLA)*
glaQueryResponse	id-skd 12	GLAQueryResponse	*(réponse d'interrogation du GLA)*
glProvideCert	id-skd 13	GLManageCert	*(fournir le certificat de la GL)*
glUpdateCert	id-skd 14	GLManageCert	*(mettre à jour le certificat de la GL)*
glKey	id-skd 15	GLKey	*(clé de GL)*

Dans les tableaux de conformité suivants, les en-têtes des colonnes ont la signification suivante : O pour origine, R pour receveur, et F pour transmission. Il y a trois types de mises en œuvre : les GLO, les GLA, et les membres de la GL. Le GLO est un composant facultatif, donc tous les messages GLO O et GLO R sont facultatifs, et les messages GLA F sont facultatifs. Le premier tableau inclut des messages que les mises en œuvre conformes DOIVENT prendre en charge. Le second tableau inclut des messages qui PEUVENT être mis en œuvre. Le second tableau devrait être interprété comme suit : si l'attribut de commande est mis en œuvre par un composant, alors il doit être mis en œuvre comme indiqué. Par exemple, si une mise en œuvre de GLA prend en charge l'attribut de commande glAddMember, alors elle DOIT prendre en charge la réception du message glAddMember. Noter que "-" signifie non applicable.

Exigé						
Exigence de mise en œuvre						
GLO		GLA			Membre de GL	Attribut
O	R	O	R	F	O R	de commande
PEUT	-	DOIT	-	PEUT	- DOIT	glProvideCert
PEUT	PEUT	-	DOIT	PEUT	DOIT -	glUpdateCert
-	-	DOIT	-	-	- DOIT	glKey

Facultatif						
Exigence de mise en œuvre						Attribut
GLO		GLA			Membre GL	de commande
O	R	O	R	F	O R	
PEUT	-	-	PEUT	-	- -	glUseKEK
PEUT	-	-	PEUT	-	- -	glDelete
PEUT	PEUT	-	DOIT	PEUT	DOIT -	glAddMember
PEUT	PEUT	-	DOIT	PEUT	DOIT -	glDeleteMember
PEUT	-	-	PEUT	-	- -	glRekey
PEUT	-	-	PEUT	-	- -	glAddOwner
PEUT	-	-	PEUT	-	- -	glRemoveOwner
PEUT	PEUT	-	DOIT	PEUT	DOIT -	glkCompromise
PEUT	-	-	DOIT	-	DOIT -	glkRefresh
PEUT	-	-	DEVRAIT	-	PEUT -	glaQueryRequest
-	PEUT	DEVRAIT	-	-	- PEUT	glaQueryResponse

glaQueryResponse est porté dans le type de contenu de CMC PKIResponse ; tous les autres attributs de commande sont portés dans le type de contenu de CMC PKIData. Une exception est glUpdateCert, qui peut être porté dans PKIData ou PKIResponse.

Les messages de succès et d'échec utilisent le CMC (voir le paragraphe 3.2.4).

3.1.1 La GL utilise une KEK

Le GLO utilise glUseKEK pour demander qu'une KEK partagée soit allouée à une GL. Les messages glUseKEK DOIVENT être signés par le GLO. L'attribut de commande glUseKEK a la syntaxe GLUseKEK :

```
GLUseKEK ::= SEQUENCE {
  glInfo             GLInfo,
```

```
  glOwnerInfo          SEQUENCE TAILLE (1..MAX) DE GLOwnerInfo,
  glAdministration     GLAdministration PAR DÉFAUT 1,
  glKeyAttributes      GLKeyAttributes FACULTATIF }

GLInfo ::= SEQUENCE {
  glName     GeneralName,
  glAddress  GeneralName }

GLOwnerInfo ::= SEQUENCE {
  glOwnerName     GeneralName,
  glOwnerAddress  GeneralName,
  certificate           Certificates FACULTATIF }

Certificates ::= SEQUENCE {
   pKC           [0] Certificate FACULTATIF,                    -- voir la [RFC5280]
     aC          [1] SEQUENCE TAILLE (1.. MAX) DE AttributeCertificate FACULTATIF,      -- voir la [RFC3281]
     certPath    [2] CertificateSet FACULTATIF }                -- d'après la [RFC3852]

-- CertificateSet et CertificateChoices sont inclus seulement pour illustration car ils sont importés de la [RFC3852].

CertificateSet ::= SET TAILLE (1..MAX) DE CertificateChoices

-- CertificateChoices prend en charge les certificats de clé publique X.509 dans certificates et les certificats d'attribut v2
dans v2AttrCert.

GLAdministration ::= ENTIER {
   non gérée  (0),
   géré       (1),
   clos       (2) }

GLKeyAttributes ::= SEQUENCE {
  rekeyControlledByGLO        [0] BOOLÉEN PAR DÉFAUT FAUX,
  recipientsNotMutuallyAware [1] BOOLÉEN PAR DÉFAUT VRAI,
  duration                    [2] ENTIER PAR DÉFAUT 0,
  generationCounter           [3] ENTIER PAR DÉFAUT 2,
  requestedAlgorithm          [4] AlgorithmIdentifier PAR DÉFAUT { id-aes128-wrap } }
```

Les champs dans GLUseKEK ont la signification suivante :

- glInfo indique le nom de la GL dans glName et l'adresse de la GL dans glAddress. glName et glAddress peuvent être les mêmes, mais ce n'est pas toujours le cas. Le nom et l'adresse DOIVENT tous deux être uniques pour un GLA.

- glOwnerInfo indique :
  - -- glOwnerName indique le nom du possesseur de la GL. Un des noms dans glOwnerName DOIT correspondre à un des noms dans le certificat (soit le nom distinctif de sujet, soit un des noms de remplacement du sujet) utilisé pour signer ces SignedData.PKIData créant la GL (c'est-à-dire, le signataire immédiat).
  - -- glOwnerAddress indique l'adresse du possesseur de la GL.
  - -- des certificats PEUVENT être inclus. Il contiennent les trois champs suivants :
    - --- certificates.pKC inclut le certificat de chiffrement pour le GLO. Il va être utilisé pour chiffrer les réponses pour le GLO.
    - --- certificates.aC PEUT être inclus pour porter tout certificat d'attribut (voir la [RFC3281]) associé au certificat de chiffrement du GLO inclus dans certificates.pKC.
    - --- certificates.certPath PEUT aussi être inclus pour porter des certificats qui pourraient aider le receveur à construire des chemins de certification valides pour le certificat fourni dans certificates.pKC et les certificats d'attribut fournis dans certificates.aC. Ces certificats sont facultatifs parce que ils pourraient être déjà inclus ailleurs dans le message (par exemple, dans la couche CMS externe).
  - -- glAdministration indique comment la GL devrait être administrée. Par défaut, la liste est gérée. Trois valeurs sont prises en charge pour glAdministration :
    - --- Non gérée : quand le GLO règle glAdministration à non gérée, il permet l'ajout et la suppression de membres prospectifs de la GL sans intervention du GLO.
    - --- Gérée : quand le GLO règle glAdministration à géré, il permet que des membres prospectifs demandent leur

ajout et leur suppression de la GL, mais la demande est redirigée par le GLA sur le GLO pour révision. Le GLO détermine si il va honorer la demande.

--- Close : quand le GLO règle glAdministration à close, il ne permet pas à des membres prospectifs de demander leur ajout ou suppression de la GL. Le GLA va seulement accepter des demandes glAddMember et glDeleteMember du GLO.

-- glKeyAttributes indique les attributs que le GLO veut que le GLA alloue à la KEK partagée. Si ce champ est omis, les changements de clé de la GL vont être contrôlés par le GLA, il est permis aux receveurs de se connaître les uns les autres, l'algorithme va être AES-128 (voir la Section 7) la KEK partagée va être valide pendant un mois calendaire (c'est-à-dire, du premier jour du mois au dernier jour du mois) et deux KEK partagées vont être distribuées initialement. Les champs dans glKeyAttributes ont la signification suivante :

--- rekeyControlledByGLO indique si les messages de changement de clé de la GL vont être générés par le GLO ou par le GLA. Par défaut c'est le GLA qui contrôle les changements de clés. Si le changement de clé de la GL est contrôlé par la GLA, les changements de clé de la GL vont continuer jusqu'à ce que le GLO supprime la GL ou passe le contrôle des changements de clé de la GL au GLO.

--- recipientsNotMutuallyAware indique que le GLO veut que le GLA distribue la KEK partagée individuellement pour chaque membre de la GL (c'est-à-dire, un message glKey séparé est envoyé à chaque receveur). Par défaut, un message glKey séparé n'est pas exigé.

Note : Cela prend en charge les listes où un membre ne connaît pas les identités des autres membres. Par exemple, une liste est configurée à accorder des permissions de soumission à seulement un membre. Tous les autres membres "écoutent". La politique de sécurité pour la liste ne permet pas aux membres de savoir qui d'autre est sur la liste. Si une glKey est construite pour tous les membres de la GL, des informations sur chaque membre peuvent être déduites des informations dans RecipientInfos. Pour s'assurer que le message glkey ne divulgue pas d'informations sur les autres receveurs, un message glKey séparé va être envoyé à chaque membre de la GL.

-- duration indique la période (en jours) durant laquelle la KEK partagée est considérée être valide. La valeur zéro (0) indique que la KEK partagée est valide pour un mois calendaire dans la zone horaire UTC Zoulou. Par exemple, si "duration" est zéro (0), si la KEK partagée de la GL est demandée le 24 juillet, la première clé va être valide jusqu'à la fin de juillet et la prochaine clé sera valide pendant tout le mois d'août. Si la valeur n'est pas zéro (0), la KEK partagée va être valide pendant le nombre de jours indiqué par la valeur. Par exemple, si la valeur de "duration" est sept (7) et si la KEK partagée est demandée le lundi mais n'est pas générée avant le mardi (13 mai 2008) les KEK partagées vont être valides du mardi (13 mai 2008) au mardi (20 mai 2008). L'heure exacte du jour est déterminée quand la clé est générée.

--- generationCounter indique le nombre de clés que le GLO veut que distribue le GLA. Pour assurer un fonctionnement ininterrompu de la GL, deux (2) KEK partagées au minimum DOIVENT être distribuées initialement. La seconde KEK partagée est distribuée avec la première KEK partagée, de sorte que quand la première KEK partagée n'est plus valide, la seconde clé peut être utilisée. Si le GLA contrôle les changements de clés, alors il indique aussi le nombre de KEK partagées que le GLO veut avoir en instance à chaque instant. Voir le paragraphe 4.5 et la Section 5 pour des détails sur le changement de clés.

--- requestedAlgorithm indique l'algorithme et tous les paramètres que le GLO veut que le GLA utilise avec la KEK partagée. Les paramètres sont portés via l'attribut SMIMECapabilities (voir la [RFC3851]). La Section 6 en dit plus sur les algorithmes.

3.1.2 Suppression de GL

Les GLO utilisent glDelete pour demander qu'une GL soit supprimée du GLA. L'attribut de commande glDelete a la syntaxe de GeneralName. Le message glDelete DOIT être signé par le GLO. Le nom de la GL à supprimer est inclus dans GeneralName :

```
DeleteGL ::= GeneralName
```

3.1.3 Ajout de membre de GL

Les GLO utilisent glAddMember pour demander l'ajout de nouveaux membres, et les membres prospectifs de la GL utilisent glAddMember pour demander leur propre ajout à la GL. Le message glAddMember DOIT être signé par le GLO ou le membre prospectif de la GL. L'attribut de commande glAddMember a la syntaxe de GLAddMember :

```
GLAddMember ::= SEQUENCE {
```

```
   glName      GeneralName,
   glMember  GLMember }

GLMember ::= SEQUENCE {
   glMemberName     GeneralName,
   glMemberAddress  GeneralName FACULTATIF,
   certificates           Certificates FACULTATIF }
```

Les champs dans GLAddMembers ont la signification suivante :

- glName indique le nom de la GL à laquelle le membre devrait être ajouté.

- glMember indique les particularités pour le membre de la GL. Les deux champs suivants doivent être uniques pour une GL donnée :
   -- glMemberName indique le nom du membre de la GL.
   -- glMemberAddress indique l'adresse du membre de la GL. Il DOIT être inclus.

Note : dans certaines instances, glMemberName et glMemberAddress peuvent être les mêmes, mais ce n'est pas toujours le cas.

   -- certificates DOIT être inclus. Il contient les trois champs suivants :
      --- certificates.pKC inclut le certificat de chiffrement du membre. Il va être utilisé, au moins initialement, pour chiffrer la KEK partagée pour ce membre. Si le message est généré par un membre prospectif de la GL, le pKC DOIT être inclus. Si le message est généré par un GLO, le pKC DEVRAIT être inclus.
      --- des certificates.aC PEUVENT être inclus pour porter tout certificat d'attribut (voir la [RFC3281]) associé au certificat de chiffrement du membre.
      --- des certificates.certPath PEUVENT aussi être inclus pour porter des certificats qui pourraient aider le receveur à construire des chemins de certification valides pour le certificat fourni dans certificates.pKC et les certificats d'attribut fournis dans certificates.aC. Ces certificats sont faculiatofs parce que qu'ils pourraient être déjà inclus ailleurs dans le message (par exemple, dans la couche de CMS externe).

3.1.4 Suppression de membre de GL

Les GLO utilisent glDeleteMember pour demander la suppression de membres de la GL, et les membres de la GL utilisent glDeleteMember pour demander leur propre retrait de la GL. Le message glDeleteMember DOIT être signé par le GLO ou par le membre de la GL. L'attribut de commande glDeleteMember a la syntaxe de GLDeleteMember :

```
GLDeleteMember ::= SEQUENCE {
  glName                GeneralName,
  glMemberToDelete  GeneralName }
```

Les champs dans GLDeleteMembers ont la signification suivante :
- glName indique le nom de la GL d'où le membre devrait être retiré.
- glMemberToDelete indique le nom ou l'adresse du membre à supprimer.

3.1.5 Changement de clé de GL

Les GLO utilisent glRekey pour demander un changement de clé de la GL. Le message glRekey DOIT être signé par le GLO. L'attribut de commande glRekey a la syntaxe de GLRekey :

```
GLRekey ::= SEQUENCE {
  glName                GeneralName,
  glAdministration      GLAdministration      FACULTATIF,
  glNewKeyAttributes  GLNewKeyAttributes FACULTATIF,
  glRekeyAllGLKeys    BOOLÉEN               FACULTATIF }

GLNewKeyAttributes ::= SEQUENCE {
  rekeyControlledByGLO        [0] BOOLÉEN FACULTATIF,
  recipientsNotMutuallyAware [1] BOOLÉEN FACULTATIF,
```

```
  duration                [2] ENTIER FACULTATIF,
  generationCounter       [3] ENTIER FACULTATIF,
  requestedAlgorithm      [4] AlgorithmIdentifier FACULTATIF }
```

Les champs dans GLRekey ont la signification suivante :

- glName indique le nom de la GL dont les clés sont à changer.

- glAdministration indique si il y a un changement de la façon dont la GL devrait être administrée. Voir au paragraphe 3.1.1 les trois options. Ce champ n'est inclus que si il y a un changement par rapport à la glAdministration précédemment enregistrée.

- glNewKeyAttributes indique si le changement de clés du GLO est contrôlé par le GLA ou la GL, quels algorithmes et paramètres le GLO souhaite utiliser, la durée de la clé, et combien de clés vont être produites. Le champ n'est inclus que si il y a un changement par rapport aux glKeyAttributes précédemment enregistrés.

- glRekeyAllGLKeys indique si le GLO veut que toutes les KEK partagées de la GL en instance soient changées. Si il est réglé à VRAI, alors toutes les KEK en instance DOIVENT être produites. Si il est réglé à FAUX, alors toutes les KEK en instance n'ont pas besoin d'être produites à nouveau.

3.1.6 Ajout de possesseur de GL

Les GLO utilisent glAddOwner pour demander qu'il soit permis à un nouveau GLO d'administrer la GL. Le message glAddOwner DOIT être signé par un GLO enregistré. L'attribut de commande glAddOwner a la syntaxe de GLOwnerAdministration :

```
GLOwnerAdministration ::= SEQUENCE {
  glName        GeneralName,
  glOwnerInfo  GLOwnerInfo }
```

Les champs dans GLAddOwners ont la signification suivante :
- glName indique le nom de la GL à laquelle le nouveau GLO devrait être associé.
- glOwnerInfo indique le nom, l'adresse, et les certificats du nouveau GLO. Comme le message inclut les noms des nouveaux GLO, les certificates.pKC DOIVENT être inclus, et il DOIVENT inclure le certificat de chiffrement du nouveau GLO.

3.1.7 Supprimer le possesseur de GL

Les GLO utilisent glRemoveOwner pour demander qu'un GLO soit dissocié de la GL. Le message glRemoveOwner DOIT être signé par un GLO enregistré. L'attribut de commande glRemoveOwner a la syntaxe de GLOwnerAdministration :

```
GLOwnerAdministration ::= SEQUENCE {
  glName        GeneralName,
  glOwnerInfo  GLOwnerInfo }
```

Les champs dans GLRemoveOwners ont la signification suivante :
- glName indique le nom de la GL de laquelle le GLO devrait être dissocié.
- glOwnerInfo indique le nom et l'adresse du GLO à retirer. Le champ certificates DEVRAIT être omis car il va être ignoré.

3.1.8 Compromission de clé de GL

Les membres de la GL et les GLO utilisent glkCompromise pour indiquer que la KEK partagée possédée a été compromise. L'attribut de commande glKeyCompromise a la syntaxe de GeneralName. Ce message est toujours redirigé par le GLA sur le GLO pour action ultérieure. Le glkCompromise PEUT être inclus dans une EnvelopedData générée avec la KEK partagée compromise. Le nom de la GL à laquelle la clé compromise est associée est placé dans GeneralName :

```
GLKCompromise ::= GeneralName
```

3.1.9 Rafraîchissement de clé de GL

Les membres de la GL utilisent glkRefresh pour demander que la KEK partagée leur soit redistribuée. L'attribut de commande glkRefresh a la syntaxe de GLKRefresh :

```
GLKRefresh ::= SEQUENCE {
  glName  GeneralName,
  dates       SEQUENCE TAILLE (1..MAX) DE Date }

Date ::= SEQUENCE {
  start  GeneralizedTime,
  end   GeneralizedTime FACULTATIF }
```

Les champs dans GLKRefresh ont la signification suivante :
- glName indique le nom de la GL pour laquelle le membre de la GL veut des KEK partagées.
- dates indique une gamme de dates pour les clés que veut le membre de la GL. Le champ start indique la première date que veut le membre de la GL et le champ end indique la dernière date. La date de fin PEUT être omise pour indiquer que le membre de la GL veut toutes les clés depuis la date de début spécifiée jusqu'à la date courante. Noter qu'un mécanisme de procédure est nécessaire pour empêcher les utilisateurs d'accéder aux messages auxquels ils ne sont pas autorisés à accéder.

3.1.10 Demande et réponse d'interrogation de GLA

Il y a des situations où les GLO et membres de la GL peuvent avoir besoin de déterminer certaines informations du GLA sur la GL. Les GLO et les membres de la GL utilisent glaQueryRequest, défini au paragraphe 3.1.10.1, pour demander des informations et les GLA utilisent glaQueryResponse, défini au paragraphe 3.1.10.2, pour retourner les informations demandées. Le paragraphe 3.1.10.3 inclut un type et une valeur de demande et de réponse ; d'autres peuvent être définis dans des documents supplémentaires.

3.1.10.1 Demande d'interrogation de GLA

Les GLO et les membres de la GL utilisent glaQueryRequest pour s'assurer des informations sur le GLA. L'attribut de commande glaQueryRequest a la syntaxe de GLAQueryRequest :

```
GLAQueryRequest ::= SEQUENCE {
  glaRequestType   IDENTIFIANT D'OBJET,
  glaRequestValue  ANY DÉFINI PAR glaRequestType }
```

3.1.10.2 Réponse d'interrogation de GLA

Les GLA retournent une glaQueryResponse après avoir reçu une GLAQueryRequest. La glaQueryResponse DOIT être signée par un GLA. L'attribut de commande glaQueryResponse a la syntaxe de GLAQueryResponse :

```
GLAQueryResponse ::= SEQUENCE {
  glaResponseType   IDENTIFIANT D'OBJET,
  glaResponseValue  ANY DÉFINI PAR glaResponseType }
```

3.1.10.3 Types de demande et réponse

Les demandes et réponses sont enregistrées comme une paire sous l'arc d'identifiants d'objet suivant :

```
IDENTIFIANT D'OBJET id-cmc-glaRR  ::= { id-cmc 99 }
```

Le présent document définit une paire de demande/réponse pour que les membres de la GL et les GLO interrogent le GLA sur la liste d'algorithmes qu'il prend en charge. L'identifiant d'objet (OID) suivant est inclus dans le champ glaQueryType :

```
IDENTIFIANT D'OBJET id-cmc-gla-skdAlgRequest  ::={ id-cmc-glaRR 1 }

    SKDAlgRequest ::= NULL
```

Si le GLA prend en charge les messages GLAQueryRequest et GLAQueryResponse, le GLA peut retourner l'OID suivant dans le champ glaQueryType :

IDENTIFIANT D'OBJET id-cmc-gla-skdAlgResponse ::= { id-cmc-glaRR 2 }

glaQueryValue a la forme des attributs smimeCapabilities comme défini dans la [RFC3851].

3.1.11 Fourniture de certificat

Les GLA et les GLO utilisent glProvideCert pour demander qu'un membre de la GL fournisse un certificat de chiffrement nouveau ou mis à jour. Le message glProvideCert DOIT être signé du GLA ou du GLO. Si le PKC du membre de la GL a été révoqué, le GLO ou GLA NE DOIT PAS l'utiliser pour générer les EnvelopedData qui encapsulent la demande glProvideCert. L'attribut de commande glProvideCert a la syntaxe de GLManageCert :

```
GLManageCert ::= SEQUENCE {
  glName      GeneralName,
  glMember  GLMember }
```

Les champs dans GLManageCert ont la signification suivante :
- glName indique le nom de la Gl à laquelle le nouveau certificat du membre de la GL va être associé.
- glMember indique les particularités du membre de la GL :
    -- glMemberName indique le nom du membre de la GL.
    -- glMemberAddress indique l'adresse du membre de la GL. Elle PEUT être omise.
    -- certificates DEVRAIT être omis.

3.1.12 Mise à jour de certificat

Les membres de la GL et les GLO utilisent glUpdateCert pour fournir un nouveau certificat pour la GL. Les membres de la GL peuvent générer un glUpdateCert non sollicité ou générer une réponse glUpdateCert par suite de la réception d'un message glProvideCert. Les membres de la GL DOIVENT signer le glUpdateCert. Si le certificat de chiffrement du membre de la GL a été révoqué, le membre de la GL NE DOIT PAS l'utiliser pour générer les EnvelopedData qui encapsulent la demande ou réponse glUpdateCert. L'attribut de commande glUpdateCert a la syntaxe de GLManageCert :

```
GLManageCert ::= SEQUENCE {
  glName      GeneralName,
  glMember  GLMember }
```

Les champs dans GLManageCert ont la signification suivante :
- glName indique le nom de la GL à laquelle le nouveau certificat du membre de la GL devrait être associé.
- glMember indique les particularités du membre de la GL :
    -- glMemberName indique le nom du membre de la GL.
    -- glMemberAddress indique l'adresse du membre de la GL. Elle PEUT être omise.
    -- certificates PEUT être omis si le message GLManageCert est envoyé pour demander le certificat du membre de la GL ; autrement, il DOIT être inclus. Il inclut les trois champs suivants :
        --- certificates.pKC inclut le certificat de chiffrement du membre qui va être utilisé pour chiffrer la KEK partagée pour ce membre.
        --- certificates.aC PEUT être inclus pour porter un ou plusieurs certificats d'attribut associés au certificat d'attribut du membre.
        --- certificates.certPath PEUT aussi être inclus pour porter des certificats qui pourraient aider le receveur à construire des chemins de certification valides pour le certificat fourni dans certificates.pKC et les certificats d'attribut fournis dans certificates.aC. Ces certificats sont facultatifs parce que ils pourraient être déjà inclus ailleurs dans le message (par exemple, dans la couche externe de CMS).

3.1.13 Clé de GL

Le GLA utilise glKey pour distribuer la KEK partagée. Le message glKey DOIT être signé par le GLA. L'attribut de commande glKey a la syntaxe de GLKey :

```
GLKey ::= SEQUENCE {
  glName          GeneralName,
  glIdentifier    KEKIdentifier,                    -- voir la [RFC3852]
  glkWrapped      RecipientInfos,                   -- voir la [RFC3852]
  glkAlgorithm    AlgorithmIdentifier,
  glkNotBefore    GeneralizedTime,
  glkNotAfter     GeneralizedTime }

-- KEKIdentifier est inclus seulement pour illustration car il est importé de la [RFC3852].

KEKIdentifier ::= SEQUENCE {
  keyIdentifier CHAÎNE D'OCTETS,
  date          GeneralizedTime   FACULTATIF,
  other         OtherKeyAttribute FACULTATIF }
```

Les champs dans GLKey ont la signification suivante :

- glName est le nom de la GL.
- glIdentifier est l'identifiant de la clé de la KEK partagée. Voir au paragraphe 6.2.3 de la [RFC3852] la description des sous champs.
- glkWrapped est la KEK partagée enveloppée pour la GL pour une durée particulière. Les RecipientInfos DOIVENT être générées comme spécifié au paragraphe 6.2 de la [RFC3852]. Le choix ktri de RecipientInfo DOIT être pris en charge. La clé dans le champ EncryptedKey (c'est-à-dire, la KEK partagée distribuée) DOIT être générée conformément au paragraphe concernant la génération de nombres aléatoires dans les considérations sur la sécurité de la [RFC3852].
- glkAlgorithm identifie l'algorithme avec lequel la KEK partagée est utilisée. Comme aucun contenu de données chiffré n'est porté à ce moment, les paramètres codés avec l'algorithme devraient être la structure définie pour smimeCapabilities plutôt que du contenu chiffré.
- glkNotBefore indique la date à laquelle la KEK partagée est considérée valide. Les valeurs de GeneralizedTime DOIVENT être exprimées en UTC (Zoulou) et DOIVENT inclure les secondes (c'est-à-dire, les temps sont en forme AAAAMMJJHHMMSSZ) même quand le nombre de secondes est zéro. Les valeurs de GeneralizedTime NE DOIVENT PAS inclure de fractions de secondes.
- glkNotAfter indique la date après laquelle la KEK partagée est considérée comme invalide. Les valeurs de GeneralizedTime DOIVENT être exprimées en UTC (Zoulou) et DOIVENT inclure les secondes (c'est-à-dire, les temps sont en forme AAAAMMJJHHMMSSZ) même lorsque le nombre de secondes est zéro. Les valeurs de GeneralizedTime NE DOIVENT PAS inclure de fractions de secondes.

Si le message glKey est en réponse à un message glUseKEK :

- Le GLA DOIT générer des messages glKey séparés pour chaque receveur si glUseKEK.glKeyAttributes.recipientsNotMutuallyAware est réglé à VRAI. Pour chaque receveur, on veut générer un message qui contient la clé de ce receveur (c'est-à-dire, un message avec un attribut).
- Le GLA DOIT générer le nombre demandé de messages glKey. La valeur dans glUseKEK.glKeyAttributes.generationCounter indique le nombre de messages glKey demandés.

Si le message glKey est en réponse à un message glRekey :

- Le GLA DOIT générer des messages glKey séparés pour chaque receveur si glRekey.glNewKeyAttributes.recipientsNotMutuallyAware est réglé à VRAI.
- Le GLA DOIT générer le nombre demandé de messages glKey. La valeur dans glUseKEK.glKeyAttributes.generationCounter indique le nombre de messages glKey demandés.
- Le GLA DOIT générer un message glKey pour chaque KEK partégée en instance pour la GL quand glRekeyAllGLKeys est régélé à VRAI.

Si le message glKey n'était pas en réponse à glRekey ou glUseKEK (par exemple, lorsque le GLA contrôle les changements de clés) :

- Le GLA DOIT générer des messages glKey séparés pour chaque receveur quand le glUseKEK.glNewKeyAttributes.recipientsNotMutuallyAware qui a établi la GL était réglé à VRAI.
- Le GLA PEUT générer des messages glKey avant l'expiration de la durée de la dernière KEK partagée en instance, lorsque le nombre de messages glKey générés est generationCounter moins un (1). D'autres mécanismes de distribution peuvent aussi être pris en charge pour assurer cette fonction.

3.2 Utilisation de CMC, CMS, et PKIX

Les paragraphes qui suivent traitent de l'utilisation de la CMC, de la CMS, et du profil de certificat PKIX et de CRL.

3.2.1 Couches de protection

Les sous paragraphes qui suivent traitent de la protection requise pour les attributs de commande définis dans ce document.

Note : il y a plusieurs façons d'encapsuler SignedData et EnvelopedData. La première est d'utiliser une enveloppe MIME autour de chaque ContentInfo, comme spécifié dans la [RFC3851]. La seconde est de ne pas utiliser d'enveloppe MIME autour de chaque ContentInfo, comme spécifié dans la [RFC3855] "Transport d'objets S/MIME dans X.400".

3.2.1.1 Protection minimum

Au minimum, une SignedData DOIT protéger chaque demande et réponse encapsulée dans PKIData et PKIResponse. Voici une description de l'enveloppement minimum :

Protection minimum
SignedData
  PKIData ou PKIResponse
  controlSequence

Avant toute action sur toute demande ou réponse, SignedData DOIT être traité conformément à la [RFC3852].

3.2.1.2 Protection supplémentaire

Une EnvelopedData supplémentaire PEUT aussi être utilisée pour assurer la confidentialité de la demande et réponse. Une SignedData supplémentaire PEUT aussi être ajoutée pour assurer l'authentification et la protection de l'intégrité des EnvelopedData encapsulées. Voici une description des enveloppes supplémentaires facultatives :

Protection de la confidentialité	Authentification et intégrité de la protection de la confidentialité
EnvelopedData	SignedData
SignedData	EnvelopedData
PKIData ou PKIResponse	SignedData
controlSequence	controlSequence de PKIData ou PKIResponse

Si un message entrant est chiffré, la confidentialité du message DOIT être préservée. Tous les objets EnvelopedData DOIVENT être traités comme spécifié dans la [RFC3852]. Si une SignedData est ajoutée sur une EnvelopedData, un attribut ContentHints DEVRAIT être ajouté. Voir au paragraphe 2.9 de la [RFC2634] "Services de sécurité étendus pour S/MIME".

Si le GLO ou membre de la GL applique la confidentialité à une demande, les EnvelopedData DOIVENT inclure le GLA comme receveur. Si le GLA transmet la demande du membre de la GL au GLO, alors le GLA DOIT déchiffrer le contenu des EnvelopedData, supprimer la couche de confidentialité, et appliquer sa propre couche de confidentialité comme un EnvelopedData avec le GLO comme receveur.

3.2.2 Combinaison de demandes et réponses

Plusieurs demandes et réponses correspondant à une GL PEUVENT être incluses dans une PKIData.controlSequence ou PKIResponse.controlSequence. Les demandes et réponses pour plusieurs GL PEUVENT être combinées dans une PKIData ou PKIResponse en utilisant PKIData.cmsSequence et PKIResponse.cmsSequence. Une cmsSequence séparée DOIT être utilisée pour des GL différentes. C'est-à-dire que les demandes correspondant à deux GL différentes sont incluses dans des cmsSequences différentes. Le diagramme suivant décrit plusieurs demandes et réponses combinées dans une PKIData et PKIResponse :

Demandes et réponses multiples

Demande	**Réponse**
SignedData	SignedData
PKIData	PKIResponse

cmsSequence	cmsSequence
SignedData	SignedData
PKIData	PKIResponse
controlSequence	controlSequence
Une ou plusieurs demandes correspondant à une GL	Une ou plusieurs réponses correspondant à une GL
SignedData	SignedData
PKIData	PKIResponse
controlSequence	controlSequence
Une ou plusieurs demandes correspondant à une autre GL	Une ou plusieurs réponses correspondant à une autre GL

Quand on applique la confidentialité à plusieurs demandes et réponses, toutes les demandes/réponses PEUVENT être incluses dans une EnvelopedData. En voici la description :

Confidentialité de plusieurs demandes et réponses enveloppées ensemble

```
----------------
EnvelopedData
 SignedData
  PKIData
   cmsSequence
    SignedData
     PKIResponse
      controlSequence
       Une ou plusieurs demandes correspondant à une GL
    SignedData
     PKIData
      controlSequence
       Une ou plusieurs demandes correspondant à une GL
```

Certaines combinaisons de demandes dans une PKIData.controlSequence et une PKIResponse.controlSequence ne sont pas permises. Les combinaisons invalides mentionnées ici NE DOIVENT PAS être générées :

Combinaisons invalides
glUseKEK & glDeleteMember
glUseKEK & glRekey
glUseKEK & glDelete
glDelete & glAddMember
glDelete & glDeleteMember
glDelete & glRekey
glDelete & glAddOwner
glDelete & glRemoveOwner

Pour éviter des erreurs inutiles, certaines demandes et réponses DEVRAIENT être traitées avant d'autres. Voici la priorité du traitement de message, si elle n'est pas mentionnée, c'est une décision de mise en œuvre de dire laquelle traiter en premier : glUseKEK avant glAddMember, glRekey avant glAddMember, et glDeleteMember avant glRekey. Noter que c'est une priorité de traitement, mais elle n'implique pas d'ordre quant au contenu.

3.2.3 Messages générés par le GLA

Quand le GLA génère un message de succès ou d'échec, il en génère un pour chaque demande. Les valeurs de SKDFailInfo de unsupportedDuration, unsupportedDeliveryMethod, unsupportedAlgorithm, noGLONameMatch, nameAlreadyInUse, alreadyAnOwner, et notAnOwner ne sont pas retournées aux membres de la GL.

Si GLKeyAttributes.recipientsNotMutuallyAware est réglé à VRAI, une PKIResponse.cMCStatusInfoExt et PKIData.glKey séparée DOIT être générée pour chaque receveur. Cependant, il est valide d'envoyer un message avec plusieurs attributs au même receveur.

Si le GL a plusieurs GLO, le GLA DOIT envoyer des messages cMCStatusInfoExt au GLO demandeur. Le mécanisme pour déterminer quel GLO a fait la demande sort du domaine d'application de ce document.

Si une GL est géré et si le GLA reçoit un message glAddMember, glDeleteMember, ou glkCompromise, le GLA redirige la

demande sur le GLO pour révision. Une SignedData supplémentaire DOIT être appliquée à la demande redirigée comme suit :

Demandes transmises par le GLA
SignedData
 PKIData
  cmsSequence
   SignedData
    PKIData
     controlSequence

3.2.4 Attributs de commande de CMC et attributs signés par CMS

CMC porte des attributs de commande comme attributs de CMS signés. Ces attributs sont définis dans les [RFC5272] et [RFC3852]. Certains de ces attributs sont EXIGÉS ; d'autres sont FACULTATIFS. Les attributs exigés sont : cMCStatusInfoExt transactionId, senderNonce, recipientNonce, queryPending, et signingTime. D'autres attributs peuvent aussi être utilisés ; cependant, leur utilisation sort du domaine d'application de ce document. Les paragraphes qui suivent spécifient les exigences en plus de celles déjà spécifiées dans les [RFC5272] et [RFC3852].

3.2.4.1 Utilisation de cMCStatusInfoExt

cMCStatusInfoExt est utilisé par les GLA pour indiquer aux GLO et membres de la GL qu'une demande a échoué. Deux classes de codes d'échec sont utilisées dans ce document. Les erreurs de la liste de CMCFailInfo, figurant au paragraphe 6.1.4 de la RFC 5272, sont codées comme défini dans la CMC. Les codes d'erreur définis dans le présent document sont codés en utilisant le champ ExtendedFailInfo de la structure cmcStatusInfoExt. Si le même code d'échec s'applique à plusieurs commandes, une seule structure cmcStatusInfoExt peut être utilisée avec plusieurs éléments dans cMCStatusInfoExt.bodyList. Le GLA PEUT aussi retourner d'autres informations pertinentes dans statusString. L'identifiant d'objet SKDFailInfo et sa valeur sont :

IDENTIFIANT D'OBJET id-cet-skdFailInfo ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) cet(15) skdFailInfo(1) }

```
SKDFailInfo ::= ENTIER {
  unspecified             (0),      (non spécifié)
  closedGL                (1),      (GL close)
  unsupportedDuration     (2),      (durée non prise en charge)
  noGLACertificate        (3),      (pas de certificat de GLA)
  invalidCert             (4),      (certificat invalide)
  unsupportedAlgorithm    (5),      (algorithme non pris en charge)
  noGLONameMatch          (6),      (pas de correspondance du nom de GLO)
  invalidGLName           (7),      (nom de GL invalide)
  nameAlreadyInUse        (8),      (nom déjà utilisé)
  noSpam                  (9),      (pas de pourriel)
-- obsolète               (10),
  alreadyAMember          (11),     (déjà membre)
  notAMember              (12),     (non membre)
  alreadyAnOwner          (13),     (déjà un possesseur)
  notAnOwner              (14) }    (n'est pas un possesseur)
```

Les valeurs ont la signification suivante :
- unspecified indique que le GLA est incapable ou ne veut pas effectuer l'action demandée et ne veut pas en indiquer la raison.
- closedGL indique que des membres peuvent seulement être ajoutés ou supprimés par le GLO.
- unsupportedDuration indique que le GLA ne prend pas en charge la génération de clés valides pour la durée demandée.
- noGLACertificate indique que le GLA n'a pas de certificat valide.
- invalidCert indique que le certificat d'attribut du membre n'était pas vérifiable (c'est-à-dire, la signature n'est pas valide, le numéro de série du certificat est présent sur une CRL, le certificat a expiré, etc.).
- unsupportedAlgorithm indique que le GLA ne prend pas en charge l'algorithme demandé.
- noGLONameMatch indique qu'un des noms dans le certificat utilisé pour signer une demande ne correspond pas au nom

d'un GLO enregistré.
- invalidGLName indique que le GLA ne prend pas en charge le glName présent dans la demande.
- nameAlreadyInUse indique que le glName est déjà alloué sur le GLA.
- noSpam indique que le membre prospectif de la GL n'a pas signé la demande (c'est-à-dire, si le nom dans glMember.glMemberName ne correspond à un des noms (soit le nom distinctif de sujet, soit un des noms de remplacement du sujet) dans le certificat utilisé pour signer la demande).
- alreadyAMember indique que le membre prospectif de la GL est déjà membre de la GL.
- notAMember indique que le membre prospectif de la GL à supprimer n'est pas présentement membre de la GL.
- alreadyAnOwner indique que le GLO prospectif est déjà un GLO.
- notAnOwner indique que le GLO prospectif à supprimer n'est pas présentement un GLO.

cMCStatusInfoExt est utilisé par les GLA pour indiquer aux GLO et membres de la GL qu'une demande s'est achevée avec succès. Si la demande a réussi, le GLA retourne une réponse cMCStatusInfoExt avec cMCStatus.success et facultativement d'autres informations pertinentes dans statusString.

Quand la GL est gérée et quand le GLO a revu les demandes glAddMember, glDeleteMember, et glkComrpomise initiées par le membre de la GL, le GLO utilise cMCStatusInfoExt pour indiquer le succès ou l'échec de la demande. Si la demande est permise, cMCStatus.success est retourné et statusString est facultativement retourné pour porter des informations supplémentaires. Si la demande est refusée, cMCStatus.failed est retourné et statusString est facultativement retourné pour porter des informations supplémentaires. De plus, les SKDFailInfo appropriées peuvent être incluses dans cMCStatusInfoExt.extendedFailInfo.

cMCStatusInfoExt est utilisé par les GLO, les GLA, et les membres de la GL pour indiquer que la vérification de signature a échoué. Si la signature a échoué à se vérifier sur tout attribut de commande sauf cMCStatusInfoExt, un attribut de commande cMCStatusInfoExt DOIT être retourné en indiquant cMCStatus.failed et otherInfo.failInfo.badMessageCheck. Si la signature sur les PKIData les plus externes a échoué, la valeur de bodyList est zéro (0). Si la signature sur tout autre PKIData a échoué, la valeur de bodyList est celle du bodyPartId provenant de la demande ou réponse. Les GLO et membres de la GL qui reçoivent des messages cMCStatusInfoExt dont la signature est invalide DEVRAIENT générer une nouvelle demande pour éviter des boucles de message badMessageCheck.

cMCStatusInfoExt est aussi utilisé par les GLO et GLA pour indiquer qu'une demande ne peut pas être effectuée immédiatement. Si la demande ne pourrait pas être traitée immédiatement par le GLA ou GLO, l'attribut de commande cMCStatusInfoExt DOIT être retourné en indiquant cMCStatus.pending et otherInfo.pendInfo. Quand les demandes sont redirigées sur le GLO pour approbation (pour les listes gérées) le GLA NE DOIT PAS retourner un cMCStatusInfoExt indiquant une interrogation en instance.

cMCStatusInfoExt est aussi utilisé par les GLA pour indiquer qu'une glaQueryRequest n'est pas prise en charge. Si la glaQueryRequest n'est pas prise en charge, l'attribut de commande cMCStatusInfoExt DOIT être retourné en indiquant cMCStatus.noSupport et statusString est facultativement retourné pour porter des informations supplémentaires.

cMCStatusInfoExt est aussi utilisé par les membres de la GL, les GLO, et les GLA pour indiquer que l'heure de signature (signingTime, voir au paragraphe 3.2.4.3) n'est pas assez proche de l'heure spécifiée en local. Si l'heure locale n'est pas assez proche de l'heure spécifiée dans signingTime, une cMCStatus.failed et une otherInfo.failInfo.badTime PEUVENT être retournées.

3.2.4.2 Utilisation de transactionId

Un transactionId PEUT être inclus par les GLO, GLA, ou membres de la GL pour identifier une certaine transaction. Toutes les demandes et réponses suivantes relatives à la demande originale DOIVENT inclure le même attribut de commande transactionId. Si les membres de la GL incluent un transactionId et si la demande est redirigée sur le GLO, le GLA PEUT inclure un transactionId supplémentaire dans les PKIData externes. Si le GLA a inclus un transactionId supplémentaire dans les PKIData externes, quand le GLO génère une réponse cMCStatusInfoExt, il en génère une pour le GLA avec le transactionId du GLA et une pour le membre de la GL avec le transactionId du membre de la GL.

3.2.4.3 Utilisation de noms occasionnels et de signingTime

Des noms occasionnels (voir au paragraphe 5.6 de la [RFC5272]) et une indication de quand le message a été signé (voir au paragraphe 11.3 de la [RFC3852]) peuvent être utilisés pour assurer la protection contre la répétition au niveau de l'application.

Pour protéger la GL, tous les messages DOIVENT inclure l'attribut signingTime. Les générateurs et les receveurs de message peuvent alors utiliser l'heure fournie dans cet attribut pour déterminer si ils ont précédemment reçu le message.

Si le générateur du message inclut un senderNonce, la réponse au message DOIT inclure la valeur du senderNonce reçue comme le recipientNonce et une nouvelle valeur comme valeur de senderNonce dans la réponse.

Si un GLA agrège plusieurs messages ou transmet un message à un GLO, le GLA PEUT facultativement générer une nouvelle valeur de nom occasionnel et l'inclure dans le message enveloppant. Quand la réponse revient du GLO, le GLA construit une réponse au générateur du message et traite chacune des valeurs de nom occasionnel provenant des messages générés.

Pour ces attributs, il est nécessaire de conserver les échanges d'informations d'état pour comparer les résultats. La durée pendant laquelle ces informations sont conservées est une affaire de politique locale.

3.2.4.4 Exigences de prise en charge d'attribut de CMC et CMS

Les exigences de mise en œuvre suivantes pour les attributs de commande de CMC et les attributs signés de CMS doivent être respectés pour qu'une mise en œuvre soit considérée conforme à la présente spécification :

Exigence de mise en œuvre							
GLO		GLA			Membre GL		
O	R	O	R	F	O	R	Attribut
DOIT	DOIT	DOIT	DOIT	-	DOIT	DOIT	cMCStatusInfoExt
PEUT	PEUT	DOIT	DOIT	-	PEUT	PEUT	transactionId
PEUT	PEUT	DOIT	DOIT	-	PEUT	PEUT	senderNonce
PEUT	PEUT	DOIT	DOIT	-	PEUT	PEUT	recepientNonce
DOIT	DOIT	DOIT	DOIT	-	DOIT	DOIT	SKDFailInfo
DOIT	DOIT	DOIT	DOIT	-	DOIT	DOIT	signingTime

3.2.5 Messages de membre de GL resoumis

Quand la GL est gérée, le GLA transmet les demandes du membre de la GL au GLO pour qu'il les approuve en créant un nouveau message de demande contenant la ou les demandes du membre de la GL comme un élément de cmsSequence. Si le GLO approuve la demande, il peut ajouter une nouvelle couche d'enveloppe et la renvoyer au GLA ou créer un nouveau message et l'envoyer au GLA. (Noter que dans ce cas, il y a maintenant trois couches de messages PKIData avec les couches appropriées de signature.)

3.2.6 Certificat PKIX et profil de CRL

Les signatures, les certificats, et les CRL sont vérifiés conformément au profil PKIX [RFC5280].

La correspondance de noms est effectuée conformément au profil PKIX [RFC5280].

Tous les noms distinctifs doivent suivre la convention UTF8String notée dans le profil PKIX [RFC5280].

Un certificat par GL va être produit au GLA.

La politique de la GL peut obliger que l'adresse du membre de la GL soit incluse dans le certificat du membre de la GL.

4. Messages administratifs

Un certain nombre de messages administratifs doivent être échangés pour gérer une GL. Les paragraphes qui suivent décrivent en détail chaque combinaison de message de demande et réponse. Les procédures définies dans cette section ne sont pas de mise en œuvre obligatoire.

4.1 Allouer une KEK au GL

Avant de générer une clé de groupe, une GL doit être établie et une KEK partagée allouée à la GL. La Figure 3 décrit les interactions de protocole pour établir et allouer une KEK partagée. Noter que les messages d'erreur ne sont pas décrits dans la Figure 3. De plus, le comportement pour les attributs facultatifs de CMC transactionId, senderNonce, et recipientNonce n'est pas traité dans ces procédures.

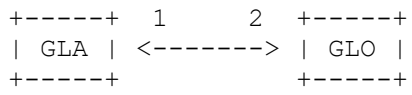


Figure 3 : Création de liste de groupe

Le processus est le suivant :

1. Le GLO est l'entité chargée de demander la création de la GL. Le GLO envoie une demande SignedData.PKIData.controlSequence.glUseKEK au GLA (1 dans la Figure 3). Le GLO DOIT inclure glName, glAddress, glOwnerName, glOwnerAddress, et glAdministration. Le GLO PEUT aussi inclure ses préférences pour la KEK partagée dans glKeyAttributes en indiquant si il contrôle le changement de clés dans rekeyControlledByGLO, si des messages glKey séparés devraient être envoyés à chaque receveur dans recipientsNotMutuallyAware, l'algorithme dont l'utilisation est demandée avec la KEK partagée dans requestedAlgorithm, la durée de la KEK partagée, et combien de KEK partagées devraient être initialement distribuées dans generationCounter. Le GLO DOIT aussi inclure l'attribut signingTime avec cette demande.

1.a - Si le GLO connaît les membres à ajouter à la GL, la ou les demandes glAddMember PEUVENT être incluses dans la même controlSequence que la demande glUseKEK (voir au paragraphe 3.2.2). Le GLO indique le même glName dans la demande glAddMember que dans glUseKEK.glInfo.glName. Les autres procédures de glAddMember sont traitées au paragraphe 4.3.

1.b - Le GLO peut appliquer la confidentialité à la demande en encapsulant les SignedData.PKIData dans un EnvelopedData (voir au paragraphe 3.2.1.2).

1.c - Le GLO peut aussi facultativement appliquer un autre SignedData sur les EnvelopedData (voir au paragraphe 3.2.1.2).

2 - À réception de la demande, le GLA vérifie le signingTime et vérifie la signature sur les SignedData.PKIData les plus internes. Si des SignedData et/ou EnvelopedData supplémentaires encapsulent la demande (voir les paragraphes 3.2.1.2 et 3.2.2) le GLA vérifie la ou les signatures externes et/ou déchiffre la ou les couches externes avant de vérifier la signature sur les SignedData les plus internes.

2.a - Si la valeur d'attribut signingTime n'est pas dans la fenêtre de temps acceptée localement, le GLA PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

2.b - Autrement si le traitement de la signature se continue et si les signatures ne sont pas valides, le GLA retourne une réponse cMCStatusInfoExt indiquant cMCStatus.failed et otherInfo.failInfo.badMessageCheck. De plus, un attribut signingTime est inclus dans la réponse.

2.c - Autrement si les signatures sont vérifiées mais si le GLA n'a pas de certificat valide, le GLA retourne un cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de noValidGLACertificate. De plus, un attribut signingTime est inclus dans la réponse. Au lieu de retourner immédiatement le code d'erreur, le GLA tente d'obtenir un certificat, éventuellement en utilisant la [RFC5272].

2.d - Autrement les signatures sont valides et le GLA a bien un certificat valide, le GLA vérifie qu'un des noms dans le certificat utilisé pour signer la demande correspond à un des noms dans glUseKEK.glOwnerInfo.glOwnerName.

2.d.1 - Si les noms ne correspondent pas, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de noGLONameMatch. De plus, un attribut signingTime est inclus dans la réponse.

2.d.2 - Autrement si les noms correspondent tous, le GLA vérifie que le glName et la glAddress ne sont pas déjà utilisés. Le GLA vérifie aussi tous les glAddMember inclus dans la controlSequence avec sa glUseKEK. La suite du traitement de glAddMember est traitée au paragraphe 4.3.

2.d.2.a - Si le glName est déjà utilisé, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et la valeur de otherInfo.extendedFailInfo.SKDFailInfo de nameAlreadyInUse. De plus, un attribut signingTime est inclus dans la réponse.

2.d.2.b – Autrement, si le requestedAlgorithm n'est pas pris en charge, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de unsupportedAlgorithm. De plus, un attribut signingTime est inclus dans la réponse.

2.d.2.c - Autrement si la durée ne peut pas être prise en charge, le déterminer sort du domaine d'application de ce document, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et la valeur de otherInfo.extendedFailInfo.SKDFailInfo de unsupportedDuration. De plus, un attribut signingTime est inclus dans la réponse.

2.d.2.d - Autrement si la GL ne peut pas être prise en charge pour d'autres raisons, que le GLA ne souhaite pas divulguer, il retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et la valeur de otherInfo.extendedFailInfo.SKDFailInfo de "unspecified". De plus, un attribut signingTime est inclus dans la réponse.

2.d.2.e - Autrement si le glName n'est pas déjà utilisé, la durée peut être prise en charge, et si l'algorithme demandé est pris en charge, le GLA DOIT retourner un cMCStatusInfoExt indiquant cMCStatus.success et un attribut signingTime (2 dans la Figure 3). Le GLA prend aussi des actions administratives, qui sortent du domaine d'application de ce document, pour mémoriser le glName, glAddress, glKeyAttributes, glOwnerName, et glOwnerAddress. Le GLA envoie aussi un message glKey comme décrit à la Section 5.

2.d.2.e.1 - Le GLA peut appliquer la confidentialité à la réponse en encapsulant la SignedData.PKIResponse dans un EnvelopedData si la demande était encapsulée dans un EnvelopedData (voir au paragraphe 3.2.1.2).

2.d.2.e.2 - Le GLA peut aussi facultativement appliquer un autre SignedData sur le EnvelopedData (voir au paragraphe 3.2.1.2).

3 - À réception des réponses cMCStatusInfoExt, le GLO vérifie le signingTime et la ou les signatures du GLA. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la réponse (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLO vérifie la signature externe et/ou déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

3.a - Si la valeur d'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLO PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

3.b - Autrement si le traitement de la signature se continue et si les signatures sont vérifiées, le GLO DOIT s'assurer qu'un des noms dans le certificat utilisé pour signer la réponse correspond au nom de la GL.

3.b.1 - Si le nom de la GL ne correspond pas au nom présent dans le certificat utilisé pour signer le message, le GLO ne devrait pas croire la réponse.

3.b.2 - Autrement si le nom de la GL correspond bien au nom présent dans le certificat et,
3.b.2.a - si les signatures se vérifient et si la réponse était cMCStatusInfoExt indiquant cMCStatus.success, le GLO a réussi à créer la GL ;
3.b.2.b - autrement si les signatures sont valides et si la réponse est cMCStatusInfoExt.cMCStatus.failed pour n'importe quelle raison, le GLO peut tenter à nouveau de créer la GL en utilisant les informations fournies dans la réponse. Le GLO peut aussi utiliser la glaQueryRequest pour déterminer les algorithmes et autres caractéristiques prises en charge par le GLA (voir au paragraphe 4.9).

4.2 Supprimer le GL du GLA

De temps en temps, il y a des instances où une GL n'est plus nécessaire. Dans ce cas, le GLO supprime la GL. La Figure 4 décrit les interactions de protocole pour supprimer une GL. Noter que le comportement pour les attributs de commande de CMC transactionId, senderNonce, et recipientNonce n'est pas traité dans ces procédures.

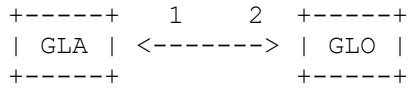


Figure 4 : suppression de liste de groupe

Le processus est le suivant :

1 - Le GLO est chargé de demander la suppression de la GL. Le GLO envoie une demande SignedData.PKIData.controlSequence.glDelete au GLA (1 dans la Figure 4). Le nom de la GL à supprimer est inclus dans GeneralName. Le GLO DOIT aussi inclure l'attribut signingTime et peut aussi inclure des attributs transactionId et senderNonce.

1.a - Le GLO peut facultativement appliquer la confidentialité à la demande en encapsulant le SignedData.PKIData dans un EnvelopedData (voir au paragraphe 3.2.1.2).

1.b - Le GLO PEUT facultativement appliquer un autre SignedData sur le EnvelopedData (voir au paragraphe 3.2.1.2).

2 - À réception de la demande, le GLA vérifie le signingTime et vérifie la signature sur les SignedData.PKIData les plus internes. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la demande (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLA vérifie la signature externe et/ou déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

2.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLA PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

2.b - Autrement si le traitement de la signature se continue et si les signatures ne peuvent pas être vérifiées, le GLA retourne une réponse cMCStatusInfoExt indiquant cMCStatus.failed et otherInfo.failInfo.badMessageCheck. De plus, un attribut signingTime est inclus dans la réponse.

2.c - Autrement si les signatures se vérifient, le GLA s'assure que la GL est prise en charge en vérifiant que le nom de la GL correspond à un glName mémorisé sur le GLA.

2.c.1 - Si le glName n'est pas pris en charge par le GLA, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de invalidGLName. De plus, un attribut signingTime est inclus dans la réponse.

2.c.2 - Autrement si le glName est pris en charge par le GLA, celui-ci s'assure qu'un GLO enregistré a signé la demande glDelete en vérifiant si un des noms présents dans le certificat de signature numérique utilisé pour signer la demande glDelete correspond à un GLO enregistré.

2.c.2.a - Si les noms ne correspondent pas, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de noGLONameMatch. De plus, un attribut signingTime est inclus dans la réponse.

2.c.2.b - Autrement si les noms correspondent bien, mais si la GL ne peut pas être supprimée pour d'autres raisons que le GLA ne souhaite pas divulguer, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et la valeur de otherInfo.extendedFailInfo.SKDFailInfo de "unspecified". De plus, un attribut signingTime est inclus dans la réponse. Des actions qui sortent du domaine d'application de ce document doivent alors être effectuées pour supprimer la GL du GLA.

2.c.2.c - Autrement si les noms correspondent, le GLA retourne un cMCStatusInfoExt indiquant cMCStatus.success et un attribut signingTime (2 dans la Figure 4). Le GLA ne devrait pas accepter d'autres demandes d'ajout de membre, suppression de membre, ou changement de clé de groupe pour cette GL.

2.c.2.c.1 - Le GLA peut appliquer la confidentialité à la réponse en encapsulant la SignedData.PKIResponse dans un EnvelopedData si la demande était encapsulée dans un EnvelopedData (voir au paragraphe 3.2.1.2).

2.c.2.c.2 - Le GLA PEUT facultativement appliquer un autre SignedData sur le EnvelopedData (voir au paragraphe 3.2.1.2).

3 - À réception de la réponse cMCStatusInfoExt, le GLO vérifie l'heure de signature et la ou les signatures du GLA. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la réponse (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLO vérifie la signature externe et/ou déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

3.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLO PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

3.b - Autrement si le traitement de signature se continue et si les signatures se vérifient, le GLO vérifie qu'un des noms dans le certificat utilisé pour signer la réponse correspond au nom de la GL.

3.b.1 - Si le nom de la GL ne correspond pas au nom présent dans le certificat utilisé pour signer le message, le GLO ne devrait pas croire la réponse.

3.b.2 - Autrement si le nom de la GL correspond bien au nom présent dans le certificat et,
3.b.2.a - si les signatures se vérifient et si la réponse était cMCStatusInfoExt indiquant cMCStatus.success, le GLO a réussi à supprimer la GL ;
3.b.2.b – autrement, si les signatures se vérifient et si la réponse était cMCStatusInfoExt.cMCStatus.failed avec n'importe quelle raison, le GLO peut tenter à nouveau de supprimer la GL en utilisant les informations fournies dans la réponse.

4.3 Ajouter des membres au GL

Pour ajouter des membres aux GL, le GLO ou les membres prospectifs utilisent la demande glAddMember. Le GLA traite cependant différemment les demandes de GLO et de membre prospectif de la GL. Les GLO peuvent soumettre la demande à tout moment pour ajouter des membres à la GL, et le GLA, une fois qu'il a vérifié que la demande vient d'un GLO enregistré, devrait la traiter. Si un membre prospectif envoie la demande, le GLA a besoin de déterminer comment la GL est administrée. Quand le GLO a initialement configuré la GL, il a réglé la GL à être non gérée, gérée, ou close (voir au paragraphe 3.1.1). Dans le cas non gérée, le GLA traite simplement la demande du membre. Dans le cas de GL gérée, le GLA transmet les demandes des membres prospectifs au GLO pour revue. Lorsque il y a plusieurs GLO pour une GL, il sort du domaine d'application de ce document de déterminer à quel GLO la demande est transmise. Le GLO revoit la demande et la rejette ou soumet une demande réformée au GLA. Dans le cas de GLA close, le GLA ne va pas accepter de demande de membres prospectifs. Les paragraphes qui suivent décrivent le traitement pour les GLO, GLA, et membres prospectifs de la GL selon que la demande glAddMember a pour origine un GLO ou un membre prospectif. La Figure 5 décrit les interactions de protocole pour les trois options. Noter que les messages d'erreur ne sont pas décrits. Noter de plus que le comportement pour les attributs de commande facultatifs de CMC transactionId, senderNonce, et recipientNonce n'est pas traité dans ces procédures.

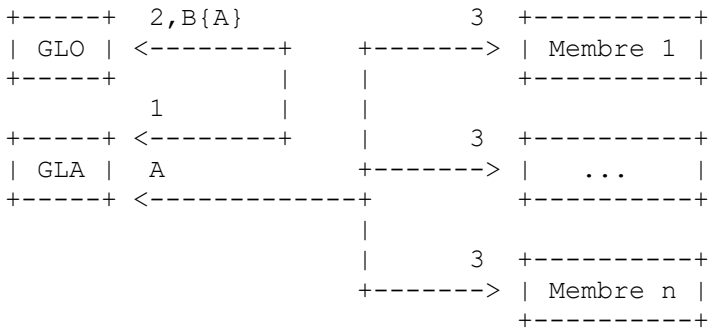


Figure 5 : Ajout de membre

Une importante décision qui doit être prise groupe par groupe est de changer les clés de groupe chaque fois qu'un nouveau membre est ajouté. Normalement, les GL non gérées ne devraient pas voir de changement de clé quand un nouveau membre est ajouté, car les frais généraux associés au changement de clés du groupe deviennent prohibitifs lorsque le groupe est grand. Cependant, les clés des GL gérées et closes peuvent être changées pour maintenir la confidentialité du trafic envoyé par les membres du groupe. Une option de changer les clés des GL gérées ou closes quand un membre est ajouté est de générer une nouvelle GL avec une différente clé de groupe. Le changement de clé de groupe est discuté dans le paragraphe 4.5 et la Section 5.

4.3.1 Ajouts initiés par le GLO

Le processus pour les demandes glAddMember initiées par le GLO est le suivant :

1 - Le GLO collecte les informations pertinentes pour le ou les membres à ajouter (cela peut être fait par des moyens hors bande). Le GLO envoie alors une SignedData.PKIData.controlSequence avec une demande glAddMember séparée pour chaque membre au GLA (1 dans la la Figure 5). Le GLO inclut le nom de la GL dans glName, le nom du membre dans glMember.glMemberName, l'adresse du membre dans glMember.glMemberAddress, et le certificat d'attribut du membre dans glMember.certificates.pKC. Le GLO peut aussi inclure tous les certificats d'attribut associés au certificat d'attribut du membre dans glMember.certificates.aC, et le chemin de certification associé aux certificats de chiffrement et d'attribut du membre dans glMember.certificates.certPath. Le GLO DOIT aussi inclure l'attribut signingTime dans cette demande.

1.a - Le GLO peut facultativement appliquer la confidentialité à la demande en encapsulant le SignedData.PKIData dans un EnvelopedData (voir au paragraphe 3.2.1.2).

1.b - Le GLO peut aussi facultativement appliquer un autre SignedData sur les EnvelopedData (voir au paragraphe 3.2.1.2).

2 - À réception de la demande, le GLA vérifie l'heure de signature et vérifie la signature sur les SignedData.PKIData les plus internes. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la demande (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLA vérifie la signature externe et/ou déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

2.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLA PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

2.b - Autrement si le traitement de la signature se continue et si les signatures ne peuvent pas être vérifiées, le GLA retourne une réponse cMCStatusInfoExt indiquant cMCStatus.failed et otherInfo.failInfo.badMessageCheck. De plus, un attribut signingTime est inclus dans la réponse.

2.c - Autrement si les signatures se vérifient, la demande glAddMember est incluse dans une controlSequence avec la demande glUseKEK, et le traitement du paragraphe 4.1 point 2.d est achevé avec succès, le GLA retourne un cMCStatusInfoExt indiquant cMCStatus.success et un attribut signingTime (2 dans la Figure 5).

2.c.1 - Le GLA peut appliquer la confidentialité à la réponse en encapsulant le SignedData.PKIData dans un EnvelopedData si la demande était encapsulée dans un EnvelopedData (voir au paragraphe 3.2.1.2).

2.c.2 - Le GLA peut aussi facultativement appliquer un autre SignedData sur les EnvelopedData (voir au paragraphe 3.2.1.2).

2.d - Autrement si les signatures se vérifient et si la demande GLAddMember n'est pas incluse dans une controlSequence avec la demande GLCreate, le GLA s'assure que la GL est prise en charge en vérifiant que le glName correspond à un glName mémorisé par le GLA.

2.d.1 - Si le glName n'est pas pris en charge par le GLA, celui-ci retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de invalidGLName. De plus, un attribut signingTime est inclus dans la réponse.

2.d.2 - Autrement si le glName est pris en charge par le GLA, le GLA vérifie si le glMemberName est présent sur la GL.

2.d.2.a - Si le glMemberName est présent sur la GL, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de alreadyAMember. De plus, un attribut signingTime est inclus dans la réponse.

2.d.2.b - Autrement si le glMemberName n'est pas présent sur la GL, le GLA vérifie comment la GL est administrée.

2.d.2.b.1 - Si la GL est close, le GLA vérifie qu'un GLO enregistré a signé la demande en regardant si un des noms dans le certificat de signature numérique utilisé pour signer la demande correspond à un GLO enregistré.

2.d.2.b.1.a - Si les noms ne correspondent pas, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de noGLONameMatch. De plus, un attribut

signingTime est inclus dans la réponse.

2.d.2.b.1.b – Autrement, si les noms correspondent, le GLA vérifie le certificat d'attribut du membre.

2.d.2.b.1.b.1 - Si le certificat d'attribut du membre ne peut pas être vérifié, le GLA peut retourner une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de invalidCert au GLO. De plus, un attribut signingTime est inclus dans la réponse. Si le GLA ne retourne pas une réponse cMCStatusInfoExt.cMCStatus.failed, le GLA produit une demande glProvideCert (voir au paragraphe 4.10).

2.d.2.b.1.b.2 - Autrement si le certificat du membre se vérifie, le GLA retourne un cMCStatusInfoExt indiquant cMCStatus.success et un attribut signingTime (2 dans la Figure 5). Le GLA prend aussi des actions administratives, qui sortent du domaine d'application de ce document, pour ajouter le membre à la GL mémorisé au GLA. Le GLA distribue aussi la KEK partagée au membre via le mécanisme décrit dans la Section 5.

2.d.2.b.1.b.2.a - Le GLA applique la confidentialité à la réponse en encapsulant les SignedData.PKIData dans un EnvelopedData si la demande était encapsulée dans un EnvelopedData (voir au paragraphe 3.2.1.2).

2.d.2.b.1.b.2.b - Le GLA peut aussi facultativement appliquer un autre SignedData sur le EnvelopedData (voir au paragraphe 3.2.1.2).

2.d.2.b.2 - Autrement si la GL est gérée, le GLA vérifie qu'un GLO enregistré, ou le membre prospectif, a signé la demande. Pour les GLO, un des noms dans le certificat utilisé pour signer la demande doit correspondre à un GLO enregistré. Pour un membre prospectif, le nom dans glMember.glMemberName doit correspondre à un des noms dans le certificat utilisé pour signer la demande.

2.d.2.b.2.a - Si le signataire n'est ni un GLO enregistré ni le membre prospectif de la GL, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de noSpam. De plus, un attribut signingTime est inclus dans la réponse.

2.d.2.b.2.b - Autrement si le signataire est un GLO enregistré, le GLA vérifie le certificat d'attribut du membre.

2.d.2.b.2.b.1 - Si le certificat du membre ne peut pas être vérifié, le GLA peut retourner une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de invalidCert. De plus, un attribut signingTime est inclus dans la réponse. Si le GLA ne retourne pas une réponse cMCStatus.failed, le GLA DOIT produire une demande glProvideCert (voir au paragraphe 4.10).

2.d.2.b.2.b.2 - Autrement si le certificat du membre se vérifie, le GLA DOIT retourner un cMCStatusInfoExt indiquant cMCStatus.success et un attribut signingTime au GLO (2 dans la Figure 5). Le GLA prend aussi des actions administratives, qui sortent du domaine d'appication de ce document, pour ajouter le membre à la GL mémorisée dans le GLA. Le GLA distribue aussi la KEK partagée au membre via le mécanisme décrit à la Section 5. La politique de la GL peut rendre obligatoire que l'adresse du membre de la GL soit incluse dans le certificat du membre de la GL.

2.d.2.b.2.b.2.a - Le GLA applique la confidentialité à la réponse en encapsulant les SignedData.PKIData dans un EnvelopedData si la demande était encapsulée dans un EnvelopedData (voir au paragraphe 3.2.1.2).

2.d.2.b.2.b.2.b - Le GLA peut aussi facultativement appliquer un autre SignedData sur les EnvelopedData (voir au paragraphe 3.2.1.2).

2.d.2.b.2.c - Autrement si le signataire est le membre prospectif, le GLA transmet la demande glAddMember (voir au paragraphe 3.2.3) à un GLO enregistré (B{A} dans la Figure 5). Si il y a plus d'un GLO enregistré, le choix du GLO auquel la demande est transmise sort du domaine d'application de ce document. La suite du traitement de la demande transmise par les GLO est traitée dans le point 3 du paragraphe 4.3.2.

2.d.2.b.2.c.1 - Le GLA applique la confidentialité à la demande transmise en encapsulant les SignedData.PKIData dans un EnvelopedData si la demande d'origine était encapsulée dans un EnvelopedData (voir au paragraphe 3.2.1.2).

2.d.2.b.2.c.2 - Le GLA peut aussi facultativement appliquer un autre SignedData sur le EnvelopedData (voir au paragraphe 3.2.1.2).

2.d.2.b.3 - Autrement si la GL est non gérée, le GLA vérifie que un GLO enregistré ou le membre prospectif a signé la demande. Pour les GLO, un des noms dans le certificat utilisé pour signer la demande doit correspondre au nom d'un

GLO enregistré. Pour le membre prospectif, le nom dans glMember.glMemberName doit correspondre à un des noms dans le certificat utilisé pour signer la demande.

2.d.2.b.3.a - Si le signataire n'est ni un GLO enregistré ni le membre prospectif, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de noSpam. De plus, un attribut signingTime est inclus dans la réponse.

2.d.2.b.3.b - Autrement si le signataire est soit un GLO enregistré, soit le membre prospectif, le GLA vérifie le certificat d'attribut du membre.

2.d.2.b.3.b.1 - Si le certificat du membre ne peut pas être vérifié, le GLA peut retourner une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de invalidCert et un attribut signingTime, soit au GLO, soit au membre prospectif, selon l'origine de la demande. Si le GLA ne retourne pas une réponse cMCStatus.failed, le GLA produit une demande glProvideCert (voir au paragraphe 4.10) au GLO ou au membre prospectif selon l'origine de la demande.

2.d.2.b.3.b.2 - Autrement si le certificat du membre se vérifie, le GLA retourne un cMCStatusInfoExt indiquant cMCStatus.success et un attribut signingTime au GLO (2 dans la Figure 5) si le GLO a signé la demande, et au membre de la GL (3 dans la Figure 5) si le membre de la GL a signé la demande. Le GLA prend aussi des actions administratives, qui sortent du domaine d'application de ce document, pour ajouter le membre à la GL mémorisée sur le GLA. Le GLA distribue aussi la KEK partagée au membre via le mécanisme décrit à la Section 5.

2.d.2.b.3.b.2.a - Le GLA applique la confidentialité à la réponse en encapsulant les SignedData.PKIData dans un EnvelopedData si la demande était encapsulée dans un EnvelopedData (voir au paragraphe 3.2.1.2).

2.d.2.b.3.b.2.b - Le GLA peut aussi facultativement appliquer un autre SignedData sur les EnvelopedData (voir au paragraphe 3.2.1.2).

3 - À réception de la réponse cMCStatusInfoExt, le GLO vérifie l'heure de signature et vérifie la ou les signatures du GLA. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la réponse (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLO vérifie la signature externe et/ou déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

3.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLO PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

3.b - Autrement si le traitement de la signature se continue et si les signatures se vérifient, le GLO vérifie qu'un des noms dans le certificat utilisé pour signer la réponse correspond au nom de la GL.

3.b.1 - Si le nom de la GL ne correspond pas au nom présent dans le certificat utilisé pour signer le message, le GLO ne devrait pas croire la réponse.

3.b.2 - Autrement si le nom de la GL correspond au nom présent dans le certificat et,
3.b.2.a - Si les signatures se vérifient et si la réponse est cMCStatusInfoExt indiquant cMCStatus.success, le GLA ajoute le membre à la GL. Si le membre ajouté à une liste gérée et si la demande originale a été signée par le membre, le GLO envoie une cMCStatusInfoExt.cMCStatus.success et un attribut signingTime au membre de la GL.
3.b.2.b - Autrement si le GLO a reçu un cMCStatusInfoExt.cMCStatus.failed avec une raison quelconque, le GLO peut réessayer d'ajouter le membre à la GL en utilisant les informations fournies dans la réponse.

4 - À réception de la réponse cMCStatusInfoExt, le membre prospectif vérifie l'heure de signature et vérifie les signatures du GLA ou GLO. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la réponse (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLO vérifie la signature externe et/ou déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

4.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le membre prospectif PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

4.b - Autrement si le traitement de signature se continue et si les signatures se vérifient, le membre de la GL vérifie qu'un des noms dans le certificat utilisé pour signer la réponse correspond au nom de la GL.

4.b.1 - Si le nom de la GL ne correspond pas au nom présent dans le certificat utilisé pour signer le message, le membre de

la GL ne devrait pas croire la réponse.

4.b.2 - Autrement si le nom de la GL correspond au nom présent dans le certificat et,
4.b.2.a - si les signatures se vérifient, le membre prospectif va être ajouté à la GL ;
4.b.2.b - autrement si le membre prospectif a reçu un cMCStatusInfoExt.cMCStatus.failed, pour n'importe quelle raison, le membre prospectif PEUT tenter à nouveau de s'ajouter lui-même à la GL en utilisant les informations fournies dans la réponse.

4.3.2. Ajouts initiés par le membre prospectif

Le traitement pour les demandes glAddMember initiées par le membre prospectif est le suivant :

1 - Le membre prospectif de la GL envoie une demande SignedData.PKIData.controlSequence.glAddMember au GLA (A dans la Figure 5). Le membre prospectif de la GL inclut le nom de la GL dans glName, son nom dans glMember.glMemberName, son adresse dans glMember.glMemberAddress, et son certificat d'attribut dans glMember.certificates.pKC. Le membre prospectif de la GL peut aussi inclure tous certificats d'attribut associés à son certificat d'attribut dans glMember.certificates.aC, et le chemin de certification associé à ses certificats de chiffrement et d'attribut dans glMember.certificates.certPath. Le membre prospectif DOIT aussi inclure l'attribut signingTime dans cette demande.

1.a - Le membre prospectif de la GL peut facultativement appliquer la confidentialité à la demande en encapsulant les SignedData.PKIData dans un EnvelopedData (voir au paragraphe 3.2.1.2).

1.b - Le membre prospectif de la GL PEUT facultativement appliquer un autre SignedData sur les EnvelopedData (voir au paragraphe 3.2.1.2).

2 - À réception de la demande, le GLA vérifie la demande conformément au point 2 du paragraphe 4.3.1.

3 - À réception de la demande transmise, le GLO vérifie l'heure de signature et vérifie la signature du membre prospectif de la GL sur les SignedData.PKIData les plus internes et la signature du GLA sur la couche externe. Si un EnvelopedData encapsule la couche la plus interne (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLO déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

Note : Dans les cas où la GL est close et où soit a) un membre prospectif envoie directement au GLO, soit b) le GLA a par erreur transmis la demande au GLO, le GLO devrait d'abord déterminer si il honore la demande.

3.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLO PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime.

3.b - Autrement si le traitement de signature se continue et si les signatures se vérifient, le GLO s'assure que un des noms dans le certificat utilisé pour signer la demande correspond au nom dans glMember.glMemberName.

3.b.1 - Si les noms ne correspondent pas, le GLO renvoie un message SignedData.PKIResponse.controlSequence au membre prospectif avec cMCStatusInfoExt.cMCStatus.failed indiquant pourquoi le membre prospectif a été refusé dans cMCStausInfo.statusString. Cela empêche les gens d'ajouter des membres aux GL sans permission. De plus, un attribut signingTime est inclus dans la réponse.

3.b.2 - Autrement si les noms correspondent, le GLO détermine si le membre prospectif a la permission d'être ajouté. Le mécanisme sort du domaine d'application de ce document ; cependant, le GLO devrait vérifier pour voir si le glMember.glMemberName n'est pas déjà dans la GL.

3.b.2.a - Si le GLO détermine qu'il n'est pas permis au membre prospectif de se joindre à la GL, le GLO peut retourner un message SignedData.PKIResponse.controlSequence au membre prospectif avec cMCStatusInfoExt.cMCtatus.failed indiquant pourquoi le membre prospectif est refusé dans cMCStatus.statusString. De plus, un attribut signingTime est inclus dans la réponse.

3.b.2.b - Autrement si le GLO détermine qu'il est permis au membre prospectif de se joindre à la GL, le GLO vérifie le certificat d'attribut du membre.

3.b.2.b.1 - Si le certificat du membre ne peut pas être vérifié, le GLO retourne SignedData.PKIResponse.controlSequence

au membre prospectif avec cMCStatusInfoExt.cMCtatus.failed indiquant que le certificat d'attribut du membre ne s'est pas vérifié dans cMCStatus.statusString. De plus, un attribut signingTime est inclus dans la réponse. Si le GLO ne retourne pas une réponse cMCStatusInfoExt, le GLO envoie au membre prospectif un message SignedData.PKIData.controlSequence.glProvideCert lui demandant un nouveau certificat d'attribut (voir au paragraphe 4.10).

3.b.2.b.2 - Autrement si le certificat du membre se vérifie, le GLO soumet à nouveau la demande glAddMember (voir au paragraphe 3.2.5) au GLA (1 dans la Figure 5).

3.b.2.b.2.a - Le GLO applique la confidentialité à la nouvelle demande GLAddMember en encapsulant le SignedData.PKIData dans un EnvelopedData si la demande initiale était encapsulée dans un EnvelopedData (voir au paragraphe 3.2.1.2).

3.b.2.b.2.b - Le GLO peut aussi facultativement appliquer un autre SignedData sur le EnvelopedData (voir au paragraphe 3.2.1.2).

4 – Le traitement se continue comme au point 2 du paragraphe 4.3.1.

4.4 Supprimer des membre de la GL

Pour supprimer des membres des GL, le GLO ou les membres à retirer utilisent la demande glDeleteMember. Le GLA traite le GLO, et les membres qui demandent leur propre suppression font les demandes différemment. Le GLO peut soumettre la demande à tout moment pour supprimer des membres de la GL, et le GLA, une fois qu'il a vérifié que la demande vient d'un GLO enregistré, devrait supprimer le membre. Si un membre envoie la demande, le GLA doit déterminer comment la GL est administrée. Quand le GLO a initialement configuré la GL, il règle la GL à être non gérée, gérée, ou close (voir au paragraphe 3.1.1). Dans le cas de GL non gérée, le GLA traite simplement la demande du membre. Dans le cas géré, le GLA transmet les demandes du membre au GLO pour revue. Lorsque il y a plusieurs GLO pour une GL, à quel GLO transmettre la demande sort du domaine d'application de ce document. Le GLO revoit la demande et la rejette ou soumet une demande reformée au GLA. Dans le cas de GL close, le GLA ne va pas accepter de demandes des membres. Les paragraphes qui suivent décrivent le traitement pour les GLO, GLA, et membres de la GL selon l'origine de la demande, soit d'un GLO, soit de membres qui veulent être retirés. La Figure 6 décrit les interactions de protocole pour les trois options. Noter que les messages d'erreur ne sont pas décrits. De plus, le comportement pour les attributs de commande de CMC facultatifs transactionId, senderNonce, et recipientNonce ne sont pas traités dans ces procédures.

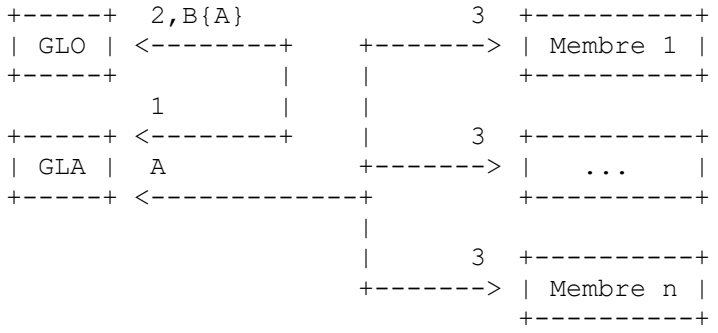


Figure 6 : Suppression de membre

Si le membre n'est pas retiré de la GL, il va continuer de recevoir et être capable de déchiffrer les données protégées avec la KEK partagée et va continuer de recevoir les changements de clés. Pour les listes non gérées, il n'y a pas de sens à un changement de clé de groupe parce que il n'est pas garanti que le membre qui demande à être retiré ne s'est pas déjà rajouté lui-même à la GL sous un nom différent. Pour les GL gérées et closes, le GLO a besoin de prendre des mesures pour s'assurer que le membre supprimé n'est pas deux fois sur la GL. Après s'en être assuré, les GL gérées et closes peuvent avoir un changement de clés pour maintenir la confidentialité du trafic envoyé par les membres du groupe. Si le GLO est sûr que le membre a été supprimé, le mécanisme de changement de clé peut être utilisé pour distribuer la nouvelle clé (voir le paragraphe 4.5 et la Section 5).

4.4.1 Suppressions initiées par le GLO

Le traitement pour les demandes glDeleteMember initiées par le GLO est le suivant :

1 - Le GLO collecte les informations pertinentes pour le ou les membres à supprimer (cela peut être fait par un moyen hors bande). Le GLO envoie alors une SignedData.PKIData.controlSequence avec une demande glDeleteMember séparée pour chaque membre de la GLA (1 dans la Figure 6). Le GLO DOIT inclure le nom de la GL dans glName et le nom du membre dans glMemberToDelete. Si la GL d'où le membre est supprimé est une GL close ou gérée, le GLO DOIT aussi générer une demande glRekey et l'inclure avec la demande glDeletemember (voir au paragraphe 4.5). Le GLO DOIT aussi inclure l'attribut signingTime dans cette demande.

1.a - Le GLO peut facultativement appliquer la confidentialité à la demande en encapsulant les SignedData.PKIData dans un EnvelopedData (voir au paragraphe 3.2.1.2).

1.b - Le GLO peut aussi facultativement appliquer un autre SignedData su la EnvelopedData (voir au paragraphe 3.2.1.2).

2 - À réception de la demande, le GLA vérifie l'attribut signingTime et vérifie la signature sur les SignedData.PKIData les plus internes. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la demande (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLA vérifie la signature externe et/ou déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

2.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLA PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

2.b - Autrement si le traitement de signature se continue et si les signatures ne peuvent pas être vérifiées, le GLA retourne une réponse cMCStatusInfoExt indiquant cMCStatus.failed et otherInfo.failInfo.badMessageCheck. De plus, un attribut signingTime est inclus dans la réponse.

2.c - Autrement si les signatures se vérifient, le GLA s'assure que la GL est prise en charge par le GLA en vérifiant que le glName correspond à un glName mémorisé sur le GLA.

2.c.1 - Si le glName n'est pas pris en charge par le GLA, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de invalidGLName. De plus, un attribut signingTime est inclus dans la réponse.

2.c.2 - Autrement si le glName est pris en charge par le GLA, le GLA vérifie si le glMemberName est présent sur la GL.

2.c.2.a - Si le glMemberName n'est pas présent sur la GL, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de notAMember. De plus, un attribut signingTime est inclus dans la réponse.

2.c.2.b - Autrement si le glMemberName est déjà sur la GL, le GLA vérifie comment la GL est administrée.

2.c.2.b.1 - Si la GL est close, le GLA vérifie que le GLO enregistré a signé la demande en s'assurant qu'un des noms dans le certificat de signature numérique utilisé pour signer la demande correspond au GLO enregistré.

2.c.2.b.1.a - Si les noms ne correspondent pas, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de closedGL. De plus, un attribut signingTime est inclus dans la réponse.

2.c.2.b.1.b - Autrement si les noms correspondent, le GLA retourne un cMCStatusInfoExt.cMCStatus.success et un attribut signingTime (2 dans la Figure 5). Le GLA prend aussi des actions administratives, qui sortent du domaine d'application de ce document, pour supprimer le membre de la GL mémorisée sur le GLA. Noter que les clés de la GL ont aussi besoin d'être changées comme décrit à la Section 5.

2.c.2.b.1.b.1 - Le GLA applique la confidentialité à la réponse en encapsulant les SignedData.PKIData dans un EnvelopedData si la demande était encapsulée dans un EnvelopedData (voir au paragraphe 3.2.1.2).

2.c.2.b.1.b.2 - Le GLA peut aussi facultativement appliquer un autre SignedData sur le EnvelopedData (voir au paragraphe 3.2.1.2).

2.c.2.b.2 - Autrement si la GL est gérée, le GLA vérifie qu'un GLO enregistré ou le membre prospectif a signé la demande. Pour les GLO, un des noms dans le certificat utilisé pour signer la demande doit correspondre à un GLO enregistré. Pour le membre prospectif, le nom dans glMember.glMemberName doit correspondre à un des noms dans le certificat

utilisé pour signer la demande.

2.c.2.b.2.a - Si le signataire n'est ni un GLO enregistré ni le membre prospectif de la GL, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de noSpam. De plus, un attribut signingTime est inclus dans la réponse.

2.c.2.b.2.b - Autrement si le signataire est un GLO enregistré, le GLA retourne un cMCStatusInfoExt.cMCStatus.success et un attribut signingTime (2 dans la Figure 6). Le GLA prend aussi des actions administratives, qui sortent du domaine d'application de ce document, pour supprimer le membre de la GL mémorisée chez le GLA. Noter que la clé de la GL va aussi être changée comme décrit à la Section 5.

2.c.2.b.2.b.1 - Le GLA applique la confidentialité à la réponse en encapsulant les SignedData.PKIData dans un EnvelopedData si la demande était encapsulée dans un EnvelopedData (voir au paragraphe 3.2.1.2).

2.c.2.b.2.b.2 - Le GLA peut aussi facultativement appliquer un autre SignedData sur les EnvelopedData (voir au paragraphe 3.2.1.2).

2.c.2.b.2.c - Autrement si le signataire est le membre prospectif, le GLA transmet la demande glDeleteMember (voir au paragraphe 3.2.3) au GLO (B{A} dans la Figure 6). Si il y a plus d'un GLO enregistré, le choix du GLO auquel la demande est transmise sort du domaine d'application de ce document. La suite du traitement de la demande par les GLO est traitée au point 3 du paragraphe 4.4.2.

2.c.2.b.2.c.1 - Le GLA applique la confidentialité à la demande transmise en encapsulant les SignedData.PKIData dans un EnvelopedData si la demande était encapsulée dans un EnvelopedData (voir au paragraphe 3.2.1.2).

2.c.2.b.2.c.2 - Le GLA peut aussi facultativement appliquer un autre SignedData sur le EnvelopedData (voir au paragraphe 3.2.1.2).

2.c.2.b.3 - Autrement si la GL est non gérée, le GLA vérifie qu'un GLO enregistré ou le membre prospectif a signé la demande. Pour les GLO, un des noms dans le certificat utilisé pour signer la demande doit correspondre au nom d'un GLO enregistré. Pour le membre prospectif, le nom dans glMember.glMemberName doit correspondre à un des noms dans le certificat utilisé pour signer la demande.

2.c.2.b.3.a - Si le signataire n'est ni le GLO ni le membre prospectif, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de noSpam. De plus, un attribut signingTime est inclus dans la réponse.

2.c.2.b.3.b - Autrement si le signataire est un GLO enregistré ou le membre, le GLA retourne un cMCStatusInfoExt.cMCStatus.success et un attribut signingTime au GLO (2 dans la Figure 6) si le GLO a signé la demande, et au membre de la GL (3 dans la Figure 6) si le membre de la GL a signé la demande. Le GLA prend aussi des actions administratives, qui sortent du domaine d'application de ce document, pour supprimer le membre de la GL mémorisée par le GLA.

2.c.2.b.3.b.1 - Le GLA applique la confidentialité à la réponse en encapsulant les SignedData.PKIData dans un EnvelopedData si la demande était encapsulée dans un EnvelopedData (voir au paragraphe 3.2.1.2).

2.c.2.b.3.b.2 - Le GLA peut aussi facultativement appliquer un autre SignedData sur le EnvelopedData (voir au paragraphe 3.2.1.2).

3 - À réception d'une réponse cMCStatusInfoExt, le GLO vérifie l'heure de signature et vérifie les signatures du GLA. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la réponse (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLO vérifie la signature externe et/ou déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

3.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLO PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

3.b - Autrement si le traitement de signature se continue et si les signatures se vérifient, le GLO vérifie qu'un des noms dans le certificat utilisé pour signer la réponse correspond au nom de la GL.

3.b.1 - Si le nom de la GL ne correspond pas au nom présent dans le certificat utilisé pour signer le message, le GLO ne

devrait pas croire la réponse.

3.b.2 - Autrement si le nom de la GL correspond au nom présent dans le certificat et,
3.b.2.a - si les signatures se vérifient et si la réponse est cMCStatusInfoExt.cMCStatus.success, le GLO a supprimé le membre de la GL. Si le membre a été supprimé d'une liste gérée et si la demande originale était signée par le membre, le GLO envoie un cMCStatusInfoExt.cMCStatus.success et un attribut signingTime au membre de la GL.
3.b.2.b - Autrement si le GLO a reçu un cMCStatusInfoExt.cMCStatus.failed avec n'importe quelle raison, le GLO peut tenter à nouveau de supprimer le membre de la GL en utilisant les informations fournies dans la réponse.

4 - À réception de la réponse cMCStatusInfoExt, le membre vérifie l'heure de signature et vérifie la ou les signatures du GLA ou du GLO. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la réponse (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLO vérifie la signature externe et/ou déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

4.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le membre prospectif PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

4.b - Autrement si le traitement de signature se continue et si les signatures se vérifient, le membre de la GL vérifie qu'un des noms dans le certificat utilisé pour signer la réponse correspond au nom de la GL.

4.b.1 - Si le nom de la GL ne correspond pas au nom présent dans le certificat utilisé pour signer le message, le membre de la GL ne devrait pas croire la réponse.

4.b.2 - Autrement si le nom de la GL correspond au nom présent dans le certificat et,
4.b.2.a -si la ou les signatures se vérifient, le membre a été supprimé de la GL ;
4.b.2.b - autrement si le membre a reçu un cMCStatusInfoExt.cMCStatus.failed avec n'importe quelle raison, le membre peut tenter à nouveau de se supprimer lui-même de la GL en utilisant les informations fournies dans la réponse.

4.4.2 Suppressions initiées par le membre

Le processus de suppression initiée par le membre de sa propre adhésion en utilisant les demandes glDeleteMember est le suivant :

1 - Le membre envoie une demande SignedData.PKIData.controlSequence.glDeleteMember au GLA (A dans la Figure 6). Le membre inclut le nom de la GL dans glName et le propre nom du membre dans glMemberToDelete. Le membre de la GL DOIT aussi inclure l'attribut signingTime dans cette demande.

1.a - Le membre peut facultativement appliquer la confidentialité à la demande en encapsulant le SignedData.PKIData dans un EnvelopedData (voir au paragraphe 3.2.1.2).

1.b - Le membre peut aussi facultativement appliquer un autre SignedData sur le EnvelopedData (voir au paragraphe 3.2.1.2).

2 - À réception de la demande, le GLA vérifie la demande conformément au point 2 du paragraphe 4.4.1.

3 - À réception de la demande transmise, le GLO vérifie l'heure de signature et vérifie la signature du membre sur les SignedData.PKIData les plus internes et la signature du GLA sur la couche externe. Si un EnvelopedData encapsule la couche la plus interne (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLO déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

Note : dans le cas où la GL est close et soit (a) un membre prospectif envoie directement au GLO, soit (b) le GLA a par erreur transmis la demande au GLO, le GLO devrait d'abord déterminer si il honorera la demande.

3.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLO PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

3.b - Autrement si le traitement de signature se continue et si les signatures ne peuvent pas être vérifiées, le GLO retourne une réponse cMCStatusInfoExt indiquant cMCStatus.failed et otherInfo.failInfo.badMessageCheck et un attribut signingTime.

3.c - Autrement si les signatures se vérifient, le GLO s'assure qu'un des noms dans les certificats utilisés pour signer la demande correspond au nom dans glMemberToDelete.

3.c.1 - Si les noms ne correspondent pas, le GLO renvoie un message SignedData.PKIResponse.controlSequence au membre prospectif avec cMCStatusInfoExt.cMCtatus.failed indiquant pourquoi le membre prospectif a été refusé dans cMCStatusInfoExt.statusString. Cela empêche les gens d'ajouter des membres aux GL sans leur permission. De plus, un attribut signingTime est inclus dans la réponse.

3.c.2 - Autrement si les noms correspondent, le GLO soumet à nouveau la demande glDeleteMember (voir au paragraphe 3.2.5) au GLA (1 dans la Figure 6). Le GLO s'assure que le glMemberName est déjà dans la GL. Le GLO génère aussi une demande glRekey et l'inclut avec la demande GLDeleteMember (voir au paragraphe 4.5).

3.c.2.a - Le GLO applique la confidentialité à la nouvelle demande GLDeleteMember en encapsulant les SignedData.PKIData dans un EnvelopedData si la demande initiale était encapsulée dans un EnvelopedData (voir au paragraphe 3.2.1.2).

3.c.2.b - Le GLO peut aussi facultativement appliquer un autre SignedData sur le EnvelopedData (voir au paragraphe 3.2.1.2).

4 – La suite du traitement est comme au point 2 du paragraphe 4.4.1.

4.5 Demande de changement de clé de la GL

De temps en temps, la GL va devoir changer ses clés. Voici certaines de ces situations :

- Quand un membre est retiré d'une GL close ou gérée. Dans ce cas, la PKIData.controlSequence contenant le glDeleteMember devrait contenir une demande glRekey.

- Selon la politique, quand un membre est retiré d'une GL non gérée. Si la politique est de changer la clé de la GL, la PKIData.controlSequence contenant le glDeleteMember pourrait aussi contenir une demande glRekey ou un moyen hors bande pourrait être utilisé pour dire au GLA de changer la clé de la GL. Il n'est pas conseillé de changer les clés des GL non gérées quand des membres sont supprimés.

- Quand la KEK partagée en cours a été compromise.

- Quand la KEK partagée en cours est sur le point d'expirer. On considère deux cas :
  -- Si le GLO contrôle le changement de clé de la GL, le GLA ne devrait pas supposer qu'une nouvelle KEK partagée devrait être distribuée, mais plutôt attendre le message glRekey.
  -- Si le GLA contrôle le changement de clé de la GL, le GLA devrait initier un message glKey comme spécifié à la Section 5.

Si le generationCounter (voir au paragraphe 3.1.1) est réglé à une valeur supérieure à un (1) et si le GLO contrôle le changement de clé de la GL, le GLO peut générer un glRekey à tout moment avant l'expiration de la dernière KEK partagée. Pour se donner une marge de sécurité, le GLO devrait demander un changement de clé une (1) "duration" avant l'expiration de la dernière KEK partagée.

Le GLA et le GLO sont les seules entités autorisées à initier un changement de clé GL. Le GLO indique si il va contrôler les changement de clés ou si le GLA va le faire quand il alloue la KEK partagée à la GL (voir au paragraphe 3.1.1). Le GLO peut initier un changement de clé de la GL à tout moment. Le GLA peut être configuré à changer automatiquement les clés à la GL avant l'expiration de la KEK partagée (la durée avant l'expiration est une décision de mise en œuvre). Le GLA peut aussi changer automatiquement les clés des GL qui ont été compromises, mais ceci est traité à la Section 5. La Figure 7 décrit les interactions de protocole pour demander un changement de clé de GL. Noter que les messages d'erreur ne sont pas décrits. De plus, le comportement des attributs de commande facultatifs de CMC transactionId, senderNonce, et recipientNonce n'est pas traité dans ces procédures.

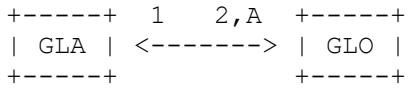


Figure 7 : Demande de changement de clé de GL

4.5.1 Demande de changement de clé initiée par le GLO

Le processus des demandes glRekey initiées par le GLO est le suivant :

1 - Le GLO envoie une demande SignedData.PKIData.controlSequence.glRekey au GLA (1 dans la Figure 7). Le GLO inclut le glName. Si glAdministration et glKeyNewAttributes sont omis, il n'y a alors pas de changement par rapport aux valeurs précédemment enregistrées de la GL pour ces champs. Si le GLO veut forcer un changement de clé pour toutes les KEK partagées en instance, il inclut le glRekeyAllGLKeys réglé à VRAI. Le GLO DOIT aussi inclure un attribut signingTime dans cette demande.

1.a - Le GLO peut facultativement appliquer la confidentialité à la demande en encapsulant les SignedData.PKIData dans un EnvelopedData (voir au paragraphe 3.2.1.2).

1.b - Le GLO peut aussi facultativement appliquer un autre SignedData sur les EnvelopedData (voir au paragraphe 3.2.1.2).

2 - À réception de la demande, le GLA vérifie l'heure de signature et vérifie la signature sur les SignedData.PKIData les plus internes. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la demande (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLA vérifie la signature externe et/ou déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

2.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLA PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

2.b - Autrement si le traitement de signature se continue et si les signatures ne se vérifient pas, le GLA retourne une réponse cMCStatusInfoExt indiquant cMCStatus.failed et otherInfo.failInfo.badMessageCheck. De plus, un attribut signingTime est inclus dans la réponse.

2.c - Autrement si les signatures se vérifient, le GLA s'assure que la GL est prise en charge par le GLA en vérifiant que le glName correspond à un glName mémorisé dans le GLA.

2.c.1 - Si le glName présent ne correspond pas à une GL mémorisée dans le GLA, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de invalidGLName. De plus, un attribut signingTime est inclus dans la réponse.

2.c.2 - Autrement si le glName présent correspond à une GL mémorisée dans le GLA, le GLA vérifie qu'un GLO enregistré a signé la demande en s'assurant que un des noms dans le certificat utilisé pour signer la demande est un GLO enregistré.

2.c.2.a - Si les noms ne correspondent pas, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de noGLONameMatch. De plus, un attribut signingTime est inclus dans la réponse.

2.c.2.b - Autrement si les noms correspondent, le GLA vérifie les valeurs de glNewKeyAttribute.

2.c.2.b.1 - Si la nouvelle valeur pour requestedAlgorithm n'est pas prise en charge, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de unsupportedAlgorithm. De plus, un attribut signingTime est inclus dans la réponse.

2.c.2.b.2 - Autrement si la durée de la nouvelle valeur n'est pas supportable (le déterminer sort du domaine d'application de ce document) le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de unsupportedDuration. De plus, un attribut signingTime est inclus dans la réponse.

2.c.2.b.3 - Autrement si la GL ne peut pas être prise en charge pour d'autres raisons que le GLA ne souhaite pas divulguer, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de unspecified. De plus, un attribut signingTime est inclus dans la réponse.

2.c.2.b.4 - Autrement si les nouveaux requestedAlgorithm et duration sont acceptables ou si le glNewKeyAttributes est omis, le GLA retourne un cMCStatusInfoExt.cMCStatus.success et un attribut sigingTime (2 dans la Figure 7). Le GLA

utilise aussi le message glKey pour distribuer la KEK partagée de changement de clé (voir la Section 5).

2.c.2.b.4.a - Le GLA applique la confidentialité à la réponse en encapsulant les SignedData.PKIData dans un EnvelopedData si la demande était encapsulée dans un EnvelopedData (voir au paragraphe 3.2.1.2).

2.c.2.b.4.b - Le GLA peut aussi facultativement appliquer un autre SignedData sur le EnvelopedData (voir au paragraphe 3.2.1.2).

3 - À réception de la réponse cMCStatusInfoExt, le GLO vérifie l'heure de signature et vérifie les signatures du GLA. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la réponse transmise (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLO vérifie la signature externe et/ou déchiffre la réponse transmise avant de vérifier la signature sur les SignedData les plus internes.

3.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLA PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

3.b - Autrement si le traitement de signature se continue et si les signatures se vérifient, le GLO vérifie qu'un des noms dans le certificat utilisé pour signer la réponse correspond au nom de la GL.

3.b.1 - Si le nom de la GL ne correspond pas au nom présent dans le certificat utilisé pour signer le message, le GLO ne devrait pas croire la réponse.

3.b.2 - Autrement si le nom de la GL correspond au nom présent dans le certificat et,
3.b.2.a - si les signatures se vérifient et si la réponse est cMCStatusInfoExt.cMCStatus.success, le GLO a réussi à changer les clés de la GL ;
3.b.2.b - autrement, si le GLO a reçu un cMCStatusInfoExt.cMCStatus.failed avec n'importe quelle raison, le GLO peut tenter à nouveau de changer les clés de la GL en utilisant les informations fournies dans la réponse.

4.5.2 Demandes de changement de clé initié par le GLA

Si le GLA est chargé de changer les clés de la GL, le GLA va automatiquement produire un message glKey (voir la Section 5). En plus, le GLA va générer un cMCStatusInfoExt pour indiquer à la GL qu'un changement de clé réussi s'est produit. Le processus du changement de clé initié par le GLA est le suivant :

1 - Le GLA génère pour tous les GLO une SignedData.PKIData.controlSequence.cMCStatusInfoExt.cMCStatus de succès et inclut un attribut signingTime (A dans la Figure 7).

1.a - Le GLA peut facultativement appliquer la confidentialité à la demande en encapsulant les SignedData.PKIData dans un EnvelopedData (voir au paragraphe 3.2.1.2).

1.b - Le GLA peut aussi facultativement appliquer un autre SignedData sur le EnvelopedData (voir au paragraphe 3.2.1.2).

2 - À réception de la réponse cMCStatusInfoExt.cMCStatus.success, le GLO vérifie l'heure de signature et vérifie la ou les signatures du GLA. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la réponse transmise (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLO DOIT vérifier la signature externe et/ou déchiffrer la couche externe avant de vérifier la signature sur les SignedData les plus internes.

2.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLO PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

2.b - Autrement si le traitement de signature se continue et si les signatures se vérifient, le GLO vérifie qu'un des noms dans le certificat utilisé pour signer la réponse correspond au nom de la GL.

2.b.1 - Si le nom de la GL ne correspond pas au nom présent dans le certificat utilisé pour signer le message, le GLO ne devrait pas croire la réponse.

2.b.2 - Autrement si le nom de la GL correspond au nom présent dans le certificat et si la réponse est cMCStatusInfoExt.cMCStatus.success, le GLO sait que le GLA a réussi a changer la clé de la GL.

4.6 Changer le GLO

La gestion des GL gérées et closes peut devenir difficile pour un GLO si le nombre des membres de la GL devient important. Pour prendre en charge la distribution de la charge de travail, les GLA prennent en charge d'avoir les GL gérées par plusieurs GLO. Les messages glAddOwner et glRemoveOwner sont destinés à prendre en charge l'ajout et la suppression des GLO enregistrés. La Figure 8 décrit les interactions de protocole pour envoyer les messages glAddOwner et glRemoveOwner et les messages de réponse résultants. Noter que les messages d'erreur ne sont pas montrés. De plus, le comportement pour les attributs de commande de CMC facultatifs transactionId, senderNonce, et recipientNonce n'est pas traité dans ces procédures.

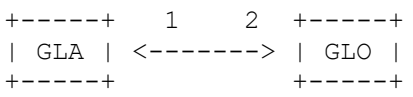


Figure 8 : Ajout et suppression de GLO

Le processus pour glAddOwner et glDeleteOwner est le suivant :

1 - Le GLO envoie une demande SignedData.PKIData.controlSequence.glAddOwner ou glRemoveOwner au GLA (1 dans la Figure 8). Le GLO inclut le nom de la GL dans glName, et le nom et l'adresse du GLO dans glOwnerName et glOwnerAddress, respectivement. Le GLO DOIT aussi inclure l'attribut signingTime dans cette demande.

1.a - Le GLO peut facultativement appliquer la confidentialité à la demande en encapsulant les SignedData.PKIData dans un EnvelopedData (voir au paragraphe 3.2.1.2).

1.b - Le GLO peut aussi facultativement appliquer un autre SignedData sur les EnvelopedData (voir au paragraphe 3.2.1.2).

2 - À réception de la demande glAddOwner ou glRemoveOwner, le GLA vérifie l'heure de signature et vérifie la ou les signatures du GLO. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la demande (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLA vérifie la signature externe et/ou déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

2.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLA PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

2.b - Autrement si le traitement de signature se continue et si les signatures ne peuvent pas être vérifiées, le GLA retourne une réponse cMCStatusInfoExt indiquant cMCStatus.failed et otherInfo.failInfo.badMessageCheck. De plus, un attribut signingTime est inclus dans la réponse.

2.c - Autrement si les signatures se vérifient, le GLA s'assure que la GL est prise en charge en vérifiant que le glName correspond à un glName mémorisé sur le GLA.

2.c.1 - Si le glName n'est pas pris en charge par le GLA, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de invalidGLName. De plus, un attribut signingTime est inclus dans la réponse.

2.c.2 - Autrement si le glName est pris en charge par le GLA, le GLA s'assure qu'un GLO enregistré a signé la demande glAddOwner ou glRemoveOwner en vérifiant qu'un des noms présents dans le certificat de signature numérique utilisé pour signer la demande glAddOwner ou glDeleteOwner correspond au nom d'un GLO enregistré.

2.c.2.a - Si les noms ne correspondent pas, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de noGLONameMatch. De plus, un attribut signingTime est inclus dans la réponse.

2.c.2.b - Autrement si les noms correspondent, le GLA retourne un cMCStatusInfoExt.cMCStatus.success et un attribut signingTime (2 dans la Figure 4). Le GLA prend aussi des actions administratives pour associer le nouveau glOwnerName à la GL dans le cas de glAddOwner ou pour dissocier l'ancien glOwnerName de la GL dans le cas de glRemoveOwner.

2.c.2.b.1 - Le GLA applique la confidentialité à la réponse en encapsulant la SignedData.PKIResponse dans un EnvelopedData si la demande était encapsulée dans un EnvelopedData (voir au paragraphe 3.2.1.2).

2.c.2.b.2 - Le GLA peut aussi facultativement appliquer un autre SignedData sur les EnvelopedData (voir au paragraphe 3.2.1.2).

3 - À réception de la réponse cMCStatusInfoExt, le GLO vérifie l'heure de signature et vérifie la ou les signatures du GLA. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la réponse (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLO vérifie la signature externe et/ou déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

3.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLO PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

3.b - Autrement si le traitement de signature se continue et si les signatures se vérifient, le GLO vérifie qu'un des noms dans le certificat utilisé pour signer la réponse correspond au nom de la GL.

3.b.1 - Si le nom de la GL ne correspond pas au nom présent dans le certificat utilisé pour signer le message, le GLO ne devrait pas croire la réponse.

3.b.2 - Autrement si le nom de la GL ne correspond pas au nom présent dans le certificat et,
3.b.2.a - si les signatures se vérifient et si la réponse était cMCStatusInfoExt.cMCStatus.success, le GLO a réussi à ajouter ou retirer le GLO ;
3.b.2.b -autrement si les signatures se vérifient et si la réponse était cMCStatusInfoExt.cMCStatus.failed avec n'importe quelle raison, le GLO peut tenter à nouveau d'ajouter ou supprimer le GLO en utilisant les informations fournies dans la réponse.

4.7 Indiquer la compromission de la KEK

Il peut arriver que la KEK partagée soit compromise. Les membres de la GL et les GLO utilisent glkCompromise pour dire au GLA que la KEK partagée a été compromise. La Figure 9 décrit les interactions de protocole pour la compromission de clé de GL. Noter que les messages d'erreur ne sont pas montrés. De plus, le comportement pour les attributs de commande de CMC facultatifs transactionId, senderNonce, et recipientNonce n'est pas traité dans ces procédures.

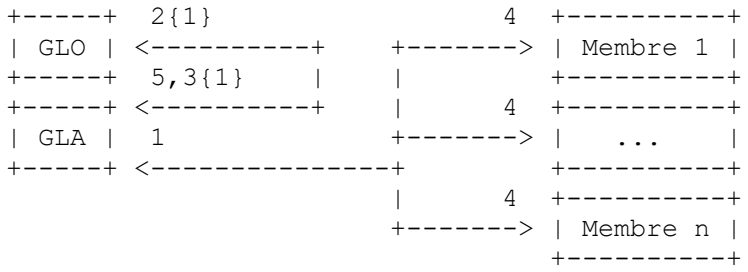


Figure 9 : Compromission de clé de GL

4.7.1 Message KEK compromise initié par un membre de la GL

Le traitement des messages glkCompromise initiés par le membre de la GL est le suivant :

1 - Le membre de la GL envoie une demande SignedData.PKIData.controlSequence.glkCompromise au GLA (1 dans la Figure 9). Le membre de la GL inclut le nom de la GL dans GeneralName. Le membre de la GL DOIT aussi inclure l'attribut signingTime dans cette demande.

1.a - Le membre de la GL peut facultativement appliquer la confidentialité à la demande en encapsulant les SignedData.PKIData dans un EnvelopedData (voir au paragraphe 3.2.1.2). Le glkCompromise peut être inclus dans un EnvelopedData généré avec la KEK partagée compromise.

1.b - Le membre de la GL peut aussi facultativement appliquer un autre SignedData sur le EnvelopedData (voir au paragraphe 3.2.1.2).

2 - À réception de la demande glkCompromise, le GLA vérifie l'heure de signature et vérifie la ou les signatures du

membre de la GL. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la demande (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLA vérifie la signature externe et/ou déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

2.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLA PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

2.b - Autrement si le traitement de signature se continue et si les signatures ne peuvent pas être vérifiées, le GLA retourne une réponse cMCStatusInfoExt indiquant cMCStatus.failed et otherInfo.failInfo.badMessageCheck. De plus, un attribut signingTime est inclus dans la réponse.

2.c - Autrement si les signatures se vérifient, le GLA s'assure que la GL est prise en charge en vérifiant que le nom de la GL indiquée correspond à un glName mémorisé sur le GLA.

2.c.1 - Si le glName n'est pas pris en charge par le GLA, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de invalidGLName. De plus, un attribut signingTime est inclus dans la réponse.

2.c.2 - Autrement si le glName est pris en charge par le GLA, le GLA vérifie qui a signé la demande. Pour les GLO, un des noms dans le certificat utilisé pour signer la demande doit correspondre à un GLO enregistré. Pour le membre, le nom dans glMember.glMemberName doit correspondre à un des noms dans le certificat utilisé pour signer la demande.

2.c.2.a - Si le GLO a signé la demande, le GLA génère un message glKey comme décrit à la Section 5 pour changer la clé de la GL (4 dans la Figure 9).

2.c.2.b - Autrement si quelqu'un d'autre que le GLO a signé la demande, le GLA transmet le message glkCompromise (voir au paragraphe 3.2.3) au GLO (2{1} dans la Figure 9). Si il y a plus d'un GLO, le choix du GLO auquel la demande est transmise sort du domaine d'application de ce document. La suite du traitement par le GLO est discutée au paragraphe 4.7.2.

4.7.2 Message KEK compromise initié par le GLO

Le traitement du message glkCompromise initié par le GLO est le suivant :

1 - Le GLO :

1.a – Génère lui-même le message glkCompromise en envoyant une demande SignedData.PKIData.controlSequence.glkCompromise au GLA (5 dans la Figure 9). Le GLO inclut le nom de la GL dans GeneralName. Le GLO DOIT aussi inclure un attribut signingTime dans cette demande.

1.a.1 - Le GLO peut facultativement appliquer la confidentialité à la demande en encapsulant les SignedData.PKIData dans un EnvelopedData (voir au paragraphe 3.2.1.2). Le glkCompromise peut être inclus dans un EnvelopedData généré avec la KEK partagée compromise.

1.a.2 - Le GLO peut aussi facultativement appliquer un autre SignedData sur les EnvelopedData (voir au paragraphe 3.2.1.2).

1.b - Autrement, il vérifie l'heure de signature et vérifie les signatures du GLA et du membre de la GL sur le message glkCompromise transmis. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la demande (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLO vérifie la signature externe et/ou déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

1.b.1 - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLO PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

1.b.2 - Autrement si le traitement de signature se continue et si les signatures ne peuvent pas être vérifiées, le GLO retourne une réponse cMCStatusInfoExt indiquant cMCStatus.failed et otherInfo.failInfo.badMessageCheck. De plus, un attribut signingTime est inclus dans la réponse.

1.b.2.a - Si les signatures se vérifient, le GLO vérifie que les noms dans le certificat correspondent au nom du signataire

(c'est-à-dire, que le nom dans le certificat utilisé pour signer la demande du membre de la GL est celui du membre de la GL).

1.b.2.a.1 - Si un des noms ne correspond pas, le GLO ne devrait pas faire confiance au signataire et il ne devrait pas transmettre le message au GLA.

1.b.2.a.2 - Autrement si les noms correspondent et si les signatures se vérifient, le GLO détermine si il retransmet le message glkCompromise au GLA (3{1} dans la Figure 9). La suite du traitement par le GLA est au point 2 du paragraphe 4.7.1. Le GLO peut aussi retourner une réponse au membre prospectif avec cMCStatusInfoExt.cMCtatus.success indiquant que le message glkCompromise a bien été reçu.

4.8 Demande de rafraîchissement de la KEK

Il peut arriver que les membres de la GL aient perdu de façon irrécupérable leur KEK partagée. La KEK partagée n'est pas compromise et un changement de clé de la GL entière n'est pas nécessaire. Les membres de la GL utilisent le message glkRefresh pour demander que la ou les KEK partagées leur soient redistribuées. La Figure 10 décrit les interactions de protocole pour le rafraîchissement de clé de GL. Noter que les messages d'erreur ne sont pas montrés. De plus, le comportement pour les attributs de commande de CMC facultatifs transactionId, senderNonce, et recipientNonce n'est pas traité dans ces procédures.

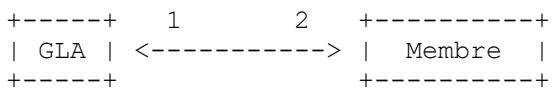


Figure 10 – Rafraîchissement de KEK de la GL

Le traitement de glkRefresh est le suivant :

1 - Le membre de la GL envoie une demande SignedData.PKIData.controlSequence.glkRefresh au GLA (1 dans la Figure 10). Le membre de la GL inclut le nom de la GL dans GeneralName. Le membre de la GL DOIT aussi inclure un attribut signingTime dans cette demande.

1.a - Le membre de la GL peut facultativement appliquer la confidentialité à la demande en encapsulant les SignedData.PKIData dans un EnvelopedData (voir au paragraphe 3.2.1.2).

1.b - Le membre de la GL peut aussi facultativement appliquer un autre SignedData sur les EnvelopedData (voir au paragraphe 3.2.1.2).

2 - À réception de la demande glkRefresh, le GLA vérifie l'heure de signature et vérifie la ou les signatures du membre de la GL. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la demande (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLA vérifie la signature externe et/ou déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

2.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLA PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

2.b - Autrement si le traitement de signature se continue et si les signatures ne peuvent pas être vérifiées, le GLA retourne une réponse cMCStatusInfoExt indiquant cMCStatus.failed et otherInfo.failInfo.badMessageCheck. De plus, un attribut signingTime est inclus dans la réponse.

2.c - Autrement si les signatures se vérifient, le GLA s'assure que la GL est prise en charge en vérifiant que le GLGeneralName correspond à un glName mémorisé dans la GLA.

2.c.1 - Si le nom de la GL n'est pas pris en charge par le GLA, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de invalidGLName. De plus, un attribut signingTime est inclus dans la réponse.

2.c.2 - Autrement si le glName est pris en charge par le GLA, le GLA s'assure que le membre de la GL est sur la GL :
2.c.2.a - Si le glMemberName n'est pas présent sur la GL, le GLA retourne une réponse indiquant cMCStatusInfoExt avec cMCStatus.failed et une valeur de otherInfo.extendedFailInfo.SKDFailInfo de noSpam. De plus, un attribut

signingTime est inclus dans la réponse.

2.c.2.b - Autrement si le glMemberName est présent dans la GL, le GLA retourne cMCStatusInfoExt.cMCStatus.success, un attribut signingTime, et un message glKey (2 dans la Figure 10) comme décrit dans la Section 5.

4.9 Demande et réponse d'interrogation de GLA

Il arrive parfois qu'un GLO ait des problèmes pour établir une GL parce que il ne sait pas le ou les algorithmes ou quelque autre caractéristique que prend en charge le GLA. Il peut aussi arriver que des membres prospectifs de la GL ou des membres de la GL aient besoin de savoir quelque chose sur le GLA (ces demandes ne sont pas définies dans ce document). Les messages glaQueryRequest et glaQueryResponse ont été définis pour prendre en charge la détermination de ces informations. La Figure 11 décrit les interactions de protocole pour glaQueryRequest et glaQueryResponse. Noter que les messages d'erreur ne sont pas montrés. De plus, le comportement pour les attributs de commande de CMC facultatifs transactionId, senderNonce, et recipientNonce n'est pas traité dans ces procédures.

Figure 11 – Demande et réponse d'interrogation de GLA

Le traitement de glaQueryRequest et glaQueryResponse est le suivant :

1 - Le GLO, le membre de la GL, ou le membre prospectif de la GL envoie une demande SignedData.PKIData.controlSequence.glaQueryRequest au GLA (1 dans la Figure 11). Le GLO, membre de la GL, ou membre prospectif de la GL indique les informations qu'il est intéressé à recevoir du GLA. De plus, un attribut signingTime est inclus dans la réponse.

1.a - Le GLO, membre de la GL, ou membre prospectif de la GL peut facultativement appliquer la confidentialité à la demande en encapsulant les SignedData.PKIData dans un EnvelopedData (voir au paragraphe 3.2.1.2).

1.b - Le GLO, membre de la GL, ou membre prospectif de la GL peut aussi facultativement appliquer un autre SignedData sur les EnvelopedData (voir au paragraphe 3.2.1.2).

2 - À réception de la glaQueryRequest, le GLA détermine si il accepte les messages glaQueryRequest.

2.a - Si le GLA n'accepte pas les messages glaQueryRequest, le GLA retourne une réponse cMCStatusInfoExt indiquant cMCStatus.noSupport et toutes les autres informations dans statusString.

2.b - Autrement si le GLA accepte bien les GLAQueryRequest, le GLA vérifie l'heure de signature et vérifie les signatures du GLO, membre de la GL, ou membre prospectif de la GL. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la demande (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLA vérifie la signature externe et/ou déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

2.b.1 - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLA PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

2.b.2 - Autrement si le traitement de signature se continue et si les signatures ne peuvent pas être vérifiées, le GLA retourne une réponse cMCStatusInfoExt indiquant cMCStatus.failed et otherInfo.failInfo.badMessageCheck. De plus, un attribut signingTime est inclus dans la réponse.

2.b.3 - Autrement si les signatures se vérifient, le GLA retourne une glaQueryResponse (2 dans la Figure 11) avec la réponse correcte si le glaRequestType est pris en charge ou retourne une réponse cMCStatusInfoExt indiquant cMCStatus.noSupport si le glaRequestType n'est pas pris en charge. De plus, un attribut signingTime est inclus dans la réponse.

2.b.3.a - Le GLA applique la confidentialité à la réponse en encapsulant la SignedData.PKIResponse dans un EnvelopedData si la demande était encapsulée dans un EnvelopedData (voir au paragraphe 3.2.1.2).

2.b.3.b - Le GLA peut aussi facultativement appliquer un autre SignedData sur les EnvelopedData (voir au paragraphe 3.2.1.2).

3 - À réception de la glaQueryResponse, le GLO, membre de la GL, ou membre prospectif de la GL, vérifie l'heure de signature et vérifie la ou les signatures du GLA. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la réponse (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLO, membre de la GL, ou membre prospectif de la GL, vérifie la signature externe et/ou déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

3.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLO, membre de la GL, ou membre prospectif de la GL, PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

3.b - Autrement si le traitement de signature se continue et si les signatures ne se vérifient pas, le GLO, membre de la GL, ou membre prospectif de la GL, retourne une réponse cMCStatusInfoExt indiquant cMCStatus.failed et otherInfo.failInfo.badMessageCheck. De plus, un attribut signingTime est inclus dans la réponse.

3.c - Autrement si les signatures se vérifient, alors le GLO, membre de la GL, ou membre prospectif de la GL, vérifie qu'un des noms dans le certificat utilisé pour signer la réponse correspond au nom de la GL.

3.c.1 - Si le nom de la GL ne correspond pas au nom présent dans le certificat utilisé pour signer le message, le GLO ne devrait pas croire la réponse.

3.c.2 - Autrement si le nom de la GL correspond au nom présent dans le certificat et si la réponse était glaQueryResponse, alors le GLO, membre de la GL, ou membre prospectif de la GL peut utiliser les informations qui y sont contenues.

4.10 Mise à jour de certificat de membre

Quand le GLO génère une demande glAddMember, quand le GLA génère un message glKey, ou quand le GLA traite un glAddMember, il peut y avoir des cas où le certificat du membre de la GL est arrivé à expiration ou est invalide. Dans ce cas, le GLO ou GLA peut demander que le membre de la GL fournisse un nouveau certificat pour éviter que le GLA soit incapable de générer un message glKey pour le membre de la GL. Il pourrait aussi arriver que le membre de la GL sache que son certificat est sur le point d'arriver à expiration ou a été révoqué, et le membre de la GL ne va plus être capable de recevoir les changements de clé de GL. Le comportement pour les attributs de commande de CMC facultatifs transactionId, senderNonce, et recipientNonce n'est pas traité dans ces procédures.

4.10.1 Mise à jour de certificat de membre initiée par le GLO et le GLA

Le traitement du glUpdateCert initié par le GLO est le suivant :

1 - Le GLO ou GLA envoie une demande SignedData.PKIData.controlSequence.glProvideCert au membre de la GL. Le GLO ou GLA indique le nom de la GL dans glName et le nom du membre de la GL dans glMemberName. De plus, un attribut signingTime est inclus dans cette demande.

1.a - Le GLO ou GLA peut facultativement appliquer la confidentialité à la demande en encapsulant les SignedData.PKIData dans un EnvelopedData (voir au paragraphe 3.2.1.2). Si le PKC du membre de la GL a été révoqué, le GLO ou GLA ne devrait pas l'utiliser pour générer les EnvelopedData qui encapsulent la demande glProvideCert.

1.b - Le GLO ou GLA peut aussi facultativement appliquer un autre SignedData sur le EnvelopedData (voir au paragraphe 3.2.1.2).

2 - À réception du message glProvideCert, le membre de la GL vérifie l'heure de signature et vérifie la ou les signatures du GLO ou GLA. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la réponse (voir au paragraphe 3.2.1.2 ou 3.2.2) le membre de la GL vérifie la signature externe et/ou déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

2.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le membre de la GL PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

2.b - Autrement si le traitement de signature se continue et si les signatures ne peuvent pas être vérifiées, le membre de la GL retourne une réponse cMCStatusInfoExt indiquant cMCStatus.failed et otherInfo.failInfo.badMessageCheck. De

plus, un attribut signingTime est inclus dans la réponse.

2.c - Autrement si les signatures se vérifient, le membre de la GL génère une Signed.PKIResponse.controlSequence.glUpdateCert qui inclut le nom de la GL dans glName, le nom du membre dans glMember.glMemberName, le certificat d'attribut du membre dans glMember.certificates.pKC. Le membre de la GL peut aussi inclure tous les certificats d'attribut associés au certificat d'attribut du membre dans glMember.certificates.aC, et le chemin de certification associé aux certificats de chiffrement et d'attribut du membre dans glMember.certificates.certPath. De plus, un attribut signingTime est inclus dans la réponse.

2.c.1 - Le membre de la GL peut facultativement appliquer la confidentialité à la demande en encapsulant la SignedData.PKIResponse dans un EnvelopedData (voir au paragraphe 3.2.1.2). Si le PKC du membre de la GL a été révoqué, le membre de la GL ne devrait pas l'utiliser pour générer les EnvelopedData qui encapsulent la demande glProvideCert.

2.c.2 - Le membre de la GL peut aussi facultativement appliquer un autre SignedData sur le EnvelopedData (voir au paragraphe 3.2.1.2).

3 - À réception du message glUpdateCert, le GLO ou GLA vérifie l'heure de signature et vérifie la ou les signatures du membre de la GL. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la réponse (voir au paragraphe 3.2.1.2 ou 3.2.2) le membre de la GL vérifie la signature externe et/ou déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

3.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLO ou GLA PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

3.b - Autrement si le traitement de signature se continue et si les signatures ne peuvent pas être vérifiées, le GLO ou GLA retourne une réponse cMCStatusInfoExt indiquant cMCStatus.failed et otherInfo.failInfo.badMessageCheck. De plus, un attribut signingTime est inclus dans la réponse.

3.c - Autrement si les signatures se vérifient, le GLO ou GLA vérifie le certificat d'attribut du membre.

3.c.1 - Si le certificat d'attribut du membre ne peut pas être vérifié, le GLO retourne soit une autre demande glProvideCert, soit une cMCStatusInfoExt avec cMCStatus.failed et la raison dans cMCStatus.statusString. glProvideCert devrait être retourné seulement un certain nombre de fois parce que si le membre de la GL n'a pas un certificat valide, il ne va jamais être capable d'en retourner un. De plus, un attribut signingTime est inclus dans la réponse.

3.c.2 - Autrement si le certificat d'attribut du membre ne peut pas être vérifié, le GLA retourne une autre demande glProvideCert au membre de la GL ou un cMCStatusInfoExt avec cMCStatus.failed et la raison dans cMCStatus.statusString au GLO. glProvideCert devrait être retourné seulement un certain nombre de fois parce que si le membre de la GL n'a pas un certificat valide, il ne va jamais être capable d'en retourner un. De plus, un attribut signingTime est inclus dans la réponse.

3.c.3 - Autrement si le certificat d'attribut du membre se vérifie, le GLO ou GLA va l'utiliser dans les demandes glAddMember suivantes et les messages glKey associés au membre de la GL.

4.10.2 Mise à jour de certificat de membre initiés par un membre de la GL

Le traitement d'un glUpdateCert non sollicité du membre de la GL est le suivant :

1 - Le membre de la GL envoie un Signed.PKIData.controlSequence.glUpdateCert qui inclut le nom du GL dans glName, le nom du membre dans glMember.glMemberName, le certificat d'attribut du membre dans glMember.certificates.pKC. Le membre de la GL peut aussi inclure tous certificats d'attribut associés au certificat d'attribut du membre dans glMember.certificates.aC, et le chemin de certification associé aux certificats de chiffrement et d'attribut du membre dans glMember.certificates.certPath. Le membre de la GL DOIT aussi inclure un attribut signingTime dans cette demande.

1.a - Le membre de la GL peut facultativement appliquer la confidentialité à la demande en encapsulant le SignedData.PKIData dans un EnvelopedData (voir au paragraphe 3.2.1.2). Si le PKC du membre de la GL a été révoqué, le GLO ou GLA ne devrait pas l'utiliser pour générer les EnvelopedData qui encapsulent la demande glProvideCert.

1.b - Le membre de la GL peut aussi facultativement appliquer un autre SignedData sur le EnvelopedData (voir au paragraphe 3.2.1.2).

2 - À réception du message glUpdateCert, le GLA vérifie l'heure de signature et vérifie la ou les signatures du membre de la GL. Si un SignedData et/ou EnvelopedData supplémentaire encapsule la réponse (voir au paragraphe 3.2.1.2 ou 3.2.2) le GLA vérifie la signature externe et/ou déchiffre la couche externe avant de vérifier la signature sur les SignedData les plus internes.

2.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLA PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

2.b - Autrement si le traitement de signature se continue et si les signatures ne peuvent pas être vérifiées, le GLA retourne une réponse cMCStatusInfoExt indiquant cMCStatus.failed et otherInfo.failInfo.badMessageCheck.

2.c - Autrement si les signatures se vérifient, le GLA vérifie le certificat d'attribut du membre.

2.c.1 - Si le certificat d'attribut du membre ne peut pas être vérifié, le GLA retourne une autre demande glProvideCert au membre de la GL ou un cMCStatusInfoExt avec cMCStatus.failed et la raison dans cMCStatus.statusString au GLO. glProvideCert ne devrait pas être retourné indéfiniment ; si le membre de la GL n'a pas de certificat valide, il ne va jamais être capable d'en retourner un. De plus, un attribut signingTime est inclus dans la réponse.

2.c.2 - Autrement si le certificat d'attribut du membre se vérifie, le GLA va l'utiliser dans les demandes glAddMember suivantes et les messages glKey associés au membre de la GL. Le GLA transmet aussi le message glUpdateCert au GLO.

5. Message de distribution

Le GLA utilise le message glKey pour distribuer les nouvelles KEK partagées après la réception de demandes glAddMember, glDeleteMember (pour les GL closes et gérées) glRekey, glkCompromise, ou glkRefresh et pour retourner une réponse cMCStatusInfoExt pour les demandes respectives. La Figure 12 décrit les interactions de protocole pour envoyer les messages glKey. À la différence des procédures définies pour les messages administratifs, les procédures définies dans cette section DOIVENT être mises en œuvre par les GLA pour les générer et par les membres de la GL à réception. Noter que les messages d'erreur ne sont pas montrés. De plus, le comportement pour les attributs de commande de CMC facultatifs transactionId, senderNonce, et recipientNonce n'est pas traité dans ces procédures.

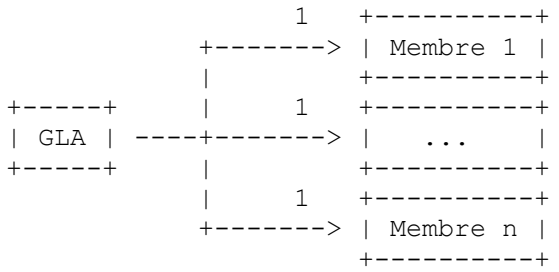


Figure 12 : Distribution de clé de GL

Si la GL a été établie avec GLKeyAttributes.recipientsNotMutuallyAware réglé à VRAI, un message séparé glKey DOIT être envoyé à chaque membre de la GL afin de ne pas divulguer d'informations sur les autres membres de la GL.

Quand le message glKey est généré par suite de :
- une demande glAddMember,
- une indication glkCompromise,
- une demande glkRefresh,
- une demande glDeleteMember avec le glAdministration de la GL réglé à gérée ou close, et
- une demande glRekey avec generationCounter réglé à zéro (0),

le GLA DOIT utiliser le choix kari (voir au paragraphe 12.3.2 de la [RFC3852]) ou ktri (voir au paragraphe 12.3.1 de la [RFC3852]) dans glKey.glkWrapped.RecipientInfo pour assurer que seuls les receveurs prévus reçoivent la KEK partagée.

Le GLA DOIT prendre en charge le choix ktri.

Quand le message glKey est généré par suite d'une demande glRekey avec generationCounter supérieur à zéro (0) ou quand le GLA contrôle les changements de clés, le GLA PEUT utiliser le kari, ktri, ou kekri (voir au paragraphe 12.3.3 de la [RFC3852]) dans glKey.glkWrapped.RecipientInfo pour assurer que seuls les receveurs prévus reçoivent la KEK partagée. Le GLA DOIT prendre en charge le choix RecipientInfo.ktri.

5.1 Processus de distribution

Quand un message glKey est généré, le traitement est le suivant :

1 - Le GLA DOIT envoyer un SignedData.PKIData.controlSequence.glKey à chaque membre en incluant glName, glIdentifier, glkWrapped, glkAlgorithm, glkNotBefore, et glkNotAfter. Si le GLA ne peut pas générer un message glKey pour le membre de la GL parce que le PKC du membre de la GL a expiré ou est autrement invalide, le GLA PEUT envoyer un glUpdateCert au membre de la GL lui demandant la production d'un nouveau certificat (voir au paragraphe 4.10). Le nombre de messages glKey générés pour la GL est décrit au paragraphe 3.1.13. De plus, un attribut signingTime est inclus avec le ou les messages de distribution.

1.a - Le GLA PEUT facultativement appliquer une autre couche de confidentialité au message en encapsulant les SignedData.PKIData dans une autre EnvelopedData (voir au paragraphe 3.2.1.2).

1.b - Le GLA PEUT aussi facultativement appliquer une autre SignedData sur les EnvelopedData.SignedData.PKIData (voir au paragraphe 3.2.1.2).

2 - À réception du message glKey, les membres de la GL DOIVENT vérifier l'heure de signature et vérifier la signature sur les SignedData.PKIData les plus internes. Si un SignedData et/ou EnvelopedData supplémentaire encapsule le message (voir au paragraphe 3.2.1.2 ou 3.2.2) le membre de la GL DOIT vérifier la signature externe et/ou déchiffrer la couche externe avant de vérifier la signature sur la SignedData.PKIData.controlSequence.glKey.

2.a - Si la valeur de l'attribut signingTime n'est pas dans la fenêtre de temps acceptée en local, le GLA PEUT retourner une réponse indiquant cMCStatus.failed et otherInfo.failInfo.badTime et un attribut signingTime.

2.b - Autrement si le traitement de signature se continue et si les signatures ne peuvent pas être vérifiées, le membre de la GL DOIT retourner une réponse cMCStatusInfoExt indiquant cMCStatus.failed et otherInfo.failInfo.badMessageCheck. De plus, un attribut signingTime est inclus dans la réponse.

2.c - Autrement si les signatures se vérifient, le membre de la GL traite les RecipientInfos conformément à la [RFC3852]. Une fois désenveloppée, le membre de la GL devrait mémoriser la KEK partagée dans un endroit sûr. Quand ils sont mémorisés, le glName, glIdentifier, et la KEK partagée devraient être associés. De plus, le membre de la GL DOIT retourner un cMCStatusInfoExt indiquant cMCStatus.success pour dire au GLA que la KEK a été reçue.

6. Algorithmes

Cette Section fait la liste des algorithmes qui DOIVENT être mis en œuvre. Des algorithmes supplémentaires qui DEVRAIENT être mis en œuvre sont aussi inclus. D'autres algorithmes PEUVENT aussi être mis en œuvre.

6.1. Algorithme de génération de KEK

Les mises en œuvre DOIVENT générer au hasard les clés de chiffrement de contenu, les clés d'authentification de message, les valeurs d'initialisation (IV), et le bourrage. Aussi, la génération de paires de clés publique/privée repose sur des nombres aléatoires. L'utilisation de générateurs de nombres pseudo aléatoires inadéquats (PRNG) pour générer des clés de chiffrement peut résulter en peu ou pas de sécurité. Un attaquant peut trouver beaucoup plus facile de reproduire l'environnement du PRNG qui a produit les clés, cherchant dans le petit ensemble résultant de possibilités, plutôt qu'une recherche en force brute sur tout l'espace de clés. La génération de nombres aléatoire de qualité est difficile. La [RFC4086] offre des lignes directrices importantes dans ce domaine, et l'Appendice 3 de la publication FIPS 186 [FIPS] fournit une technique de PRNG de qualité.

6.2 Algorithme d'enveloppement de KEK partagée

Dans les mécanismes décrits à la Section 5, la KEK partagée distribuée dans glkWrapped DOIT être protégée par une clé de force égale ou de longueur supérieure (par exemple, si une clé AES de 128 bits est distribuée, une clé de 128 bits ou plus doit être utilisée pour protéger la clé).

Les identifiants d'objet d'algorithme inclus dans glkWrapped sont comme spécifié dans les [RFC3370] et [RFC3565].

6.3 Algorithme de KEK partagée

La KEK partagée distribuée et indiquée dans glkAlgorithm DOIT prendre en charge les algorithmes de chiffrement de clé symétrique comme spécifié dans les [RFC3370] et [RFC3565].

7. Transport de message

SMTP [RFC2821] DOIT être pris en charge. D'autres mécanismes de transport PEUVENT aussi être pris en charge.

8. Considérations sur la sécurité

Comme les GLO contrôlent l'établissement et la suppression de GL et le changement de clé de la GL, et peuvent contrôler les ajouts et suppressions de membres, les GLO jouent un rôle important dans la gestion de la GL, et seuls des GLO "de confiance" devraient être utilisés.

Si un membre est supprimé ou retiré d'une GL close ou gérée, la clé de la GL a besoin d'être changée. Si la clé de la GL n'est pas changée après qu'un membre est retiré ou supprimé, le membre possède toujours la clé de groupe et va être capable de continuer à déchiffrer tous les messages qui peuvent être obtenus.

Les membres qui mémorisent les KEK DOIVENT associer le nom du GLA qui distribue la clé afin que les membres puissent s'assurer que les changements de clé suivants sont générés par la même entité.

Quand on génère des clés, on devrait s'assurer que la taille de la clé est suffisante et que sa durée n'est pas trop longue parce que des attaquants auront plus de temps pour attaquer la clé. La taille de clé devrait être choisie pour protéger adéquatement les communications d'affaire sensibles.

Les GLO et GLA doivent s'assurer que le compteur de génération et la durée ne sont pas trop grands. Par exemple, si le GLO indique que le generationCounter est 14 et que duration est un an, alors 14 clés sont générées chacune avec une période de validité d'un an. Un attaquant aura au moins 13 ans pour attaquer la clé finale.

Supposons que deux parties ou plus aient une KEK partagée, et que la KEK partagée soit utilisée pour chiffrer une seconde KEK pour la distribution confidentielle à ces parties. La seconde KEK pourrait être utilisée pour chiffrer une troisième KEK, la troisième KEK pourrait être utilisée pour chiffrer une quatrième KEK, et ainsi de suite. Si une des KEK dans une telle chaîne est compromise, toutes les KEK suivantes dans la chaîne DOIVENT aussi être considérées comme compromises.

Un attaquant peut attaquer la KEK partagée du groupe en attaquant la copie de la KEK partagée d'un membre ou en attaquant les copies de la KEK partagée de plusieurs membres. Pour l'attaquant, il peut être plus facile d'attaquer le membre du groupe qui a la plus faible sécurité pour protéger sa copie de la KEK partagée ou bien d'attaquer plusieurs membres du groupe.

Une agrégation des informations rassemblées durant les attaques peut conduire à la compromission de la KEK partagée du groupe. Les mécanismes pour protéger la KEK partagée devraient être proportionnels à la valeur des données protégées.

Les attributs nonce *(nom occasionnel)* et signingTime *(heure de signature)* sont utilisés pour protéger contre les attaques de répétition. Cependant, ces dispositions ne sont utiles que si les entités conservent des informations d'état sur les messages qu'ils ont envoyé et reçu pour comparaison. Si des informations suffisantes ne sont pas conservées sur chaque échange, les noms occasionnels et l'heure de signature ne sont pas utiles. La politique locale détermine la quantité et la durée des informations d'état qui sont conservées. De plus, sans une source horaire unifiée, il y a une possibilité de dérive des

horloges. La politique locale détermine la différence acceptable entre l'heure locale et l'heure de signature, qui doit compenser la non synchronisation des horloges. Les mises en œuvre DOIVENT traiter les messages avec des attributs siginingTime qui indiquent qu'il ont été créés dans le futur.

9. Remerciements

Merci à Russ Housley et Jim Schaad qui ont fourni les bases de ce document et l'ont relu.

10. Références

10.1 Références normatives

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

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

[RFC2821] J. Klensin, éditeur, "Protocole simple de transfert de messagerie", STD 10, avril 2001. *(Obsolète, voir* RFC5321)

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

[RFC3370] R. Housley, "Algorithmes de syntaxe de message cryptographique (CMS)", août 2002. *(P.S. ; MàJ par* RFC8702*)*

[RFC3565] J. Schaad, "Utilisation de l'algorithme de chiffrement de la norme de chiffrement évolué (AES) dans la syntaxe de message cryptographique (CMS)", juillet 2003. *(P.S.)*

[RFC3851] B. Ramsdell, "Spécification du message d'extensions de messagerie Internet multi-objets/sécurisé (S/MIME) version 3.1", juillet 2004. *(Obsolète, voir* RFC5751)

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

[RFC5272] J. Schaad, M. Myers, "Gestion de certificat sur CMS (CMC)", juin 2008. (*Remplace* RFC2797) *(P.S.)*

[RFC5280] D. Cooper et autres, "Profil de certificat d'infrastructure de clé publique X.509 et de liste de révocation de certificat (CRL) pour l'Internet", mai 2008. (*Remplace les* RFC3280, RFC4325, RFC4630) *(P.S. ; MàJ par* RFC8398, 8399)

10.2 Références pour information

[FIPS] National Institute of Standards and Technology, FIPS Pub 186-2, "Digital Signature Standard", janvier 2000.

[RFC3855] P. Hoffman, C. Bonatti, "Transport des objets d'extensions de messagerie Internet multi-objets/sécurisé (S/MIME) dans X.400", juillet 2004. *(P.S.)*

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

Appendice A. Module ASN.1

SMIMESymmetricKeyDistribution { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0) symkeydist(12) }

```
ÉTIQUETTES IMPLICITES DE DÉFINITIONS  ::=
DÉBUT

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

IMPORTE

-- PKIX Partie 1 - Implicite [RFC5280]
GeneralName
  FROM PKIX1Implicit88 { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-
mod(0) id-pkix1-implicit(19) }

-- PKIX Part 1 - Explicite [RFC5280]
AlgorithmIdentifier, Certificate
  DE PKIX1Explicit88 { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
id-pkix1-explicit(18) }

 -- Syntaxe de message cryptographique [RFC3852]
RecipientInfos, KEKIdentifier, CertificateSet
    DE CryptographicMessageSyntax2004 {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
modules(0) cms-2004(24) }

-- Norme de chiffrement évolué (AES) avec CMS [RFC3565]
id-aes128-wrap
  DE CMSAesRsaesOaep { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0) id-
mod-cms-aes(19) }

-- Profil de certificat d'attribut [RFC3281]
AttributeCertificate DE PKIXAttributeCertificate { iso(1) identified-organization(3) dod(6) internet(1) security(5)
mechanisms(5) pkix(7) id-mod(0) id-mod-attribute-cert(12) };

-- Ceci définit l'arc d'identifiant d'objet de distribution de clé symétrique de GL.

IDENTIFIANT D'OBJET id-skd ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
skd(8) }

-- Ceci définit l'attribut de commande GLUseKEK.

IDENTIFIANT D'OBJET id-skd-glUseKEK ::= { id-skd 1 }

GLUseKEK ::= SEQUENCE {
  glInfo                GLInfo,
  glOwnerInfo           SEQUENCE TAILLE (1..MAX) DE GLOwnerInfo,
  glAdministration   GLAdministration DEFAULT 1,
  glKeyAttributes     GLKeyAttributes FACULTATIF }

GLInfo ::= SEQUENCE {
  glName     GeneralName,
  glAddress  GeneralName }

GLOwnerInfo ::= SEQUENCE {
  glOwnerName      GeneralName,
  glOwnerAddress  GeneralName,
  certificates          Certificates FACULTATIF }

GLAdministration ::= ENTIER {
  non gérée  (0),
  géré          (1),
```

```
  close        (2) }

GLKeyAttributes ::= SEQUENCE {
  rekeyControlledByGLO        [0] BOOLÉEN PAR DÉFAUT FAUX,
  recipientsNotMutuallyAware [1] BOOLÉEN PAR DÉFAUT VRAI,
  duration                    [2] ENTIER PAR DÉFAUT 0,
  generationCounter           [3] ENTIER PAR DÉFAUT 2,
  requestedAlgorithm          [4] AlgorithmIdentifier PAR DÉFAUT { id-aes128-wrap } }

-- Ceci définit l'attribut de commande DeleteGL. Il a le type simple GeneralName.

IDENTIFIANT D'OBJET id-skd-glDelete  ::= { id-skd 2 }

DeleteGL ::= GeneralName

-- Ceci définit l'attribut de commande AddGLMember.

IDENTIFIANT D'OBJET id-skd-glAddMember  ::= { id-skd 3 }

GLAddMember ::= SEQUENCE {
  glName      GeneralName,
  glMember  GLMember }

GLMember ::= SEQUENCE {
  glMemberName      GeneralName,
  glMemberAddress   GeneralName FACULTATIF,
  certificates      Certificates     FACULTATIF }

Certificates ::= SEQUENCE {
  pKC       [0] Certificate FACULTATIF,                          -- voir la [RFC5280]
  aC        [1] SEQUENCE TAILLE (1.. MAX) DE AttributeCertificate FACULTATIF,        -- voir la [RFC3281]
  certPath  [2] CertificateSet FACULTATIF }                      -- de la [RFC3852]

-- Ceci définit l'attribut de commande DeleteGLMember.

IDENTIFIANT D'OBJET id-skd-glDeleteMember  ::= { id-skd 4 }

GLDeleteMember ::= SEQUENCE {
  glName          GeneralName,
  glMemberToDelete  GeneralName }

-- Ceci définit l'attribut de commande DeleteGLMember.

IDENTIFIANT D'OBJET id-skd-glRekey  ::= { id-skd 5 }

GLRekey ::= SEQUENCE {
  glName            GeneralName,
  glAdministration      GLAdministration      FACULTATIF,
  glNewKeyAttributes  GLNewKeyAttributes FACULTATIF,
  glRekeyAllGLKeys    BOOLÉEN             FACULTATIF }

GLNewKeyAttributes ::= SEQUENCE {
  rekeyControlledByGLO        [0] BOOLEAN FACULTATIF,
  recipientsNotMutuallyAware  [1] BOOLEAN FACULTATIF,
  duration                    [2] ENTIER FACULTATIF,
  generationCounter           [3] ENTIER FACULTATIF,
  requestedAlgorithm          [4] AlgorithmIdentifier FACULTATIF }

-- Ceci définit les attributs de commande Add et Delete GL Owner.

IDENTIFIANT D'OBJET id-skd-glAddOwner  ::= { id-skd 6 }
```

```
IDENTIFIANT D'OBJET id-skd-glRemoveOwner  ::= { id-skd 7 }

GLOwnerAdministration ::= SEQUENCE {
  glName         GeneralName,
  glOwnerInfo  GLOwnerInfo }

-- Ceci définit l'attribut de commande GLKeyCompromise. Il a le type simple GeneralName.

IDENTIFIANT D'OBJET id-skd-glKeyCompromise ::= { id-skd 8 }

GLKCompromise ::= GeneralName

-- Ceci définit l'attribut de commande GLKeyRefresh.

IDENTIFIANT D'OBJET id-skd-glkRefresh  ::= { id-skd 9 }

GLKRefresh ::= SEQUENCE {
   glName  GeneralName,
   dates     SEQUENCE TAILLE (1..MAX) DE Date }

Date ::= SEQUENCE {
  start   GeneralizedTime,
  end    GeneralizedTime FACULTATIF }

-- Ceci définit l'attribut de commande GLAQueryRequest.

IDENTIFIANT D'OBJET id-skd-glaQueryRequest  ::= { id-skd 11 }

GLAQueryRequest ::= SEQUENCE {
  glaRequestType   IDENTIFIANT D'OBJET,
  glaRequestValue  ANY DÉFINI PAR glaRequestType }

-- Ceci définit l'attribut de commande GLAQueryResponse.

IDENTIFIANT D'OBJET id-skd-glaQueryResponse  ::= { id-skd 12 }

GLAQueryResponse ::= SEQUENCE {
  glaResponseType   IDENTIFIANT D'OBJET,
  glaResponseValue  ANY DÉFINI PAR glaResponseType }

-- Ceci définit l'arc GLA Request/Response (glaRR) pour glaRequestType/glaResponseType.

IDENTIFIANT D'OBJET id-cmc-glaRR  ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5)
                                        mechanisms(5) pkix(7) cmc(7) glaRR(99) }

-- Ceci définit la demande d'algorithme.

IDENTIFIANT D'OBJET id-cmc-gla-skdAlgRequest  ::= { id-cmc-glaRR 1 }

SKDAlgRequest ::= NULL

-- Ceci définit la réponse d'algorithme.

IDENTIFIANT D'OBJET id-cmc-gla-skdAlgResponse  ::= { id-cmc-glaRR 2 }

-- Noter que la réponse pour la demande algorithmSupported est l'attribut smimeCapabilities comme défini dans MsgSpec
[RFC3851]. Ceci définit l'attribut de commande pour demander un certificat mis à jour au GLA.

IDENTIFIANT D'OBJET id-skd-glProvideCert ::= { id-skd 13 }

GLManageCert ::= SEQUENCE {
```

```
  glName      GeneralName,
  glMember  GLMember }

-- Ceci définit l'attribut de commande pour retourner un certificat mis à jour au GLA. Il a le type GLManageCert.

IDENTIFIANT D'OBJET id-skd-glManageCert ::= { id-skd 14 }

-- Ceci définit l'attribut de commande pour distribuer la KEK partagée de GL.

IDENTIFIANT D'OBJET id-skd-glKey ::= { id-skd 15 }

GLKey ::= SEQUENCE {
  glName          GeneralName,
  glIdentifier     KEKIdentifier,                    -- voir la [RFC3852]
   glkWrapped   RecipientInfos,                      -- voir la [RFC3852]
  glkAlgorithm  AlgorithmIdentifier,
  glkNotBefore  GeneralizedTime,
  glkNotAfter     GeneralizedTime }

-- Ceci définit les types d'erreur de CMC.

IDENTIFIANT D'OBJET id-cet-skdFailInfo  ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5)
                                              mechanisms(5) pkix(7) cet(15) skdFailInfo(1) }

SKDFailInfo ::= ENTIER {
  unspecified                (0),
  closedGL                   (1),
  unsupportedDuration        (2),
  noGLACertificate           (3),
  invalidCert                (4),
  unsupportedAlgorithm       (5),
  noGLONameMatch             (6),
  invalidGLName              (7),
  nameAlreadyInUse           (8),
  noSpam                     (9),
-- obsolète                  (10),
  alreadyAMember             (11)
  notAMember                 (12),
  alreadyAnOwner             (13),
  notAnOwner                 (14) }

FIN -- SMIMESymmetricKeyDistribution
```

Adresse de l'auteur

Sean Turner
IECA, Inc.
3057 Nutley Street, Suite 106
Fairfax, VA 22031
USA
mél : turners@ieca.com

Déclaration complète de droits de reproduction

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