Groupe de travail Réseau
Request for Comments : 3414
STD : 62
RFC rendue obsolète : 2574
Catégorie : Norme

U. Blumenthal et B. Wijnen
Lucent Technologies
décembre 2002

Traduction Claude Brière de L'Isle

Modèle de sécurité fondée sur l'utilisateur (USM) pour la version 3 du protocole simple de gestion de réseau (SNMPv3)

Résumé
Le présent document décrit le modèle de sécurité fondé sur l'utilisateur (USM, *modèle de sécurité fondé sur l'utilisateur*) pour le protocole simple de gestion de réseau (SNMP, *Simple Network Management Protocol*) version 3 à utiliser dans l'architecture SNMP. Il définit les éléments de procédure pour fournir à SNMP la sécurité au niveau du message. Le présent document comporte aussi une base de données d'informations de gestion (MIB, *Management Information Base*) pour surveiller/gérer à distance les paramètres de configuration pour ce modèle de sécurité. Le présent document rend obsolète la RFC 2574.

Table des Matières

1. Introduction

L'architecture de description des cadres de gestion de l'Internet [RFC3411] précise qu'un moteur SNMP se compose de :
1) un répartiteur,
2) un sous système de traitement de message,
3) un sous système de sécurité,
4) un sous système de contrôle d'accès.

Les applications utilisent les services de ces sous systèmes.

Il est important de comprendre l'architecture SNMP et la terminologie de l'architecture pour comprendre comment le modèle de sécurité décrit dans le présent document s'ajuste dans l'architecture et interagit avec les autres sous systèmes au sein de l'architecture. Le lecteur est supposé avoir lu et compris la description de l'architecture SNMP, telle que définie dans la [RFC3411].

Le présent mémoire décrit le modèle de sécurité fondé sur l'utilisateur comme il est utilisé au sein de l'architecture SNMP. L'idée principale est qu'on utilise le concept traditionnel d'utilisateur (identifié par un userName) auquel sont associées les informations de sécurité.

Le présent mémoire décrit l'utilisation de HMAC-MD5-96 et de HMAC-SHA-96 comme protocoles d'authentification et l'utilisation de CBC-DES comme protocole de confidentialité. Le modèle de sécurité fondé sur l'utilisateur permet cependant d'utiliser d'autres protocoles à la place ou concurremment avec ces protocoles. Donc, la description de HMAC-MD5-96, HMAC-SHA-96 et CBC-DES est dans des paragraphes séparés pour refléter leur nature autonome et indiquer qu'ils pourront être remplacés ou complétés à l'avenir.

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

1.1 Menaces

Plusieurs des menaces classiques qui pèsent sur les protocoles réseau sont applicables au problème de la gestion de réseau et seront donc applicables à tout modèle de sécurité SNMP. D'autres menaces ne sont pas applicables au problème de la gestion de réseau. Ce paragraphe discute des principales menaces, des menaces secondaires, et des menaces qui sont de moindre importance.

Les principales menaces contre lesquelles le modèle de sécurité SNMP devrait assurer la protection sont :
- La modification des informations : la menace de modification est le danger qu'une entité non autorisée puisse altérer dans le transit les messages SNMP générés au nom d'un principal autorisé de manière à réaliser des opérations de

gestion non autorisées, incluant de falsifier la valeur d'un objet.
- L'usurpation d'identité *(masquerade)* : la menace d'usurpation d'identité est le danger que des opérations de gestion non autorisées à un certain utilisateur puissent être tentées en utilisant l'identité d'un autre usager qui a les autorisations appropriées.

Deux menaces secondaires sont aussi identifiées. Le modèle de sécurité défini dans le présent mémoire fournit une protection limitée contre :
- La divulgation : la menace de divulgation est le danger d'espionnage des échanges entre les agents gérés et une station de gestion. La protection contre cette menace peut être requise au titre de la politique locale.
- La modification du flux de messages : le protocole SNMP se fonde normalement sur un service de transport sans connexion qui peut fonctionner sur tout service de sous réseau. Le réarrangement, le retard ou la répétition des messages peuvent survenir et surviennent effectivement par le fonctionnement naturel de beaucoup de ces services de sous réseau. La menace de modification du flux de messages est le danger que les messages puissent être réarrangés, retardés ou répétés dans un but malveillant dans une mesure bien plus grande que ce que peut produire le fonctionnement naturel d'un service de sous réseau, afin de réaliser des opérations de gestion non autorisées.

Il y a au moins deux menaces contre lesquelles un modèle de sécurité SNMP n'a pas besoin de protéger. Les protocoles de sécurité définis dans le présent mémoire ne fournissent pas de protection contre :
- Le déni de service : ce modèle de sécurité SNMP ne cherche pas à traiter la large gamme d'attaques par lesquelles sont refusés les services aux utilisateurs autorisés. Bien sûr, de telles attaques de déni de service sont dans de nombreux cas indistinguables des types de défaillance du réseau avec lesquels tout protocole de gestion de réseau viable doit normalement avoir à faire.
- L'analyse de trafic : le présent modèle de sécurité SNMP ne cherche pas à régler le problème des attaques d'analyse de trafic. Bien sûr, de nombreux schémas de trafic sont prévisibles – les appareils peuvent être gérés de façon régulière par un nombre relativement faible d'applications de gestion - et donc, il n'y a pas d'avantage significatif à se protéger contre l'analyse de trafic.

1.2 Buts et contraintes

Sur la base du tableau précédent des menaces dans l'environnement de gestion de réseau SNMP, les buts de ce modèle de sécurité SNMP sont les suivants.

1) Assurer la vérification que chaque message SNMP reçu n'a pas été modifié durant sa transmission à travers le réseau.

2) Assurer la vérification de l'identité de l'utilisateur au nom duquel un message SNMP reçu prétend avoir été généré.

3) Assurer la détection des messages SNMP reçus, qui demandent ou contiennent des informations de gestion, dont l'heure de génération n'est pas récente.

4) S'assurer, lorsque nécessaire, que le contenu de chaque message SNMP reçu est protégé de la divulgation.

En plus du but principal de prendre en charge la gestion sûre du réseau, la conception de ce modèle de sécurité SNMP est aussi influencée par les contraintes suivantes :

1) Lorsque les exigences de gestion efficace dans les périodes de tension du réseau sont incompatibles avec celles de la sécurité, la conception de l'USM a donné la préférence aux premières.

2) Ni le protocole de sécurité ni ses mécanismes de sécurité sous-jacents ne devraient dépendre de la disponibilité d'autres services réseau (par exemple, du protocole de l'heure du réseau (NTP, *Network Time Protocol*) ou de protocoles de gestion de clés).

3) Un mécanisme de sécurité ne devrait entraîner aucun changement à la philosophie de base de la gestion de réseau SNMP.

1.3 Services de sécurité

Les services de sécurité nécessaires pour prendre en charge les buts de ce modèle de sécurité SNMP sont les suivants :

- L'intégrité des données est la fourniture de la propriété que les données n'ont pas été altérées ou détruites d'une façon non autorisée, ni que des séquences de données ont été altérées dans une mesure supérieure à ce qui peut arriver de façon accidentelle.

- L'authentification de l'origine des données est la fourniture de la propriété que l'identité revendiquée par l'utilisateur au nom duquel les données reçues ont été générées est corroborée.

- La confidentialité des données est la fourniture de la propriété que les informations ne sont pas mises à la disposition ou divulguées à des individus, entités, ou processus non autorisés.

- L'opportunité d'un message et la protection limitée contre la répétition sont la fourniture de la propriété qu'un message dont l'heure de génération est en dehors d'une fenêtre temporelle spécifiée ne soit pas accepté. Noter que le réarrangement de message n'est pas traité et peut aussi survenir dans des conditions normales.

Pour les protocoles spécifiés dans le présent mémoire, il n'est pas possible d'assurer l'origine spécifique d'un message SNMP reçu ; c'est plutôt l'utilisateur au nom duquel le message a été généré qui est authentifié.

Pour ces protocoles, il n'est pas possible d'obtenir l'intégrité des données sans l'authentification de l'origine des données, ni d'obtenir l'authentification de l'origine des données sans l'intégrité des données. De plus, il n'y a aucune disposition pour la confidentialité des données sans l'intégrité des données et l'authentification de l'origine des données.

Les protocoles de sécurité utilisés dans le présent mémoire sont considérés comme d'une sécurité acceptable au moment de sa rédaction. Cependant, les procédures permettent que de nouvelles méthodes d'authentification et de confidentialité soient spécifiée à l'avenir si le besoin s'en fait sentir.

1.4 Organisation des modules

Les protocoles de sécurité définis dans le présent mémoire se partagent en trois modules différents et chacun a ses responsabilités spécifiques de façon qu'ensemble ils réalisent les buts et services de sécurité décrits ci dessus :
- Le module d'authentification DOIT assurer :
  - l'intégrité des données
  - l'authentification de l'origine des données,
- Le module d'opportunité DOIT assurer :
  - la protection contre le retard ou la répétition du message (dans une mesure supérieure à ce qui peut arriver en fonctionnement normal
- Le module de confidentialité DOIT assurer :
  - la protection contre la divulgation de la charge utile du message.

Le module d'opportunité est fixe pour le modèle de sécurité fondé sur l'utilisateur alors qu'il y a des dispositions pour plusieurs modules d'authentification et/ou de confidentialité, dont chacun met en œuvre un protocole spécifique, respectivement d'authentification ou de confidentialité.

1.4.1 Module d'opportunité

La section 3 (Éléments de procédure) utilise les valeurs d'opportunité dans un message SNMP pour vérifier l'opportunité. La vérification d'opportunité n'est effectuée que si l'authentification est appliquée au message. Comme l'intégrité du message complet est vérifiée, on peut supposer que les valeurs d'opportunité dans un message qui réussit au module d'authentification sont dignes de confiance.

1.4.2 Protocole d'authentification

La Section 6 décrit le protocole d'authentification HMAC-MD5-96 qui est le premier protocole d'authentification qui DOIT être pris en charge avec le modèle de sécurité fondé sur l'utilisateur. La Section 7 décrit le protocole d'authentification HMAC-SHA-96 qui est un autre protocole d'authentification qui DEVRAIT être pris en charge avec le modèle de sécurité fondé sur l'utilisateur. À l'avenir, des protocoles d'authentification supplémentaires ou de remplacement pourront être définis en tant que de besoin.

Le modèle de sécurité fondé sur l'utilisateur prescrit que, si l'authentification est utilisée, le message complet est alors vérifié en intégrité dans le module d'authentification.

Pour qu'un message soit authentifié, il doit réussir aux vérifications d'authentification du module d'authentification et aux vérifications d'opportunité qui sont une partie fixe de ce modèle de sécurité fondé sur l'utilisateur.

1.4.3 Protocole de confidentialité

La Section 8 décrit le protocole de chiffrement symétrique CBC-DES qui est le premier protocole de confidentialité à être utilisé avec le modèle de sécurité fondé sur l'utilisateur. À l'avenir, des protocoles de confidentialité supplémentaires ou de remplacement pourront être définis en tant que de besoin.

Le modèle de sécurité fondé sur l'utilisateur prescrit que la scopedPDU soit protégée de la divulgation lorsque un message est envoyé de façon confidentielle.

Le modèle de sécurité fondé sur l'utilisateur prescrit aussi qu'un message soit authentifié si la confidentialité est utilisée.

1.5 Protection contre la répétition, le retard et la redirection de message

1.5.1 Moteur SNMP d'autorité

Afin de protéger contre la répétition, le retard et la redirection de message, un des moteurs SNMP impliqués dans chaque communication est désigné comme moteur SNMP d'autorité. Lorsque un message SNMP contient une charge utile qui appelle une réponse (les messages qui contiennent une PDU de classe Confirmée [RFC3411]) le receveur de tels messages est d'autorité. Lorsque un message SNMP contient une charge utile qui n'appelle pas de réponse (les messages qui contiennent une PDU de classe Non confirmée [RFC3411]) l'envoyeur d'un tel message est d'autorité.

1.5.2 Mécanismes

Les mécanismes suivants sont utilisés :

1) Pour protéger contre la menace de retard ou de répétition de message (dans une mesure supérieure à ce qui peut arriver en fonctionnement normal) un ensemble d'indicateurs d'opportunité (pour le moteur SNMP d'autorité) est inclus dans chaque message généré. Un moteur SNMP évalue les indicateurs d'opportunité pour déterminer si un message reçu est récent. Un moteur SNMP peut évaluer les indicateurs d'opportunité pour s'assurer qu'un message reçu est au moins aussi récent que le dernier message qu'il a reçu de la même source. Un moteur SNMP non d'autorité utilise les messages reçus authentiques pour améliorer sa compréhension des indicateurs d'opportunité à la source d'autorité distante.
   Un moteur SNMP DOIT aussi utiliser un mécanisme pour confronter les réponses entrantes aux demandes en cours et il DOIT éliminer toute réponse qui ne correspond pas à une demande en cours. Par exemple, un msgID *(identifiant de message)* peut être inséré dans tout message pour satisfaire cette fonctionnalité.
   Ces mécanismes assurent la détection des messages authentifiés dont l'heure de génération n'est pas récente.
   Cette protection contre la menace de retard ou répétition de message n'implique pas ni ne fournit de protection contre la suppression non autorisée ou l'élimination de messages. Aussi, un moteur SNMP peut n'être pas capable de détecter un réarrangement de messages si tous les messages impliqués sont envoyés dans l'intervalle de la fenêtre temporelle. D'autres mécanismes définis indépendamment du protocole de sécurité peuvent aussi être utilisés pour détecter le réarrangement, la répétition, l'élimination, ou la suppression de messages contenant des opérations SET (par exemple, la variable de MIB snmpSetSerialNo [RFC3418]).

2) La vérification qu'un message envoyé à/d'un moteur SNMP d'autorité ne peut pas être répété à/comme si il venait d'un autre moteur SNMP d'autorité. Dans chaque message est inclus un identifiant unique au moteur SNMP d'autorité associé à l'envoyeur ou au receveur prévu du message.
   Un message contenant une PDU de classe Non confirmée envoyé par un moteur SNMP d'autorité à un moteur SNMP non d'autorité peut éventuellement être répété à un autre moteur SNMP non d'autorité. Le dernier moteur SNMP non d'autorité pourrait (si il connaît le même userName avec les mêmes secrets que le moteur SNMP d'autorité) ainsi mettre à jour sa compréhension des indicateurs d'opportunité du moteur SNMP d'autorité, mais ceci n'est pas considéré comme une menace. Dans ce cas, un message Report ou Response sera éliminé par le modèle de traitement de messages, parce que il ne devrait pas être un message de demande en cours. Un message Trap serait éventuellement accepté. Là encore, ceci n'est pas considéré comme une menace, parce que la communication est authentifiée et à propos. C'est comme si le moteur SNMP d'autorité était configuré à commencer par envoyer des Traps au second moteur SNMP, ce qui peut théoriquement arriver sans que le second moteur SNMP le sache. De toutes façons, le second moteur SNMP peut ne pas s'attendre à recevoir de Trap, mais il lui est permis de voir les informations de gestion qu'il contient.

3) Détection de messages qui n'ont pas été générés récemment.
   Un ensemble d'indicateurs horaires est inclus dans le message, indiquant l'heure de génération. Les messages sans indicateurs horaires récents ne sont pas considérés comme authentiques. De plus, un moteur SNMP DOIT éliminer toute réponse qui ne correspond pas à une demande en cours. Ceci est cependant de la responsabilité du modèle de traitement de message. Le présent mémoire permet que le même utilisateur soit défini sur plusieurs moteurs SNMP. Chaque moteur SNMP conserve une valeur, snmpEngineID, qui identifie de façon univoque le moteur SNMP. Cette valeur est

incluse dans chaque message envoyé au/du moteur SNMP qui est d'autorité (voir au paragraphe 1.5.1). À réception d'un message, un moteur SNMP d'autorité vérifie la valeur pour s'assurer qu'il est le receveur prévu, et un moteur SNMP non d'autorité utilise la valeur pour s'assurer que le message est traité en utilisant les informations d'état correctes.
Chaque moteur SNMP tient deux valeurs, snmpEngineBoots et snmpEngineTime, qui prises ensemble fournissent l'indication de l'heure à ce moteur SNMP. Ces deux valeurs sont incluses dans un message authentifié envoyé à/reçu de ce moteur SNMP. À réception, les valeurs sont vérifiées pour s'assurer que la valeur d'opportunité indiquée est dans la fenêtre horaire de l'heure actuelle. La fenêtre horaire représente une limite supérieure administrative sur le délai de livraison acceptable pour les messages de protocole.
Pour qu'un moteur SNMP génère un message qu'acceptera un moteur SNMP d'autorité comme authentique, et pour vérifier qu'un message reçu de ce moteur SNMP d'autorité est authentique, un tel moteur SNMP doit d'abord réaliser la synchronisation d'opportunité avec le moteur SNMP d'autorité. Voir au paragraphe 2.3.

1.6 Interfaces de service abstraites

Les interfaces de service abstraites ont été définies pour décrire les interfaces conceptuelles entre les divers sous systèmes au sein d'une entité SNMP. De même, un ensemble d'interfaces de service abstraites a été défini au sein du modèle de sécurité fondé sur l'utilisateur (USM) pour décrire les interfaces conceptuelles entre les services USM génériques et les services auto contenus d'authentification et de confidentialité.

Ces interface de service abstraites sont définies par un ensemble de primitives qui définissent les services fournis et les éléments de données abstraits qui doivent être passés lorsque les services sont invoqués. Ce paragraphe fait la liste des primitives qui ont été définies pour le modèle de sécurité fondé sur l'utilisateur.

1.6.1 Primitives du modèle de sécurité fondé sur l'usager pour l'authentification

Le modèle de sécurité fondé sur l'utilisateur fournit les primitives internes suivantes pour passer les données entre le modèle de sécurité lui-même et le service d'authentification :

```
statusInformation = authenticateOutgoingMsg(
  IN  authKey                          -- clé secrète pour l'authentification
  IN  wholeMsg                         -- message complet non authentifié
  OUT  authenticatedWholeMsg           -- message authentifié complet
  )

statusInformation =  authenticateIncomingMsg(
   IN   authKey                         -- clé secrète pour l'authentification
   IN   authParameters                  -- comme reçu du réseau
   IN   wholeMsg                        -- comme reçu du réseau
   OUT  authenticatedWholeMsg           -- message authentifié complet
    )
```

1.6.2 Primitives du modèle de sécurité fondé sur l'usager pour la confidentialité

Le modèle de sécurité fondé sur l'utilisateur fournit les primitives internes suivantes pour passer les données entre le modèle de sécurité lui-même et le service de confidentialité :

```
statusInformation =   encryptData(
   IN    encryptKey                -- clé secrète pour chiffrement
   IN    dataToEncrypt             -- données à chiffrer (scopedPDU)
   OUT   encryptedDat              -- données chiffrées (encryptedPDU)
   OUT   privParameters            -- rempli par le fournisseur de
                                   service
    )

statusInformation = decryptData(
   IN    decryptKey                -- clé secrète pour déchiffrement
   IN    privParameters            -- comme reçu du réseau
   IN    encryptedData             -- données chiffrées (encryptedPDU)
   OUT   decryptedData             -- données déchiffrées (scopedPDU)
    )
```

2. Éléments du modèle

Cette Section contient les définitions requises pour réaliser le modèle de sécurité défini par le présent mémoire.

2.1 Utilisateurs du modèle de sécurité fondé sur l'usager

Les opérations de gestion qui utilisent ce modèle de sécurité se servent d'un ensemble défini d'identités d'utilisateur. Pour tout utilisateur au nom duquel des opérations de gestion sont autorisées à un moteur SNMP particulier, ce moteur SNMP doit avoir connaissance de cet utilisateur. Un moteur SNMP qui souhaite communiquer avec un autre moteur SNMP doit aussi avoir connaissance d'un utilisateur connu de ce moteur, incluant la connaissance des attributs applicables de cet utilisateur.

Un utilisateur et ses attributs sont définis comme suit :

userName : chaîne qui représente le nom de l'usager.

securityName : chaîne lisible par l'homme qui représente l'usager dans un format indépendant du modèle de sécurité. Il y a une relation biunivoque entre userName et securityName.

authProtocol : indication de ce que les messages envoyés au nom de cet usager peuvent être ou non authentifiés, et si oui, le type de protocole d'authentification utilisé. Deux de ces protocoles sont définis dans ce mémoire :
- le protocole d'authentification HMAC-MD5-96,
- le protocole d'authentification HMAC-SHA-96.

authKey : si les messages envoyés au nom de cet usager peuvent être authentifiés, c'est la clé (privée) d'authentification à utiliser avec le protocole d'authentification. Noter que la clé d'authentification d'un utilisateur va normalement être différente sur les différents moteurs SNMP d'autorité. La authKey n'est pas accessible via SNMP. Les exigences de longueur de authKey sont définies par le authProtocol en usage.

authKeyChange et authOwnKeyChange : c'est la seule façon de mettre à jour à distance la clé d'authentification. Cela le fait de façon sûre, de sorte que la mise à jour peut se faire sans qu'il soit besoin d'employer une protection de confidentialité.

privProtocol : indique si les messages envoyés au nom de cet usager peuvent être protégés de la divulgation, et si oui, le type de protocole de confidentialité utilisé. Un de ces protocoles est défini dans le présent mémoire : le protocole de chiffrement symétrique CBC-DES.

privKey : si les messages envoyés au nom de cet usager peuvent être chiffrés/déchiffrés, c'est la clé (privée) de confidentialité à utiliser avec le protocole de confidentialité. Noter que la clé de confidentialité d'un utilisateur va normalement être différente dans les différents moteurs SNMP d'autorité. La privKey n'est pas accessible via SNMP. Les exigences de longueur de privKey sont définies par le privProtocol en usage.

privKeyChange et privOwnKeyChange : c'est la seule façon de mettre à jour à distance la clé de chiffrement. Cela le fait de façon sûre, afin que la mise à jour puisse être faite sans qu'il soit besoin d'employer de protection de la confidentialité.

2.2 Protection contre la répétition

Chaque moteur SNMP conserve trois objets :
- snmpEngineID, qui (au moins au sein d'un domaine administratif) identifie de façon univoque et sans ambiguïté un moteur SNMP ;
- snmpEngineBoots, qui est un compte du nombre de fois que le moteur SNMP a réamorcé/réinitialisé depuis la dernière configuration de snmpEngineID ;
- snmpEngineTime, qui est le nombre de secondes depuis la dernière incrémentation du compteur snmpEngineBoots.

Chaque moteur SNMP est toujours d'autorité à l'égard de ces objets dans sa propre entité SNMP. Il est de la responsabilité d'un moteur SNMP non d'autorité de se synchroniser avec le moteur SNMP d'autorité, lorsque c'est approprié.

Un moteur SNMP d'autorité est obligé de conserver les valeurs de ses snmpEngineID et snmpEngineBoots dans une mémorisation non volatile.

2.2.1 msgAuthoritativeEngineID

La valeur msgAuthoritativeEngineID contenue dans un message authentifié est utilisée pour déjouer les attaques dans lesquelles les messages d'un moteur SNMP à un autre moteur SNMP sont répétés à un moteur SNMP différent. Elle représente le snmpEngineID au moteur SNMP d'autorité impliqué dans l'échange de message.

Lorsque un moteur SNMP d'autorité est installé, il règle sa valeur locale de snmpEngineID conformément à un algorithme spécifique de l'entreprise (voir la définition de la convention textuelle pour SnmpEngineID dans le document d'architecture SNMP [RFC3411]).

2.2.2 msgAuthoritativeEngineBoots et msgAuthoritativeEngineTime

Les valeurs msgAuthoritativeEngineBoots et msgAuthoritativeEngineTime contenues dans un message authentifié sont utilisées pour déjouer les attaques dans lesquelles les messages sont répétés alors qu'ils ne sont plus valides. Elles représentent les valeurs snmpEngineBoots et snmpEngineTime au moteur SNMP d'autorité impliqué dans l'échange de message.

Par l'utilisation de snmpEngineBoots et de snmpEngineTime, il n'est pas exigé d'un moteur SNMP qu'il ait une horloge non volatile qui s'incrémente (c'est-à-dire, s'augmente au passage du temps) même lorsque le moteur SNMP est débranché. Chaque fois qu'un moteur SNMP réamorce, il récupère l'heure, l'incrémente, et ensuite mémorise snmpEngineBoots dans une mémoire non volatile, et remet snmpEngineTime à zéro.

Lorsque un moteur SNMP s'installe, il règle ses valeurs locales de snmpEngineBoots et snmpEngineTime à zéro. Si jamais snmpEngineTime atteint sa valeur maximum (2 147 483 647) snmpEngineBoots est alors incrémenté comme si le moteur SNMP avait réamorcé et snmpEngineTime est remis à zéro et recommence à s'incrémenter.

Chaque fois qu'an moteur SNMP d'autorité réamorce, tous les moteurs SNMP qui détiennent les valeurs de ce moteur SNMP d'autorité de snmpEngineBoots et snmpEngineTime doivent se resynchroniser avant d'envoyer correctement des messages authentifiés à ce moteur SNMP d'autorité (voir le paragraphe 2.3 sur les procédures de (re-)synchronisation). Noter, cependant, que la procédure assure que une notification sera acceptée comme authentique par un moteur SNMP receveur, lorsque elle est envoyée par un moteur SNMP d'autorité qui s'est réamorcé depuis que le moteur SNMP receveur s'est (re-)synchronisé pour la dernière fois.

Si un moteur SNMP d'autorité se trouve incapable de déterminer sa dernière valeur de snmpEngineBoots, il doit alors régler sa valeur de snmpEngineBoots à 2 147 483 647.

Chaque fois que la valeur locale de snmpEngineBoots a la valeur 2 147 483 647, il se bloque à cette valeur et un message authentifié cause toujours un échec d'authentification notInTimeWindow.

Pour réinitialiser un moteur SNMP dont la valeur de snmpEngineBoots a atteint la valeur de 2 147 483 647, une intervention manuelle est nécessaire. Le moteur doit être visité physiquement et reconfiguré, soit avec une nouvelle valeur de snmpEngineID, soit avec une nouvelle valeur secrète pour le protocole d'authentification et le protocole de confidentialité de tous les utilisateurs connus de ce moteur SNMP. Noter que même si un moteur SNMP réamorce une fois par seconde cela lui prendrait quand même approximativement 68 ans avant d'atteindre la valeur maximale de 2 147 483 647.

2.2.3 Fenêtre horaire

La fenêtre horaire est une valeur qui spécifie la fenêtre de temps dans laquelle un message généré au nom de tout utilisateur est valide. Le présent mémoire spécifie que la même valeur de fenêtre horaire, 150 secondes, est utilisée pour tous les usagers.

2.3 Synchronisation

La synchronisation horaire, requise par un moteur SNMP non d'autorité afin de procéder à des communications authentifiées, se produit lorsque le moteur SNMP non d'autorité a obtenu du moteur SNMP d'autorité une notion locale des valeurs du moteur SNMP d'autorité de snmpEngineBoots et snmpEngineTime. Ces valeurs doivent être (et rester) dans la fenêtre horaire du moteur SNMP d'autorité. Ainsi la notion locale des valeurs du moteur SNMP d'autorité doit être conservée lâchement synchronisée avec les valeurs mémorisées au moteur SNMP d'autorité. En plus de garder une copie locale du snmpEngineBoots et snmpEngineTime du moteur SNMP d'autorité, un moteur SNMP non d'autorité doit aussi garder une variable locale, latestReceivedEngineTime. Cette valeur enregistre la plus forte valeur de snmpEngineTime qui a été reçue par le moteur SNMP non d'autorité du moteur SNMP d'autorité et est utilisée pour éliminer la possibilité de

répétition des messages qui empêcheraient d'avancer la notion du snmpEngineTime du moteur SNMP non d'autorité.

Un moteur SNMP non d'autorité doit conserver les notions locales de ces valeurs (snmpEngineBoots, snmpEngineTime et latestReceivedEngineTime) pour chaque moteur SNMP d'autorité avec lequel il souhaite communiquer. Comme chaque moteur SNMP d'autorité est identifié de façon univoque et sans ambiguïté par sa valeur de snmpEngineID, le moteur SNMP non d'autorité peut utiliser cette valeur comme clé afin de mettre en antémémoire ses notions locales de ces valeurs.

La synchronisation horaire survient au titre des procédures de réception d'un message SNMP (paragraphe 3.2, étape 7b). À ce titre aucune procédure explicite de synchronisation temporelle n'est requise par un moteur SNMP non d'autorité. Noter que chaque fois que la valeur locale de snmpEngineID est changée (par exemple, par découverte) ou lorsque des communications sûres sont d'abord établies avec un moteur SNMP d'autorité, les valeurs locales de snmpEngineBoots et latestReceivedEngineTime devraient être réglées à zéro. Cela va causer la production de la synchronisation horaire lorsque le prochain message authentifié sera reçu.

2.4 Messages SNMP qui utilisent ce modèle de sécurité

La syntaxe de message SNMP utilisant le présent modèle de sécurité adhère au format de message défini dans le document de modèle de traitement de message spécifique de la version (par exemple la [RFC3412]).

Le champ msgSecurityParameters dans les messages SNMPv3 a un type de données de CHAINE D'OCTETS. Sa valeur est la mise en série en BER de la séquence ASN.1 suivante :

DÉFINITIONS DES ÉTIQUETTES IMPLICITES DE USMSecurityParametersSyntax ::= DÉBUT

```
UsmSecurityParameters ::=       -- paramètres de sécurité globaux fondés sur l'utilisateur
SEQUENCE {
  msgAuthoritativeEngineID      CHAINE D'OCTETS,
  msgAuthoritativeEngineBoots   ENTIER (0 à 2 147 483 647),
  msgAuthoritativeEngineTime    ENTIER (0 à 2 147 483 647),
  msgUserName                   CHAINE D'OCTETS (TAILLE(0 à 32) -- paramètres spécifiques du protocole
                                d'authentification
  msgAuthenticationParameters   CHAINE D'OCTETS, -- paramètres spécifiques du protocole de confidentialité
  msgPrivacyParameters          CHAINE D'OCTETS
  }
FIN
```

Les champs de cette séquence sont :
- msgAuthoritativeEngineID spécifie le snmpEngineID du moteur SNMP d'autorité impliqué dans l'échange de message.
- msgAuthoritativeEngineBoots spécifie la valeur snmpEngineBoots au moteur SNMP d'autorité impliqué dans l'échange de message.
- msgAuthoritativeEngineTime spécifie la valeur snmpEngineTime au moteur SNMP d'autorité impliqué dans l'échange de message.
- msgUserName spécifie l'usager (principal) au nom duquel le message est échangé. Noter qu'un userName de longueur zéro ne va correspondre à aucun usager, mais il peut être utilisé pour la découverte de snmpEngineID.
- les msgAuthenticationParameters sont définis par le protocole d'authentification utilisé pour le message, comme défini par la colonne usmUserAuthProtocol de l'entrée de l'usager dans usmUserTable.
- les msgPrivacyParameters sont définis par le protocole de confidentialité utilisé pour le message, comme défini par la colonne usmUserPrivProtocol de l'entrée de l'usager dans usmUserTable).

Voir à l'Appendice A.4 un exemple du codage en BER du champ msgSecurityParameters.

2.5 Services fournis par le modèle de sécurité fondé sur l’usager

Ce paragraphe décrit les services fournis par le modèle de sécurité fondé sur l'utilisateur avec leurs entrées et leur résultats.

Les services sont décrits comme des primitives d'une interface de service abstraite et les entrées et résultats sont décrits comme des éléments de données abstraits lorsque ils sont passés dans ces primitives de service abstraites.

2.5.1 Services pour générer un message SNMP sortant

Lorsque le sous système de traitement de message (MP, *Message Processing*) invoque le module de sécurité fondé sur

l'utilisateur pour sécuriser un message SNMP sortant, il doit utiliser le service approprié tel que fourni par le module de sécurité. Ces deux services sont fournis :

1) un service pour générer un message de demande. La primitive de service abstraite est :

```
statusInformation =   generateRequestMsg(   -- succès ou errorIndication
      IN   messageProcessingModel           -- normalement, la version SNMP
      IN   globalData                       -- en-tête de message, données
                                            administratives
      IN   maxMessageSize                   -- de l'entité SNMP d'envoi
      IN   securityModel                    -- pour le message sortant
      IN   securityEngineID                 -- entité SNMP d'autorité
      IN   securityName                     -- au nom de ce principal
      IN   securityLevel                    -- niveau de sécurité demandé
      IN   scopedPDU                        -- charge utile du message (texte source)
      OUT  securityParameters               -- rempli par le module de sécurité
      OUT  wholeMsg                         -- message généré complet
      OUT  wholeMsgLength                   -- longueur du message généré
       )
```

2) un service pour générer un message de réponse. La primitive de service abstraite est :

```
statusInformation =                         -- succès ou errorIndication
generateResponseMsg(
      IN   messageProcessingModel           -- normalement, la version SNMP
      IN   globalData                       -- en-tête de message, données administratives
      IN   maxMessageSize                   -- de l'entité SNMP d'envoi
      IN   securityModel                    -- pour le message sortant
      IN   securityEngineID                 -- entité SNMP d'autorité
      IN   securityName                     -- au nom de ce principal
      IN   securityLevel                    -- niveau de sécurité demandé
      IN   scopedPDU                        -- charge utile du message (texte source)
      IN   securityStateReference           -- référence aux informations d'état de sécurité de la demande
                                            d'origine
      OUT  securityParameters               -- rempli par le module de sécurité
      OUT  wholeMsg                         -- message généré complet
      OUT  wholeMsgLength                   -- longueur du message généré
      )
```

Les éléments de données abstraits passés comme paramètres dans la primitives de service abstraites sont :

statusInformation : indication de si le codage et la sécurisation du message ont réussi. Sinon c'est une indication du problème.

messageProcessingModel : numéro de version SNMP pour le message à générer. Ces données ne sont pas utilisées par le module de sécurité fondé sur l'utilisateur.

globalData : en-tête de message (c'est-à-dire, ses informations administratives). Ces données ne sont pas utilisées par le module de sécurité fondé sur l'utilisateur.

maxMessageSize : taille maximum de message telle qu'incluse dans le message. Ces données ne sont pas utilisées par le module de sécurité fondé sur l'utilisateur.

securityParameters : ce sont les paramètres de sécurité. Ils sont remplis par le module de sécurité fondé sur l'utilisateur.

securityModel : modèle de sécurité utilisé. Devrait être le modèle de sécurité fondé sur l'utilisateur. Ces données ne sont pas utilisées par le module de sécurité fondé sur l'utilisateur.

securityName : avec le snmpEngineID, il identifie une rangée de usmUserTable qui doit être utilisée pour sécuriser le message. Le securityName a un format indépendant du modèle de sécurité. En cas d'une réponse, ce paramètre est ignoré et c'est la valeur de l'antémémoire qui est utilisée.

securityLevel : niveau de sécurité à partir duquel le module de sécurité fondé sur l'utilisateur détermine si le message a

besoin d'être protégé de la divulgation et si le message doit être authentifié.

securityEngineID : c'est le snmpEngineID du moteur SNMP d'autorité auquel un message dateRequest est envoyé. En cas de réponse, c'est implicitement le snmpEngineID du moteur SNMP de traitement et donc si il est spécifié, il est alors ignoré.

scopedPDU : charge utile du message. Les données sont opaques en ce qui concerne le modèle de sécurité fondé sur l'utilisateur.

securityStateReference : bride/référence aux cachedSecurityData à utiliser pour sécuriser un message Response sortant. C'est exactement la même bride/référence que générée par le module de sécurité fondé sur l'utilisateur lors du traitement du message Request entrant auquel ceci est la réponse.

wholeMsg : message codé et sécurisé complet prêt à l'envoi sur le réseau.

wholeMsgLength : longueur du message codé et sécurisé (wholeMsg).

À l'achèvement du processus, le module de sécurité fondé sur l'utilisateur retourne les statusInformation. Si le processus a réussi, le message est retourné, complété avec l'application de la confidentialité et l'authentification si cela avait été demandé par le niveau de sécurité spécifié. Si le processus n'a pas réussi, une indication d'erreur *(errorIndication)* est retournée.

2.5.2 Services pour traiter un message SNMP entrant

Lorsque le sous système de traitement de message invoque le module de sécurité fondé sur l'utilisateur pour vérifier que la sécurité d'un message entrant est appropriée, il doit utiliser le service fourni pour un message entrant. La primitive de service abstraite est :

```
statusInformation =   processIncomingMsg(   -- errorIndication ou succès ; OID/valeur de compteur d'erreur si erreur
    IN   messageProcessingModel             -- normalement, la version SNMP de l'entité SNMP d'envoi
    IN   maxMessageSize                     -- taille maximale du message
    IN   securityParameters                 -- pour le message reçu
    IN   securityModel                      -- pour le message reçu
    IN   securityLevel                      -- niveau de sécurité
    IN   wholeMsg                           -- comme reçu du réseau
    IN   wholeMsgLength                     -- longueur comme reçu du réseau
    OUT  securityEngineID                   -- entité SNMP d'autorité
    OUT  securityName                       -- identification du principal
    OUT  scopedPDU                          -- charge utile du message (texte source)
    OUT  maxSizeResponseScopedPDU           -- taille maximum de la PDU Response
    OUT  securityStateReference             -- référence aux informations d'état de sécurité, nécessaires pour la
                                            réponse
    )
```

Les éléments de données abstraits passés comme paramètres dans la primitives de service abstraites sont :

statusInformation : indication de si le processus a réussi ou non. Si non, les statusInformation incluent alors l'OID et la valeur du compteur d'erreurs qui a été incrémenté.

messageProcessingModel : numéro de version SNMP qui a été reçu dans le message. Ces données ne sont pas utilisées par le module de sécurité fondé sur l'utilisateur.

maxMessageSize : taille maximum de message telle qu'incluse dans le message. Le module de sécurité fondé sur l'utilisateur utilise cette valeur pour calculer la maxSizeResponseScopedPDU.

securityParameters : ce sont les paramètres de sécurité reçus dans le message.

securityModel : c'est le modèle de sécurité utilisé. Devrait être le modèle de sécurité fondé sur l'utilisateur. Ces données ne sont pas utilisées par le module de sécurité fondé sur l'utilisateur.

securityLevel : c'est le niveau de sécurité à partir duquel le module de sécurité fondé sur l'utilisateur détermine si le message a besoin d'être protégé de la divulgation et si le message doit être authentifié.

wholeMsg : message entier tel qu'il a été reçu.

wholeMsgLength : longueur du message tel que reçu (wholeMsg).

securityEngineID : c'est l'identifiant de moteur SNMP (snmpEngineID) qui a été extrait du champ msgAuthoritativeEngineID et qui a été utilisé pour rechercher les secrets dans le usmUserTable.

securityName : c'est le nom de sécurité qui représente l'usager au nom duquel le message a été reçu. Le securityName a un format indépendant du modèle de sécurité.

scopedPDU : charge utile du message. Les données sont opaques en ce qui concerne le module de sécurité fondé sur l'utilisateur.

maxSizeResponseScopedPDU : taille maximum d'une scopedPDU à inclure dans un éventuel message de réponse. Le module de sécurité fondé sur l'utilisateur calcule cette taille sur la base de la msgMaxSize (telle que reçue dans le message) et l'espace requis pour l'en-tête de message (incluant les paramètres de sécurité) pour un tel message de réponse.

securityStateReference : bride/référence aux cachedSecurityData à utiliser pour sécuriser un message de réponse sortant. Lorsque le sous système de traitement de message invoque le module de sécurité fondé sur l'utilisateur pour générer une réponse à ce message entrant, il doit passer cette bride/référence.

À l'achèvement du processus, le module de sécurité fondé sur l'utilisateur retourne les statusInformation et, si le processus a réussi, les éléments de données supplémentaires pour le traitement ultérieur du message. Si le processus n'a pas réussi, il retourne une indication d'erreur, éventuellement avec une paire OID-valeur d'un compteur d'erreur qui a été incrémenté.

2.6 Algorithme de localisation de clé

Une clé localisée est une clé secrète partagée entre un utilisateur U et un moteur SNMP d'autorité E. Bien qu'un utilisateur puisse n'avoir qu'un mot de passe et donc une clé pour tout le réseau, les secrets partagés réels entre l'usager et chaque moteur SNMP d'autorité seront différents. Ceci est réalisé par la localisation de clé [Localized-key].
D'abord, si un utilisateur utilise un mot de passe, celui ci est converti en une clé Ku en utilisant un des deux algorithmes décrits aux Appendices A.2.1 et A.2.2.

Pour convertir la clé Ku en une clé localisée Kul de l'usager U au moteur SNMP d'autorité E, on a joute le snmpEngineID du moteur SNMP d'autorité à la clé Ku et ensuite on ajoute la clé Ku au résultat, enveloppant le snmpEngineID au sein des deux copies de la clé Ku de l'usager. On applique ensuite une fonction sûre de hachage (qui dépend du protocole d'authentification défini pour cet usager U au moteur SNMP d'autorité E ; le présent document définit deux protocoles d'authentification avec leur algorithme associé fondé sur MD5 et SHA). Le résultat de la fonction de hachage est la clé localisée Kul pour l'usager U au moteur SNMP d'autorité E.

3. Éléments de procédure

Cette Section décrit les procédures relatives à la sécurité suivies par un moteur SNMP lors du traitement des messages SNMP conformément au modèle de sécurité fondé sur l'utilisateur.

3.1 Génération d'un message SNMP sortant

Ce paragraphe décrit la procédure suivie par un moteur SNMP chaque fois qu'il génère un message contenant une opération de gestion (comme une demande, une réponse, une notification, ou un rapport) au nom d'un utilisateur, avec un niveau de sécurité particulier.

1) a) Si une securityStateReference est passée (message Response ou Report) les informations concernant l'usager sont alors extraites des cachedSecurityData. Les cachedSecurityData peuvent maintenant être éliminées. Le securityEngineID est réglé au snmpEngineID local. Le securityLevel est réglé à la valeur spécifiée par le module appelant. Autrement,
   b) Sur la base du securityName, les informations concernant l'usager à la destination snmpEngineID, spécifiée par le securityEngineID, sont extraites du magasin local de configuration (LCD, *Local Configuration Datastore*) usmUserTable. Si les informations sur l'usager sont absentes du LCD, une indication d'erreur (unknownSecurityName)

est alors retournée au module appelant.

2) Si le securityLevel spécifie que le message doit être protégé de la divulgation, mais si l'usager ne prend pas en charge de protocole d'authentification et de confidentialité, le message ne peut pas être envoyé. Une indication d'erreur (unsupportedSecurityLevel) est retournée au module appelant.

3) Si le securityLevel spécifie que le message est à authentifier, mais si l'usager ne prend pas en charge de protocole d'authentification, le message ne peut alors pas être envoyé. Une indication d'erreur (unsupportedSecurityLevel) est retournée au module appelant.

4) a) Si le securityLevel spécifie que le message doit être protégé de la divulgation, la séquence d'octets qui représente la mise en série de la scopedPDU est alors chiffrée conformément au protocole de confidentialité de l'utilisateur. Pour ce faire, un appel est passé au module de confidentialité qui met en œuvre le protocole de confidentialité de l'utilisateur conformément à la primitive abstraite :

```
statusInformation = encryptData(          -- succès ou échec
      IN    encryptKey                    -- privKey localisée de l'usager
      IN    dataToEncrypt                 -- scopedPDU mise en série
      OUT   encryptedData                 -- encryptedPDU mise en série
      OUT   privParameters                -- paramètres de confidentialité mis en série
  )
```

statusInformation : indique si le processus de chiffrement a réussi ou non.

encryptKey : la privKey privée localisée de l'usager est la clé secrète qui peut être utilisée par l'algorithme de chiffrement.

dataToEncrypt : la scopedPDU mise en série sont les données à chiffrer.

encryptedData : encryptedPDU représente la scopedPDU chiffrée, codée comme CHAINE D'OCTETS.

privParameters : paramètres de confidentialité, codés comme CHAINE D'OCTETS.

Si le module de confidentialité retourne un échec, le message ne peut alors pas être envoyé et une indication d'erreur (encryptionError) est retournée au module appelant.

Si le module de confidentialité retourne un succès, les privParameters retournés sont alors mis dans le champ msgPrivacyParameters des securityParameters et la encryptedPDU sert de charge utile du message préparé.

Autrement,

b) Si le securityLevel spécifie que le message n'est pas à protéger de la divulgation, une CHAINE D'OCTETS de longueur zéro est alors codée dans le champ msgPrivacyParameters des securityParameters et la scopedPDU en texte source sert de charge utile du message préparé.

5) Le securityEngineID est codé comme CHAINE D'OCTETS dans le champ msgAuthoritativeEngineID de securityParameters. Noter qu'un securityEngineID vide (de longueur zéro) est acceptable pour un message de demande, parce cela va amener le moteur SNMP distant (d'autorité) à retourner une PDU Report avec le securityEngineID approprié inclus dans le msgAuthoritativeEngineID dans les securityParameters de cette PDU Report retournée.

6) a) Si le securityLevel spécifie que le message est à authentifier, les valeurs courantes de snmpEngineBoots et de snmpEngineTime correspondantes du securityEngineID provenant du LCD sont utilisées.
Autrement,
b) Si c'est un message Response ou Report, les valeurs courantes de snmpEngineBoots et snmpEngineTime correspondantes du local snmpEngineID provenant du LCD sont utilisées.
Autrement,
c) Si c'est un message Request, une valeur de zéro est utilisée pour snmpEngineBoots et snmpEngineTime. Cette valeur de zéro est utilisée si snmpEngineID est vide.
Les valeurs sont codées respectivement comme ENTIER dans les champs msgAuthoritativeEngineBoots et msgAuthoritativeEngineTime des securityParameters.

7) Le userName est codé comme CHAINE D'OCTETS dans le champ msgUserName des securityParameters.

8) a) Si le securityLevel spécifie que le message est à authentifier, le message est authentifié conformément au protocole d'authentification de l'usager. Pour ce faire, un appel est passé au module d'authentification qui met en œuvre le protocole d'authentification de l'usager selon la primitive de service abstraite suivante :

```
statusInformation =    authenticateOutgoingMsg(
        IN  authKey                          -- authKey localisée de l'usager
        IN  wholeMsg                         -- message non authentifié
       OUT  authenticatedWholeMsg            -- message authentifié complet
        )
```

statusInformation : indique si l'authentification a réussi ou non.

authKey : la authKey localisée privée de l'usager est la clé secrète qui peut être utilisée par l'algorithme d'authentification.

wholeMsg : c'est le message complet mis en série à authentifier.

authenticatedWholeMsg : c'est le même que l'entrée donnée au service authenticateOutgoingMsg, mais en remplissant de façon appropriée les msgAuthenticationParameters.

Si le module d'authentification retourne un échec, le message ne peut alors pas être envoyé et une indication d'erreur (authenticationFailure) est retournée au module appelant.
Si le module d'authentification retourne un succès, le champ msgAuthenticationParameters est alors mis dans les securityParameters et le authenticatedWholeMsg représente la mise en série du message authentifié en préparation.
Autrement,

b) Si le securityLevel spécifie que le message n'est pas à authentifier, une CHAINE D'OCTETS de longueur zéro est alors codée dans le champ msgAuthenticationParameters des securityParameters. Le wholeMsg est maintenant mis en série et représente alors le message non authentifié en préparation.

9) Le message complété avec sa longueur est retourné au module appelant avec les statusInformation réglées à succès.

3.2 Traitement d'un message SNMP entrant

Ce paragraphe décrit la procédure suivie par un moteur SNMP chaque fois qu'il reçoit un message contenant une opération de gestion au nom d'un utilisateur, avec un niveau de sécurité particulier.

Pour simplifier les éléments de procédure, la livraison des informations d'état n'est pas toujours spécifiée explicitement. En règle générale, si les informations d'état sont disponibles lorsque un message est éliminé, les informations d'état devraient aussi être éliminées. Aussi, une indication d'erreur peut retourner un OID et une valeur pour un compteur incrémenté et facultativement une valeur pour le securityLevel, et des valeurs pour contextEngineID ou contextName pour le compteur. De plus, les données de securityStateReference sont retournées s'il en est de disponibles au moment où l'erreur est détectée.

1) Si le securityParameters reçu n'est pas la mise en série (selon les conventions de la [RFC3417]) d'une CHAINE D'OCTETS formatée conformément aux UsmSecurityParameters définis au paragraphe 2.4, alors le compteur snmpInASNParseErrs [RFC3418] est incrémenté, et une indication d'erreur (parseError) est retournée au module appelant. Noter que ce retour est fait sans l'OID ni la valeur du compteur incrémenté, parce que dans ce cas, il n'y a pas assez d'informations pour générer une PDU Report.

2) Les valeurs des champs de paramètre de sécurité sont extraits de securityParameters. Le securityEngineID à retourner à l'appelant est la valeur du champ msgAuthoritativeEngineID. Les cachedSecurityData sont préparées et une securityStateReference est préparée pour faire référence à ces données.
Les valeurs à mettre en antémémoire sont : msgUserName

3) Si la valeur du champ msgAuthoritativeEngineID dans les securityParameters est inconnue, alors :
   a) un moteur SNMP non d'autorité qui effectue la découverte a la faculté de créer une nouvelle entrée dans son magasin local de configurations (LCD) et de continuer le traitement ; ou
   b) le compteur usmStatsUnknownEngineIDs est incrémenté, et une indication d'erreur (unknownEngineID) ainsi que l'OID et la valeur du compteur incrémenté est retournée au module appelant.
   Noter que dans le cas de réception d'un msgAuthoritativeEngineID de longueur zéro, ou d'une autre taille illégale, b) devrait être choisi pour faciliter la découverte de l'engineID. Autrement, le choix entre a) et b) est à la discrétion de la mise en œuvre.

4) Les informations sur la valeur des champs msgUserName et msgAuthoritativeEngineID sont extraites du magasin de données de configuration locales (LCD, usmUserTable). Si aucune information n'est disponible pour l'usager, le compteur usmStatsUnknownUserNames est alors incrémenté et une indication d'erreur (unknownSecurityName) ainsi que l'OID et la valeur du compteur incrémenté est retournée au module appelant.

5) Si les informations sur l'usager indiquent qu'il ne prend pas en charge le niveau de sécurité demandé par l'appelant, le compteur usmStatsUnsupportedSecLevels est alors incrémenté et une indication d'erreur (unsupportedSecurityLevel) avec l'OID et la valeur du compteur incrémenté est retournée au module appelant.

6) Si le securityLevel spécifie que le message est à authentifier, le message est alors authentifié conformément au protocole d'authentification de l'usager. Pour ce faire, un appel est passé au module d'authentification qui met en œuvre le protocole d'authentification de l'usager selon la primitive de service abstraite suivante :

```
statusInformation =   authenticateIncomingMsg(        -- succès ou échec
      IN   authKey                                    -- authKey localisée de l'usager
      IN   authParameters                             -- comme reçu du réseau
      IN   wholeMsg                                   -- comme reçu du réseau
      OUT  authenticatedWholeMsg                      -- vérifié pour authentification
      )
```

statusInformation : indique si l'authentification a réussi ou non.

authKey : la authKey privée localisée de l'usager est la clé secrète qui peut être utilisée par l'algorithme d'authentification .

wholeMsg : le message complet à authentifier mis en série.

authenticatedWholeMsg : le même que l'entrée donnée au service authenticateIncomingMsg, mais après vérification de l'authentification.

Si le module d'authentification retourne un échec, le message ne peut alors pas être de confiance, aussi le compteur usmStatsWrongDigests est incrémenté et une indication d'erreur (authenticationFailure) ainsi que l'OID et la valeur du compteur incrémenté est retournée au module appelant.
Si le module d'authentification retourne le succès, le message est authentique et peut être cru de sorte que le traitement continue.

7) Si le securityLevel indique un message authentifié, les valeurs locales de snmpEngineBoots, snmpEngineTime et latestReceivedEngineTime correspondantes à la valeur du champ msgAuthoritativeEngineID sont alors extraites du magasin local de données de configuration.
   a) Si la valeur extraite de msgAuthoritativeEngineID est la même que la valeur de snmpEngineID du moteur SNMP traitant (c'est à dire le moteur SNMP d'autorité) alors si une des conditions suivantes est vraie, le message est considéré comme en dehors de la fenêtre horaire :
      - la valeur locale de snmpEngineBoots est 2 147 483 647 ;
      - la valeur du champ msgAuthoritativeEngineBoots diffère de la valeur locale de snmpEngineBoots ; ou,
      - la valeur du champ msgAuthoritativeEngineTime diffère de la notion locale de snmpEngineTime de plus de 150 s.

      Si le message est considéré comme en dehors de la fenêtre horaire, alors le compteur usmStatsNotInTimeWindows est incrémenté et une indication d'erreur (notInTimeWindow) avec l'OID, la valeur du compteur incrémenté, et une indication que l'erreur doit être rapportée avec un securityLevel de authNoPriv, est retournée au module appelant.
   b) Si la valeur extraite de msgAuthoritativeEngineID n'est pas la même que la valeur snmpEngineID du moteur SNMP traitant (ce qui signifie que ce n'est pas le moteur SNMP d'autorité), alors :
      1) si au moins une des conditions suivantes est vraie :
         - la valeur extraite du champ msgAuthoritativeEngineBoots est supérieure à la notion locale de la valeur de snmpEngineBoots ; ou,
         - la valeur extraite du champ msgAuthoritativeEngineBoots est égale à la notion locale de la valeur de snmpEngineBoots, et la valeur extraite du champ msgAuthoritativeEngineTime est supérieure à la valeur de latestReceivedEngineTime,

      alors l'entrée de LCD correspondant à la valeur extraite du champ msgAuthoritativeEngineID est mise à jour, en réglant :
      - la notion locale de la valeur de snmpEngineBoots à la valeur du champ msgAuthoritativeEngineBoots,
      - la notion locale de la valeur de snmpEngineTime à la valeur du champ msgAuthoritativeEngineTime, et
      - le latestReceivedEngineTime à la valeur du champ msgAuthoritativeEngineTime.

2) si une des conditions suivantes est vraie, le message est alors considéré comme en dehors de la fenêtre horaire :
- la notion locale de la valeur de snmpEngineBoots est 2 147 483 647 ;
- la valeur du champ msgAuthoritativeEngineBoots est inférieure à la notion locale de la valeur de snmpEngineBoots ; ou,
- la valeur du champ msgAuthoritativeEngineBoots est égale à la notion locale de la valeur de snmpEngineBoots et la valeur du champ msgAuthoritativeEngineTime est de plus de 150 secondes inférieure à la notion locale de la valeur de snmpEngineTime.

Si le message est considéré comme en dehors de la fenêtre horaire, une indication d'erreur (notInTimeWindow) est retournée au module appelant.
Noter que cela signifie qu'un message trop vieux (éventuellement répété) a été détecté et est réputé inauthentique
Noter que cette procédure permet que la valeur de msgAuthoritativeEngineBoots dans le message soit supérieure à la notion locale de la valeur de snmpEngineBoots pour permettre que les messages reçus soient acceptés comme authentiques lorsque ils sont reçus d'un moteur SNMP d'autorité qui a réamorcé depuis que le moteur SNMP receveur s'est (re-)synchronisé pour la dernière fois.

8) a) Si le securityLevel indique que le message était protégé contre la divulgation, la CHAINE D'OCTETS qui représente la encryptedPDU est alors déchiffrée conformément au protocole de confidentialité de l'usager pour obtenir une valeur de scopedPDU déchiffrée mise en série. Pour ce faire, on invoque le module de confidentialité qui met en œuvre le protocole de confidentialité de l'usager selon la primitive abstraite suivante :

```
statusInformation =  decryptData(              -- succès ou échec
    IN    decryptKey                           -- privKey localisée de l'usager
    IN    privParameters                       -- comme reçu du réseau
    IN    encryptedData                        -- encryptedPDU comme reçue
    OUT   decryptedData                        -- scopedPDU déchiffrée mis en série
    )
```

statusInformation : indique si le processus de déchiffrement a réussi ou non.

decryptKey : la privKey privée localisée de l'usager est la clé secrète qui peut être utilisée par l'algorithme de déchiffrement.

privParameters : les msgPrivacyParameters, codés comme une CHAINE D'OCTETS.

encryptedData : la encryptedPDU représente la scopedPDU chiffrée, codée comme CHAINE D'OCTETS.

decryptedData : la scopedPDU mise en série si le déchiffrement a réussi.

Si le module de confidentialité retourne un échec, le message ne peut alors pas être traité, donc le compteur usmStatsDecryptionErrors est incrémenté et une indication d'erreur (decryptionError) ainsi que l'OID et la valeur du compteur incrémenté est retournée au module appelant.
Si le module de confidentialité retourne le succès, alors la scopedPDU déchiffrée est la charge utile du message à retourner au module appelant. Autrement,

b) Le composant scopedPDU est supposé être en texte source et est la charge utile du message à retourner au module appelant.

9) La maxSizeResponseScopedPDU est calculée. C'est la taille maximum admise pour une scopedPDU pour un éventuel message Response. Des dispositions sont prises pour un en-tête de message permettant le même securityLevel que celui de la demande reçue.

10) Le securityName pour l'usager est restitué du usmUserTable.

11) Les données de sécurité sont mises en antémémoire comme cachedSecurityData, de sorte qu'une éventuelle réponse à ce message puisse utiliser les mêmes secrets d'authentification et de confidentialité. Les informations à sauvegarder/mettre en antémémoire sont les suivantes :
msgUserName, usmUserAuthProtocol, usmUserAuthKey, usmUserPrivProtocol, usmUserPrivKey.

12) Les statusInformation sont réglées à succès et un retour est fait au module appelant en repassant les paramètres OUT comme spécifié dans la primitive processIncomingMsg.

4. Découverte

Le modèle de sécurité fondé sur l'utilisateur exige qu'un processus de découverte obtienne des informations suffisantes sur les autres moteurs SNMP afin de communiquer avec eux. La découverte exige qu'un moteur SNMP non d'autorité apprenne la valeur du snmpEngineID du moteur SNMP d'autorité avant que la communication puisse avoir lieu. Ceci peut être fait en générant un message de demande avec un niveau de sécurité de noAuthNoPriv, un msgUserName de longueur zéro, une valeur de msgAuthoritativeEngineID de longueur zéro, et en laissant vide la varBindList. La réponse à ce message sera un message Report contenant le snmpEngineID du moteur SNMP d'autorité comme valeur du champ msgAuthoritativeEngineID au sein du champ msgSecurityParameters. Il contient une PDU Report avec le compteur usmStatsUnknownEngineIDs dans la liste varBindList.

Si une communication authentifiée est exigée, le processus de découverte devrait alors aussi établir la synchronisation horaire avec le moteur SNMP d'autorité. Cela peut se réaliser en envoyant un message Request authentifié avec la valeur de msgAuthoritativeEngineID réglée au snmpEngineID nouvellement appris et avec les valeurs de msgAuthoritativeEngineBoots et msgAuthoritativeEngineTime réglées à zéro. Pour un message de demande authentifié, un nom d'utilisateur valide doit être utilisé dans le champ msgUserName. La réponse à ce message authentifié sera un message Report contenant les valeurs à jour des snmpEngineBoots et snmpEngineTime du moteur SNMP d'autorité comme valeurs des champs msgAuthoritativeEngineBoots et msgAuthoritativeEngineTime respectivement. Elle contient aussi le compteur usmStatsNotInTimeWindows dans la varBindList de la PDU Report. La synchronisation horaire se produit alors automatiquement au titre des procédures du paragraphe 3.2 étape 7b. Voir aussi au paragraphe 2.3.

5. Définitions

DÉFINITIONS DE SNMP-USER-BASED-SM-MIB ::= DÉBUT

IMPORTE
IDENTITÉ DE MODULE, TYPE D'OBJET, IDENTITÉ D'OBJET, snmpModules, Counter32 DE SNMPv2-SMI
CONVENTION TEXTUELLE, TestAndIncr, RowStatus, RowPointer, StorageType, AutonomousType DE SNMPv2-TC
CONFORMITÉ DE MODULE, GROUPE D'OBJET DE SNMPv2-CONF
SnmpAdminString, SnmpEngineID, snmpAuthProtocols, snmpPrivProtocols DE SNMP-FRAMEWORK-MIB;

IDENTITÉ DE MODULE snmpUsmMIB
DERNIERE MISE A JOUR : "200210160000Z" -- 16 octobre 2002, minuit
ORGANISATION : "Groupe de travail SNMPv3 "
INFORMATIONS DE CONTACT "messagerie du groupe de travail : snmpv3@lists.tislabs.com
Pour s'abonner : majordomo@lists.tislabs.com ; mettre dans le corps du message : subscribe snmpv3

Coprésident : Russ Mundy
Adresse postale : Network Associates Laboratories, 15204 Omega Drive, Suite 300, Rockville, MD 20850-4601, USA
mél : mundy@tislabs.com
téléphone : +1 301-947-7107

Coprésident : David Harrington
Adresse postale : Enterasys Networks, 35 Industrial Way, P. O. Box 5004, Rochester, New Hampshire 03866-5005, USA
mél : dbh@enterasys.com
téléphone : +1 603-337-2614

Co-éditeur : Uri Blumenthal
Adresse postale : Lucent Technologies, 67 Whippany Rd., Whippany, NJ 07981, USA
mél : uri@lucent.com
téléphone : +1 973-386-2163

Co-éditeur : Bert Wijnen
Adresse postale : Lucent Technologies, Schagen 33, 3461 GL Linschoten, Netherlands
mél : bwijnen@lucent.com
téléphone : +31-348-480-685 "
DESCRIPTION : "Définitions des information de gestion pour le modèle de sécurité SNMP fondé sur l'utilisateur. Copyright (C) The Internet Society (2002). Cette version de ce module de MIB fait partie de la RFC 3414 ; voir les notices légales complètes dans la RFC elle-même. "

-- Historique des révisions

REVISION : "200210160000Z" -- 16 octobre 2002, minuit
DESCRIPTION : "Changements dans la présente révision : Mise à jour des références et des informations de contact. - Précisions à la clause DESCRIPTION de usmUserCloneFrom. Correction de "répondeur de commandes" en "générateur de commandes" dans le dernier paragraphe de la clause DESCRIPTION de usmUserTable. Cette révision est publiée comme RFC3414. "
REVISION : "199901200000Z" -- 20 janvier 1999, minuit
DESCRIPTION "Clarifications, publiée comme RFC2574"
REVISION : "199711200000Z" -- 20 novembre 1997, minuit
DESCRIPTION : "Version initiale, publiée comme RFC2274"
::= { snmpModules 15 }

-- Allocations administratives **********

usmMIBObjects IDENTIFANT D'OBJET ::= { snmpUsmMIB 1 }
usmMIBConformance IDENTIFANT D'OBJET ::= { snmpUsmMIB 2 }

-- Identification des protocoles d'authentification et de confidentialité **********

IDENTITÉ D'OBJET usmNoAuthProtocol
STATUT :actuel
DESCRIPTION : "Pas de protocole d'authentification."
::= { snmpAuthProtocols 1 }

IDENTITÉ D'OBJET usmHMACMD5AuthProtocol
STATUT : actuel
DESCRIPTION : "Le protocole d'authentification par résumé HMAC-MD5-96."
REFERENCE : " RFC2104, H. Krawczyk, M. Bellare, R. Canetti "HMAC: hachage de chiffrement pour l'authentification de message", février 1997 ; RFC1321, R Rivest, "Algorithme MD5 de résumé de message", avril 1992. "
::= { snmpAuthProtocols 2 }

IDENTITÉ D'OBJET usmHMACSHAAuthProtocol
STATUT : actuel
DESCRIPTION : "Le protocole d'authentification par résumé HMAC-SHA-96."
REFERENCE : "RFC2104, H. Krawczyk, M. Bellare, R. Canetti "HMAC: hachage de chiffrement pour l'authentification de message", février 1997 ; "Secure Hash Algorithm". NIST FIPS 180-1. "
::= { snmpAuthProtocols 3 }

IDENTITÉ D'OBJET usmNoPrivProtocol
STATUT :actuel
DESCRIPTION : "Pas de protocole de confidentialité."
::= { snmpPrivProtocols 1 }

IDENTITÉ D'OBJET usmDESPrivProtocol
STATUT : actuel
DESCRIPTION : "Protocole de chiffrement symétrique CBC-DES."
REFERENCE : "- "Data Encryption Standard", National Institute of Standards and Technology. Federal Information Processing Standard (FIPS) Publication 46-1. Remplace FIPS Publication 46, (janvier 1977 ; réaffirmée en janvier 1988) ; - "Data Encryption Algorithm", American National Standards Institute. ANSI X3.92-1981, (décembre 1980) ; - "DES Modes of Operation", National Institute of Standards and Technology. Federal Information Processing Standard (FIPS) Publication 81, (décembre 1980) : - "Data Encryption Algorithm - Modes of Operation", American National Standards Institute. ANSI X3.106-1983, (mai 1983). "
::= { snmpPrivProtocols 2 }

-- Conventions textuelles **********

KeyChange ::= CONVENTION TEXTUELLE
STATUT : actuel
DESCRIPTION : "Chaque définition d'un objet avec cette syntaxe doit identifier un protocole P, une clé secrète K, et un algorithme de hachage H qui produit un résultat de L octets. La valeur de l'objet est une valeur partiellement aléatoire générée par le gestionnaire, qui lorsque elle est modifiée, cause la modification de la valeur de la clé secrète K, via une fonction unidirectionnelle.

La valeur d'une instance de cet objet est l'enchaînement de deux composants : d'abord un composant "aléatoire" et ensuite un composant "delta". Les longueurs des composants aléatoire et delta sont données par la valeur correspondante du protocole P ; si P exige que K ait une longueur fixe, la longueur des deux composants aléatoire et delta est cette longueur fixe ; si P permet que la longueur de K soit variable jusqu'à une longueur maximum particulière, la longueur du composant aléatoire est cette longueur maximum et la longueur du composant delta est toute longueur inférieure ou égale à cette longueur maximum. Par exemple, usmHMACMD5AuthProtocol exige que K ait une longueur fixe de 16 octets et L de 16 octets. usmHMACSHAAuthProtocol exige que K ait une longueur fixe de 20 octets et L de 20 octets. D'autres protocoles peuvent définir d'autres tailles, comme ils l'estiment approprié.
Lorsque un demandeur veut changer la vieille clé K en une nouvelle clé keyNew sur une entité distante, le composant "aléatoire" est obtenu soit d'un générateur de vrai aléa, soit d'un générateur pseudo aléatoire, et le composant "delta" est calculé comme suit :
- une variable temporaire est initialisée à la valeur existante de K ;
- si la longueur de keyNew est supérieure à L octets, alors :
- le composant aléatoire est ajouté à la valeur de la variable temporaire, et le résultat est entré dans l'algorithme de hachage H pour produire une valeur de résumé, et la variable temporaire est réglée à cette valeur de résumé ;
- la valeur de la variable temporaire est OU-ixée avec les (prochains) L octets (16 octets dans le cas de MD5) de la keyNew pour produire les premiers (prochains) L octets (16 octets dans le cas de MD5) du composant "delta".
- les deux étapes ci-dessus sont répétées jusqu'à ce que la portion inutilisée du composant keyNew fasse L octets ou moins,
- le composant aléatoire est ajouté à la valeur de la variable temporaire, et le résultat est en entrée à l'algorithme de hachage H pour produire une valeur de résumé ;
- cette valeur de résumé, tronquée si nécessaire pour avoir la même longueur que la portion inutilisée de keyNew, est OU-ixée avec la portion inutilisée de keyNew pour produire le composant (la portion finale du) "delta".
Par exemple, en utilisant MD5 comme algorithme H de hachage :
itérations = (lenOfDelta - 1)/16; /* division entière */
temp = keyOld;
pour (i = 0; i < itérations ; i) {
temp = MD5 (temp || random);
delta[i*16 .. (i*16)] = temp OUX keyNew[i*16 .. (i*16)]; }
temp = MD5 (temp || random);
delta[i*16 .. lenOfDelta-1] = temp OUX keyNew[i*16 .. lenOfDelta-1];
Les composants 'random' et 'delta' sont alors enchaînés comme décrit ci-dessus, et la chaîne d'octets résultante est envoyée au receveur comme nouvelle valeur d'une instance de cet objet.
Du côté receveur, lorsque une instance de cet objet est réglée à une nouvelle valeur, une nouvelle valeur de K est calculée comme suit :
- une variable temporaire est initialisée à la valeur existante de K ;
- si la longueur du composant delta est supérieurs à L octets, alors :
- le composant aléatoire est ajouté à la valeur de la variable temporaire, et le résultat est entré dans l'algorithme de hachage H pour produire une valeur de résumé, et la variable temporaire est réglée à cette valeur de résumé ;
- la valeur de la variable temporaire est OU-ixée avec les L premiers (prochains) octets (16 octets dans le cas de MD5) du composant delta pour produire les L premiers (prochains) octets (16 octets dans le cas de MD5) de la nouvelle valeur de K.
- les deux étapes ci-dessus sont répétées jusqu'à ce que la portion inutilisée du composant delta soit de L octets ou moins,
- le composant aléatoire est ajouté à la valeur de la variable temporaire, et le résultat est entré dans l'algorithme de hachage H pour produire une valeur de résumé ;
- cette valeur de résumé, tronquée si nécessaire pour avoir la même longueur que la portion inutilisée du composant delta, est OU-ixée avec la portion inutilisée du composant delta pour produire la nouvelle (la portion finale de) valeur de K.
Par exemple, en utilisant MD5 comme algorithme de hachage H :
itérations = (lenOfDelta - 1)/16; /* division entière */
temp = keyOld; pour (i = 0; i < itérations; i) { temp = MD5 (temp || random); keyNew[i*16 .. (i*16)] = temp OUX delta[i*16 .. (i*16)]; }
temp = MD5 (temp || random); keyNew[i*16 .. lenOfDelta-1] = temp OUX delta[i*16 .. lenOfDelta-1];
La valeur d'un objet avec cette syntaxe, chaque fois qu'elle est restituée par le protocole de gestion, est toujours la chaîne de longueur zéro.
Noter que keyOld et keyNew sont les clés localisées.
Noter qu'il est probablement avisé que lorsque une entité SNMP envoie une SetRequest pour changer une

clé, elle conserve une copie de la vieille clé jusqu'à ce qu'il ait été confirmé que le changement de clé a bien réussi. "
SYNTAXE : CHAINE D'OCTETS

-- Statistiques pour le modèle de sécurité fondé sur l'utilisateur ********

IDENTIFANT D'OBJET usmStats ::= { usmMIBObjects 1 }

TYPE D'OBJET usmStatsUnsupportedSecLevels
SYNTAXE : Counter32
MAX-ACCESS : lecture seule
STATUT : actuel
DESCRIPTION : "nombre total de paquets reçus par le moteur SNMP qui ont été éliminés parce qu'ils demandaient un niveau de sécurité inconnu du moteur SNMP ou par ailleurs indisponible. "
::= { usmStats 1 }

TYPE D'OBJET usmStatsNotInTimeWindows
SYNTAXE : Counter32
MAX-ACCESS : lecture seule
STATUT : actuel
DESCRIPTION : "nombre total de paquets reçus par le moteur SNMP qui ont été éliminés parce qu'ils sont apparus en dehors de la fenêtre du moteur SNMP d'autorité. "
::= { usmStats 2 }

TYPE D'OBJET usmStatsUnknownUserNames
SYNTAXE : Counter32
MAX-ACCESS : lecture seule
STATUT : actuel
DESCRIPTION : "nombre total de paquets reçus par le moteur SNMP qui ont été éliminés parce qu'ils font référence à un usager qui n'est pas connu du moteur SNMP. "
::= { usmStats 3 }

TYPE D'OBJET usmStatsUnknownEngineIDs
SYNTAXE : Counter32
MAX-ACCESS : lecture seule
STATUT : actuel
DESCRIPTION : "nombre total de paquets reçus par le moteur SNMP qui ont été éliminés parce qu'ils font référence à un snmpEngineID qui n'est pas connu du moteur SNMP. "
::= { usmStats 4 }

TYPE D'OBJET usmStatsWrongDigests
SYNTAXE : Counter32
MAX-ACCESS : lecture seule
STATUT : actuel
DESCRIPTION : "nombre total de paquets reçus par le moteur SNMP qui ont été éliminés parce qu'ils ne contenaient pas la valeur de résumé attendue. "
::= { usmStats 5 }

TYPE D'OBJET usmStatsDecryptionErrors
SYNTAXE : Counter32
MAX-ACCESS : lecture seule
STATUT : actuel
DESCRIPTION : "nombre total de paquets reçus par le moteur SNMP qui ont été éliminés parce qu'ils n'ont pas pu être déchiffrés. "
::= { usmStats 6 }

-- Groupe usmUser ********

IDENTIFANT D'OBJET usmUser ::= { usmMIBObjects 2 }

TYPE D'OBJET usmUserSpinLock
SYNTAXE : TestAndIncr
MAX-ACCESS : lecture-écriture

STATUT : actuel
DESCRIPTION : "verrou facultatif utilisé pour permettre à plusieurs applications coopérantes de générateur de commandes de coordonner leur utilisation de facilités pour altérer les secrets dans usmUserTable. "
::= { usmUser 1 }

-- Tableau des utilisateurs valides pour le modèle de sécurité fondé sur l'utilisateur ******

TYPE D'OBJET usmUserTable
SYNTAXE : SEQUENCE DE UsmUserEntry
MAX-ACCESS : non accessible
STATUT : actuel
DESCRIPTION : "Tableau des usagers configurés dans le magasin de données locales de configuration du moteur SNMP. Pour créer un nouvel usager (instancier une nouvelle rangée conceptuelle dans le tableau) il est recommandé de suivre cette procédure :
1) GET(usmUserSpinLock.0) et le sauvegarder dans sValue.
2) SET(usmUserSpinLock.0=sValue, usmUserCloneFrom=templateUser, usmUserStatus=createAndWait). On devrait utiliser un gabarit d'utilisateur à dupliquer à partir duquel est défini le protocole approprié d'authentification/confidentialité. Si le nouvel usager va utiliser la confidentialité :
3) générer la valeur de keyChange sur la base de la privKey secrète de l'usager provenant du duplicata et la clé secrète à utiliser pour le nouvel usager. Appelons cela pkcValue.
4) GET(usmUserSpinLock.0) et sauvegarder dans sValue.
5) SET(usmUserSpinLock.0=sValue, usmUserPrivKeyChange=pkcValue usmUserPublic=randomValue1)
6) GET(usmUserPulic) et vérifier qu'il a randomValue1. Sinon, répéter les étapes 4 à 6.
Si le nouvel usager ne va jamais utiliser la confidentialité :
7) SET(usmUserPrivProtocol=usmNoPrivProtocol)
Si le nouvel usager va utiliser l'authentification :
8) générer la valeur de keyChange sur la base de la authKey secrète de l'usager provenant du duplicata et la clé secrète à utiliser pour le nouvel usager. Appelons cela akcValue.
9) GET(usmUserSpinLock.0) et sauvegarder dans sValue.
10) SET(usmUserSpinLock.0=sValue, usmUserAuthKeyChange=akcValue usmUserPublic=randomValue2)
11) GET(usmUserPulic) et vérifier qu'il a randomValue2. Sinon, répéter les étapes 9 à 11. Si le nouvel usager ne va jamais utiliser l'authentification :
12) SET(usmUserAuthProtocol=usmNoAuthProtocol)
Finalement, activer le nouvel usager :
13) SET(usmUserStatus=active)
Le nouvel usager devrait maintenant être disponible et prêt à l'usage pour la communication SNMPv3. Noter cependant que l'accès aux données de MIB doit être fourni via la configuration de la SNMP-VIEW-BASED-ACM-MIB.
L'utilisation de usmUserSpinlock est destinée à éviter des conflits avec une autre application de générateur de commandes SNMP qui pourrait aussi être en train d'agir sur le tableau usmUserTable. "
::= { usmUser 2 }

TYPE D'OBJET usmUserEntry
SYNTAXE : UsmUserEntry
MAX-ACCESS : non accessible
STATUT : actuel
DESCRIPTION : "usager configuré dans le magasin de sonnées de configuration locale du moteur SNMP pour le modèle de sécurité fondé sur l'utilisateur. "
INDEX : { usmUserEngineID, usmUserName }
::= { usmUserTable 1 }

```
UsmUserEntry ::= SEQUENCE {
    usmUserEngineID             SnmpEngineID,
    usmUserName                 SnmpAdminString,
    usmUserSecurityName         SnmpAdminString,
    usmUserCloneFrom            RowPointer,
    usmUserAuthProtocol         AutonomousType,
    usmUserAuthKeyChange        KeyChange,
    usmUserOwnAuthKeyChange     KeyChange,
    usmUserPrivProtocol         AutonomousType,
    usmUserPrivKeyChange        KeyChange,
    usmUserOwnPrivKeyChange     KeyChange,
    usmUserPublic               CHAINE D'OCTETS,
```

```
    usmUserStorageType      StorageType,
    usmUserStatus           RowStatus
  }
```

TYPE D'OBJET usmUserEngineID
SYNTAXE : SnmpEngineID
MAX-ACCESS : non accessible
STATUT :actuel
DESCRIPTION : "Identifiant administrativement unique d'un moteur SNMP. Dans un simple agent, cette valeur est toujours la valeur propre de snmpEngineID de cet agent. La valeur peut aussi prendre la valeur du snmpEngineID d'un moteur SNMP distant avec lequel cet usager peut communiquer. "
::= { usmUserEntry 1 }

TYPE D'OBJET usmUserName
SYNTAXE : SnmpAdminString (SIZE(1..32))
MAX-ACCESS : non accessible
STATUT : actuel
DESCRIPTION : "Chaîne lisible par l'homme qui représente le nom de l'usager. C'est l'identifiant de sécurité dépendant du modèle (de sécurité fondée sur l'utilisateur). "
::= { usmUserEntry 2 }

TYPE D'OBJET usmUserSecurityName
SYNTAXE : SnmpAdminString
MAX-ACCESS : lecture seule
STATUT : actuel
DESCRIPTION : "Chaîne lisible par l'homme qui représente l'usager dans un format indépendant du modèle de sécurité. La transformation par défaut de l'identifiant de sécurité dépendant du modèle de sécurité fondé sur l'utilisateur en securityName et vice versa est la fonction d'identité de sorte que le securityName est le même que le userName. "
::= { usmUserEntry 3 }

TYPE D'OBJET usmUserCloneFrom
SYNTAXE : RowPointer
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Pointeur sur une autre rangée conceptuelle dans ce usmUserTable. L'usager dans cette autre rangée conceptuelle est appelé l'usager provenant du duplicata. Lorsque un nouvel usager est créé (c'est-à-dire, qu'une nouvelle rangée conceptuelle est instanciée dans ce tableau) les paramètres de confidentialité et d'authentification du nouvel usager doivent être dupliqués à partir ce son usager provenant du duplicata. Ces paramètres sont :
- le protocole d'authentification (usmUserAuthProtocol)
- le protocole de confidentialité (usmUserPrivProtocol)
Ils seront copiés sans considération de ce qu'est la valeur réelle. Les duplicata sont aussi la cause pour laquelle les valeurs initiales de la clé d'authentification secrète (authKey) et la clé secrète de chiffrement (privKey) du nouvel usager sont réglées à la même valeur que les secrets correspondants de l'usager provenant du duplicata pour permettre que se produise le processus de changement de clé (KeyChange) comme exigé lors d'une création d'usager.
La première fois qu'une instance de cet objet est réglée par une opération de gestion (soit au moment de son instanciation, soit après) le processus de duplication est invoqué. Les inscriptions suivantes réussissent mais n'invoquent pas d'action à prendre par le receveur. Le processus de duplication échoue avec une erreur "inconsistentName" si la rangée conceptuelle qui représente l'usager provenant de la duplication n'existe pas ou n'est pas en état actif lorsque le processus de duplication est invoqué. Lorsque cet objet est en écriture, l'OID ZeroDotZero est retourné. "
::= { usmUserEntry 4 }

TYPE D'OBJET usmUserAuthProtocol
SYNTAXE : AutonomousType
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Indique si les messages envoyés au nom de cet usager au/du moteur SNMP identifié par usmUserEngineID peuvent être authentifiés, et si ils le peuvent, le type de protocole d'authentification utilisé. Une instance de cet objet est créée concurremment avec la création de toute autre instance d'objet pour le même usager (c'est-à-dire, au titre du traitement de l'opération SET qui crée la première instance

d'objet dans la même rangée conceptuelle). Si une opération SET initiale (c'est-à-dire, au moment de la création de la rangée) essaye de régler une valeur pour un protocole inconnu ou non pris en charge, une erreur 'wrongValue' doit alors être retournée. La valeur sera écrasée/réglée lorsque une opération SET sera effectuée sur l'instance correspondante de usmUserCloneFrom. Une fois instanciée, la valeur d'une telle instance de cet objet ne peut être changée que via une opération SET à la valeur du usmNoAuthProtocol. Si une opération SET essaye de changer la valeur d'une instance existante de cet objet à toute valeur autre que usmNoAuthProtocol, une erreur 'inconsistentValue' doit alors être retournée. Si une opération SET essaye de régler la valeur à usmNoAuthProtocol alors que la valeur usmUserPrivProtocol dans la même rangée n'est pas égale à usmNoPrivProtocol, une erreur 'inconsistentValue' doit alors être retournée. Cela signifie qu'une application de générateur de commandes SNMP doit d'abord s'assurer que le usmUserPrivProtocol est réglé à la valeur usmNoPrivProtocol avant qu'il puisse régler la valeur de usmUserAuthProtocol à usmNoAuthProtocol. "

DEFVAL : { usmNoAuthProtocol }
::= { usmUserEntry 5 }

TYPE D'OBJET usmUserAuthKeyChange
SYNTAXE : KeyChange -- normalement (TAILLE (0 | 32)) pour HMACMD5, et (TAILLE (0 | 40)) pour HMACSHA
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Objet, qui lorsque modifié, cause la modification de la clé secrète d'authentification utilisée pour les messages envoyés au nom de cet usager au/du moteur SNMP identifié par usmUserEngineID, via une fonction unidirectionnelle. Le protocole associé est le usmUserAuthProtocol. La clé secrète associée est la clé d'authentification secrète de l'usager (authKey). l'algorithme de hachage associé est celui utilisé par le usmUserAuthProtocol de l'usager.

Lors de la création d'un nouvel usager, c'est une erreur 'inconsistentName' pour une opération SET de se référer à cet objet sauf si il a été antérieurement ou concurremment initialisé par une opération SET sur l'instance correspondante de usmUserCloneFrom.

Lorsque la valeur du usmUserAuthProtocol correspondant est usmNoAuthProtocol, un SET réussit, mais est sans effet. Lorsque cet objet est en lecture, la chaîne de longueur zéro (vide) est retournée. La façon recommandée de faire un changement de clé est la suivante : 1) GET(usmUserSpinLock.0) et sauvegarder dans sValue. 2) générer la valeur keyChange sur la base de la vieille clé secrète (existante) et de la nouvelle clé secrète ; on l'appelle kcValue.

Si on fait le changement de clé au nom d'un autre usager : 3) SET(usmUserSpinLock.0=sValue, usmUserAuthKeyChange=kcValue usmUserPublic=randomValue)

Si on fait le changement de clé pour soi-même : 4) SET(usmUserSpinLock.0=sValue, usmUserOwnAuthKeyChange=kcValue usmUserPublic=randomValue). Si on obtient une réponse avec l'état d'erreur de noError, l'opération SET a réussi et la nouvelle clé est active. Si on n'obtient pas de réponse, on peut produire un GET(usmUserPublic) et vérifier si la valeur est égale à la randomValue qu'on a envoyée dans le SET. Si il en est ainsi, le changement de clé a réussi et la nouvelle clé est active (la réponse a probablement été perdue). Sinon, la demande SET n'a probablement jamais atteint la cible et donc on peut recommencer la procédure ci-dessus. "

DEFVAL : { "H } -- la chaîne vide.
::= { usmUserEntry 6 }

TYPE D'OBJET usmUserOwnAuthKeyChange
SYNTAXE : KeyChange -- normalement (TAILLE (0 | 32)) pour HMACMD5 et (TAILLE (0 | 40)) pour HMACSHA
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Se comporte exactement comme usmUserAuthKeyChange, avec une différence notable : afin que l'opération SET réussisse, le usmUserName du demandeur de l'opération doit correspondre au usmUserName qui est l'indice de la rangée qui est ciblée par cette opération. De plus, le modèle de sécurité USM doit être utilisé pour cette opération. L'idée est ici que l'accès à cette colonne puisse être public, car il va seulement permettre à un utilisateur de changer sa propre clé secrète d'authentification (authKey). Noter que ceci ne peut être fait qu'une fois que la rangée est active.

Lorsque un SET est reçu et que le usmUserName du demandeur n'est pas le même que le umsUserName qui indexe la rangée ciblée par cette opération, une erreur 'noAccess' doit être retournée. Lorsque un SET est reçu et que le modèle de sécurité utilisé n'est pas USM, une erreur 'noAccess' doit être retournée. "

DEFVAL : { "H } -- la chaîne vide.
::= { usmUserEntry 7 }

TYPE D'OBJET usmUserPrivProtocol
SYNTAXE : AutonomousType
MAX-ACCESS : lecture-création

STATUT : actuel
DESCRIPTION : "Indique si les messages envoyés au nom de cet usager au/du moteur SNMP identifié par usmUserEngineID, peuvent être protégés de la divulgation, et si c'est le cas, le type de protocole de confidentialité utilisé.
Une instance de cet objet est créée concurremment avec la création de toute autre instance d'objet pour le même usager (c'est-à-dire, au titre du traitement de l'opération SET qui crée la première instance d'objet dans la même rangée conceptuelle). Si une opération SET initiale (c'est-à-dire, au moment de la création de la rangée) essaye de régler une valeur pour un protocole inconnu ou non pris en charge, une erreur 'wrongValue' doit alors être retournée.
La valeur sera écrasée/réglée lorsque une opération SET sera effectuée sur l'instance correspondante de usmUserCloneFrom. Une fois instanciée, la valeur d'une telle instance de cet objet ne peut être changée que via une opération SET à la valeur du usmNoPrivProtocol. Si une opération SET essaye de changer la valeur d'une instance existante de cet objet pour toute valeur autre que usmNoPrivProtocol, une erreur 'inconsistentValue' doit alors être retournée.
Noter que si un protocole de confidentialité est utilisé, on doit alors aussi utiliser un protocole d'authentification. En d'autres termes, si usmUserPrivProtocol est réglé à autre chose que usmNoPrivProtocol, l'instance correspondante de usmUserAuthProtocol ne peut pas avoir une valeur de usmNoAuthProtocol. Si elle l'a, une erreur 'inconsistentValue' doit alors être retournée. "
DEFVAL : { usmNoPrivProtocol }
::= { usmUserEntry 8 }

TYPE D'OBJET usmUserPrivKeyChange
SYNTAXE : KeyChange -- normalement (TAILLE (0 | 32)) pour DES
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Objet qui, lorsque modifié, cause la modification de la clé de chiffrement secrète utilisée pour les messages envoyés au nom de cet usager au/du moteur SNMP identifié par usmUserEngineID via une fonction unidirectionnelle. Le protocole associé est le usmUserPrivProtocol. La clé secrète associée est la clé privée secrète de l'usager (privKey). L'algorithme de hachage associé est l'algorithme utilisé par le usmUserAuthProtocol de l'usager.
Lors de la création d'un nouvel usager, c'est une erreur 'inconsistentName' pour une opération SET de se référer à cet objet sauf si il a été précédemment ou concurremment initialisé par une opération SET sur l'instance correspondante de usmUserCloneFrom.
Lorsque la valeur du usmUserPrivProtocol correspondant est usmNoPrivProtocol, l'opération SET est alors réussie, mais est sans effet. Lorsque cet objet est en lecture, la chaîne de longueur zéro (vide) est retournée. Voir à la clause "description" de smUserAuthKeyChange la procédure recommandée de changement de clé. "
DEFVAL : { "H } -- la chaîne vide.
::= { usmUserEntry 9 }

TYPE D'OBJET usmUserOwnPrivKeyChange
SYNTAXE : KeyChange -- normalement (TAILLE (0 | 32)) pour DES
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Se comporte exactement comme usmUserPrivKeyChange, avec une différence notable : afin que l'opération SET réussisse, le usmUserName du demandeur de l'opération doit correspondre au usmUserName qui indexe la rangée ciblée par cette opération. De plus, le modèle de sécurité USM doit être utilisé pour cette opération.
L'idée est ici que l'accès à cette colonne puisse être public, car il va seulement permettre à un utilisateur de changer sa propre clé secrète de confidentialité (privKey). Noter que ceci ne peut être fait que lorsque la rangée est active.
Lorsque un SET est reçu et que le usmUserName du demandeur n'est pas le même que le umsUserName qui indexe la rangée ciblée par l'opération, une erreur 'noAccess' doit alors être retournée.
Lorsque un SET est reçu et que le modèle de sécurité utilisé n'est pas USM, une erreur 'noAccess' doit être retournée. "
DEFVAL : { "H } -- la chaîne vide.
::= { usmUserEntry 10 }

TYPE D'OBJET usmUserPublic
SYNTAXE : CHAINE D'OCTETS (SIZE(0..32))
MAX-ACCESS : lecture-créationSTATUT : actuel
DESCRIPTION : "Valeur publiquement lisible qui peut être écrite au titre de la procédure de changement de la clé secrète d'authentification et/ou de confidentialité d'un utilisateur, et ultérieurement lue pour déterminer si le

changement du secret a été effectué. "
DEFVAL : { "H } -- la chaîne vide.
::= { usmUserEntry 11 }

TYPE D'OBJET usmUserStorageType
SYNTAXE : StorageType
MAX-ACCESS : lecture-création
STATUT : actuel
DESCRIPTION : "Type de mémorisation pour cette rangée conceptuelle. Les rangées conceptuelles qui ont la valeur de 'permanent' doivent permettre au minimum l'accès en écriture à : usmUserAuthKeyChange, usmUserOwnAuthKeyChange et usmUserPublic, pour un utilisateur qui emploie l'authentification, et usmUserPrivKeyChange, usmUserOwnPrivKeyChange et usmUserPublic, pour un utilisateur qui emploie la confidentialité.
Noter que tout usager qui emploie l'authentification ou la confidentialité doit permettre que son ou ses secrets soient mis à jour et qu'ils ne peuvent donc être en 'readOnly' *(lecture seule)*.
Si une opération SET initiale essaye de régler la valeur à 'readOnly' pour un utilisateur qui emploie l'authentification ou la confidentialité, une erreur 'inconsistentValue' doit alors être retournée. Noter que si la valeur a été précédemment réglée (implicitement ou explicitement) à n'importe quelle valeur, les règles définies dans la convention textuelle StorageType s'appliquent. C'est une question de mise en œuvre de décider si une opération SET pour une rangée readOnly ou permanent. Dans certains contextes ceci peut avoir un sens, et pas dans d'autres. Si un SET pour une rangée readOnly ou permanent n'est pas accepté du tout, une erreur 'wrongValue' doit alors être retournée. "
DEFVAL : { nonVolatile }
::= { usmUserEntry 12 }

TYPE D'OBJET usmUserStatus
SYNTAXE : RowStatus
MAX-ACCESS lecture-création
STATUT : actuel
DESCRIPTION : "Statut de cette rangée conceptuelle. Jusqu'à ce que toutes les instances de toutes les colonnes correspondantes soient configurées de façon appropriée, la valeur de l'instance correspondante de la colonne usmUserStatus est 'notReady'. En particulier, une rangée nouvellement créée pour un utilisateur qui emploie l'authentification ne peut pas être rendue active tant que les usmUserCloneFrom et usmUserAuthKeyChange correspondants n'ont pas été réglés.
De plus, une rangée nouvellement créée pour un utilisateur qui emploie aussi la confidentialité, ne peut pas être rendue active tant que le usmUserPrivKeyChange n'a pas été réglé. La convention textuelle RowStatus [RFC2579] exige que cette clause DESCRIPTION déclare dans quelles circonstances les autres objets de cette rangée peuvent être modifiés: La valeur de cet objet n'a pas d'effet sur la modification des autres objets de cette rangée conceptuelle, sauf pour usmUserOwnAuthKeyChange et usmUserOwnPrivKeyChange. Pour ces deux objets, la valeur de usmUserStatus DOIT être active. "
::= { usmUserEntry 13 }

**-- Informations de conformité **

IDENTIFANT D'OBJET usmMIBCompliances ::= { usmMIBConformance 1 }
IDENTIFANT D'OBJET usmMIBGroups ::= { usmMIBConformance 2 }

-- Déclarations de conformité

CONFORMITÉ DE MODULE usmMIBCompliance
STATUT : actuel
DESCRIPTION : "Déclaration de conformité pour les moteurs SNMP qui mettent en œuvre la SNMP-USER-BASED-SM-MIB. "

MODULE -- ce module
GROUPES OBLIGATOIRES : { usmMIBBasicGroup }

OBJET : usmUserAuthProtocol
MIN-ACCESS : lecture seule
DESCRIPTION : "L'accès en écriture n'est pas exigé."

OBJET : usmUserPrivProtocol
MIN-ACCESS : lecture seule

```
DESCRIPTION : "L'accès en écriture n'est pas exigé."
   ::= { usmMIBCompliances 1 }

-- Unités de conformité

GROUPE D'OBJETS  usmMIBBasicGroup
OBJETS : { usmStatsUnsupportedSecLevels, usmStatsNotInTimeWindows, usmStatsUnknownUserNames,
           usmStatsUnknownEngineIDs, usmStatsWrongDigests, usmStatsDecryptionErrors, usmUserSpinLock,
           usmUserSecurityName, usmUserCloneFrom, usmUserAuthProtocol, usmUserAuthKeyChange,
           usmUserOwnAuthKeyChange, usmUserPrivProtocol, usmUserPrivKeyChange, usmUserOwnPrivKeyChange,
           usmUserPublic, usmUserStorageType, usmUserStatus }
STATUT : actuel
DESCRIPTION : "Collection d'objets assurant la configuration d'un moteur SNMP qui met en œuvre le modèle de sécurité
                SNMP fondé sur l'utilisateur. "
   ::= { usmMIBGroups 1 }

FIN
```

6. Protocole d'authentification HMAC-MD5-96

Cette Section décrit le protocole d'authentification HMAC-MD5-96. Ce protocole d'authentification est le premier défini pour le modèle de sécurité fondé sur l'utilisateur. Il utilise la fonction de hachage MD5 qui est décrite dans la [RFC1321], en mode HMAC décrit dans la [RFC2104], en tronquant le résultat à 96 bits.

Ce protocole est identifié par usmHMACMD5AuthProtocol.

Avec le temps, d'autres protocoles d'authentification pourront être définis soit comme remplacement, soit en plus de ce protocole.

6.1 Mécanismes

- Pour la prise en charge de l'intégrité des données, un algorithme de résumé de message est requis. Un résumé est calculé sur une portion appropriée d'un message SNMP et inclus au titre du message envoyé au receveur.

- Pour la prise en charge de l'authentification de l'origine des données et de l'intégrité des données, une valeur secrète est ajoutée au message SNMP avant de calculer le résumé ; le résumé calculé est partiellement inséré dans le message SNMP avant transmission, et la valeur ajoutée n'est pas transmise. La valeur secrète est partagée par tous les moteurs SNMP autorisés à générer des messages au nom de l'usager approprié.

6.1.1 Mécanisme d'authentification par résumé

Le mécanisme d'authentification par résumé défini dans le présent mémoire assure :

- la vérification de l'intégrité d'un message reçu, c'est-à-dire, que le message reçu est le message envoyé.
  L'intégrité du message est protégée par le calcul d'un résumé sur une portion appropriée du message. Le résumé est calculé par le générateur du message, transmis avec le message, et vérifié par le receveur du message.
- la vérification de l'usager au nom duquel le message a été généré.
  Une valeur secrète connue des seuls moteurs SNMP autorisés à générer des messages au nom d'un utilisateur est utilisée en mode HMAC (voir la [RFC2104]). Il est aussi recommandé que le résultat de la fonction de hachage utilisé comme code d'authentification de message, soit tronqué.

Ce protocole utilise l'algorithme de résumé de message MD5 [RFC1321]. Un résumé MD5 de 128 bits est calculé d'une façon particulière (HMAC) sur la portion désignée d'un message SNMP et les 96 premiers bits de ce résumé sont inclus au titre du message envoyé au receveur. La taille du résumé porté dans un message est de 12 octets. La taille de la clé privée d'authentification (le secret) est de 16 octets. Pour les détails, voir au paragraphe 6.3.

6.2 Éléments du protocole d'authentification par résumé

Ce paragraphe contient les définitions requises pour réaliser le module d'authentification défini dans le présent mémoire.

6.2.1 Utilisateurs

L'authentification qui utilise ce protocole d'authentification fait usage d'un ensemble défini de userNames. Pour tout usager au nom duquel un message doit être authentifié à un moteur SNMP particulier, ce moteur SNMP doit avoir connaissance de cet usager. Un moteur SNMP qui souhaite communiquer avec un autre moteur SNMP doit aussi avoir connaissance d'un utilisateur connu de ce moteur, incluant la connaissance des attributs applicables de cet usager.

Un usager et ses attributs sont définis comme suit :
<userName> une chaîne qui représente le nom de l'usager.
<authKey> une clé secrète de l'usager à utiliser pour calculer un résumé. Il DOIT avoir 16 octets pour MD5.

6.2.2 msgAuthoritativeEngineID

La valeur msgAuthoritativeEngineID contenue dans un message authentifié spécifie le moteur SNMP d'autorité pour le message (voir la définition de SnmpEngineID dans le document d'architecture SNMP [RFC3411]).

La clé d'authentification (privée) de l'usager est normalement différente à chaque moteur SNMP d'autorité et donc le snmpEngineID est utilisé pour choisir la clé appropriée pour le processus d'authentification.

6.2.3 Messages SNMP utilisant ce protocole d'authentification

Les messages qui utilisent ce protocole d'authentification portent un champ msgAuthenticationParameters au titre des msgSecurityParameters. Pour ce protocole, le champ msgAuthenticationParameters est la CHAINE D'OCTETS mise en série qui représente les douze premiers octets du résultat de HMAC-MD5-96 appliqué sur le wholeMsg.

Le résumé est calculé sur le wholeMsg de sorte que si un message est authentifié, cela signifie aussi que tous les champs du message sont intacts et n'ont pas été altérés.

6.2.4 Services fournis par le module d'authentification HMAC-MD5-96

Ce paragraphe décrit les entrées et résultats que le module d'authentification HMAC-MD5-96 attend et produit lorsque le module de sécurité fondé sur l'utilisateur invoque les services du module d'authentification HMAC-MD5-96.

6.2.4.1 Services pour générer un message SNMP sortant

Le protocole d'authentification HMAC-MD5-96 suppose que le choix de la authKey est fait par l'appelant et que celui-ci passe la clé secrète à utiliser.

Lorsque c'est fait, le module d'authentification retourne les statusInformation et, si le résumé de message a été correctement calculé, le wholeMsg avec le résumé inséré à l'endroit approprié. La primitive de service abstraite est :

```
statusInformation = authenticateOutgoingMsg(    -- succès ou échec
   IN   authKey                                 -- clé secrète pour l'authentification
   IN   wholeMsg                                -- message complet non authentifié
   OUT  authenticatedWholeMsg                   -- message authentifié complet
   )
```

Les éléments de données abstraits sont :

statusInformation : indication de si le processus d'authentification a réussi. Sinon c'est une indication du problème.

authKey : la clé secrète à utiliser par l'algorithme d'authentification. La longueur de cette clé DOIT être 16 octets.

wholeMsg : le message à authentifier.

authenticatedWholeMsg : le message authentifié (incluant le résumé inséré) en résultat.

Note : le champ authParameters est rempli par le module d'authentification et ce module et ce champ devraient déjà être présents dans le wholeMsg avant la génération du code d'authentification de message (MAC).

6.2.4.2 Services pour traiter un message SNMP entrant

Le protocole d'authentification HMAC-MD5-96 suppose que le choix de la authKey est fait par l'appelant et que celui-ci passe la clé secrète à utiliser.

Lorsque c'est fait, le module d'authentification retourne les statusInformation et, si le résumé de message a été correctement calculé, le wholeMsg comme il a été traité. La primitive de service abstraite est :

```
statusInformation =  authenticateIncomingMsg(    -- succès ou échec
   IN   authKey                                  -- clé secrète pour l'authentification
   IN   authParameters                           -- comme reçu du réseau
   IN   wholeMsg                                 -- comme reçu du réseau
   OUT  authenticatedWholeMsg                    -- message authentifié complet
   )
```

Les éléments de données abstraits sont :
statusInformation : indication de si le processus d'authentification a réussi. Sinon, c'est l'indication du problème.
authKey : clé secrète à utiliser par l'algorithme d'authentification. La longueur de cette clé DOIT être 16 octets.
authParameters : les authParameters provenant du message entrant.
wholeMsg : le message à authentifier en entrée et le message authentifié en sortie.
authenticatedWholeMsg : le message entier après l'achèvement de la vérification de l'authentification.

6.3 Éléments de procédure

Ce paragraphe décrit les procédures pour le protocole d'authentification HMAC-MD5-96.

6.3.1 Traitement d'un message sortant

Ce paragraphe décrit la procédure suivie par un moteur SNMP chaque fois qu'il doit authentifier un message sortant en utilisant le usmHMACMD5AuthProtocol.

1) Le champ msgAuthenticationParameters est réglé à la mise en série, conformément aux règles de la [RFC3417], d'une CHAINE D'OCTETS contenant 12 octets à zéro
2) De la authKey secrète, deux clés K1 et K2 sont déduites :
   a) étendre la authKey à 64 octets en ajoutant 48 octets à zéro ; la sauvegarder comme extendedAuthKey
   b) obtenir IPAD en dupliquant l'octet 0x36 64 fois ;
   c) obtenir K1 en OUixant extendedAuthKey avec IPAD ;
   d) obtenir OPAD en dupliquant l'octet 0x5C 64 fois ;
   e) obtenir K2 en OUixant extendedAuthKey avec OPAD.
3) Ajouter K1 au wholeMsg et calculer le résumé MD5 sur lui conformément à la [RFC1321].
4) Ajouter K2 au résultat de l'étape 4 et calculer le résumé MD5 sur lui conformément à la [RFC1321]. Prendre les 12 premiers octets du résumé final – c'est le code d'authentification du message (MAC).
5) Remplacer le champ msgAuthenticationParameters par le MAC obtenu à l'étape 4.
6) Le authenticatedWholeMsg est alors retourné à l'appelant avec les statusInformation qui indiquent le succès.

6.3.2 Traitement d'un message entrant

Ce paragraphe décrit la procédure suivie par un moteur SNMP chaque fois qu'il doit authentifier un message entrant en utilisant le usmHMACMD5AuthProtocol.

1) Si le résumé reçu dans le champ msgAuthenticationParameters ne fait pas 12 octets, on retourne un échec et une errorIndication (authenticationError) au module appelant.
2) Le MAC reçu dans le champ msgAuthenticationParameters est sauvegardé.
3) Le résumé dans le champ msgAuthenticationParameters est remplacé par les 12 octets de zéro.
4) De la authKey secrète, deux clés K1 et K2 sont déduites :
   a) on étend la authKey à 64 octets en ajoutant 48 octets de zéros ; la sauvegarder comme extendedAuthKey ;
   b) obtenir IPAD en dupliquant l'octet 0x36 64 fois ;
   c) obtenir K1 en OUixant extendedAuthKey avec IPAD ;
   d) obtenir OPAD en dupliquant l'octet 0x5C 64 fois ;
   e) obtenir K2 en OUixant extendedAuthKey avec OPAD.
5) Le MAC est calculé sur le wholeMsg :
   a) ajouter K1 au wholeMsg et calculer le résumé MD5 sur lui ;

b) ajouter K2 au résultat de l'étape 5.a et calculer le résumé MD5 sur lui ;
c) les 12 premiers octets du résultat de l'étape 5.b sont le MAC.
Le champ msgAuthenticationParameters est remplacé par la valeur du MAC qui a été sauvegardé à l'étape 2.

6) Puis le MAC qu'on vient de calculer est comparé au MAC sauvegardé à l'étape 2. Si ils ne correspondent pas, un échec et une errorIndication (authenticationFailure) sont retournés au module appelant.
7) Le authenticatedWholeMsg et les statusInformation indiquant le succès sont alors retournés à l'appelant.

7. Protocole d'authentification HMAC-SHA-96

Cette Section décrit le protocole d'authentification HMAC-SHA-96. Ce protocole utilise la fonction de hachage SHA qui est décrite dans [SHA-NIST], en mode HMAC décrit dans la [RFC2104], en tronquant le résultat à 96 bits.

Ce protocole est identifié par usmHMACSHAAuthProtocol.

Avec le temps, d'autres protocoles d'authentification pourront être définis soit en remplacement, soit en plus de ce protocole.

7.1 Mécanismes

- Pour la prise en charge de l'intégrité des données, un algorithme de résumé de message est exigé. Un résumé est calculé sur une portion appropriée d'un message SNMP et incluse au titre du message envoyé au receveur.
- Pour la prise en charge de l'authentification de l'origine des données et de l'intégrité des données, une valeur secrète est ajoutée au message SNMP avant de calculer le résumé ; le résumé calculé est alors partiellement inséré dans le message avant sa transmission. Le secret ajouté n'est pas transmis. La valeur secrète est partagée par tous les moteurs SNMP autorisés à générer des messages au nom de l'usager approprié.

7.1.1 Mécanisme d'authentification par résumé

Le mécanisme d'authentification par résumé défini dans le présent mémoire assure :

- la vérification de l'intégrité d'un message reçu, c'est-à-dire que le message reçu est le message envoyé. L'intégrité du message est protégée par le calcul d'un résumé sur une portion appropriée du message. Le résumé est calculé par le générateur du message, transmis avec le message, et vérifié par le receveur du message.
- la vérification de l'usager au nom duquel le message a été généré. Une valeur secrète connue des seuls moteurs SNMP autorisés à générer des messages au nom d'un utilisateur est utilisée en mode HMAC (voir la [RFC2104]). Elle recommande aussi que le résultat de la fonction de hachage, utilisée comme code d'authentification de message, soit tronquée.

Ce mécanisme utilise l'algorithme de résumé de message SHA [SHA-NIST]. Un résumé SHA de 160 bits est calculé d'une façon particulière (HMAC) sur la portion désignée d'un message SNMP et les 96 premiers bits de ce résumé sont inclus au titre du message envoyé au receveur. La taille du résumé porté dans un message est de 12 octets. La taille de la clé d'authentification privée (le secret) est de 20 octets. Voir les détails au paragraphe 7.3.

7.2 Éléments du protocole d'authentification HMAC-SHA-96

Ce paragraphe contient les définitions requises pour réaliser le module d'authentification défini dans le présent mémoire.

7.2.1 Utilisateurs

L'authentification faite avec ce protocole d'authentification utilise un ensemble défini de userNames. Pour tout usager au nom duquel un message doit être authentifié à un moteur SNMP particulier, ce moteur SNMP doit avoir connaissance de cet usager. Un moteur SNMP qui souhaite communiquer avec un autre moteur SNMP doit aussi avoir connaissance d'un utilisateur connu de ce moteur, incluant la connaissance des attributs applicables de cet usager.

Un usager et ses attributs sont définis comme suit :
<userName> : chaîne qui représente le nom de l'usager.
<authKey> : clé secrète de l'usager à utiliser lors du calcul d'un résumé. Elle DOIT être de 20 octets pour SHA.

7.2.2 msgAuthoritativeEngineID

La valeur msgAuthoritativeEngineID contenue dans un message authentifié spécifie le moteur SNMP d'autorité pour ce message particulier (voir la définition de SnmpEngineID dans le document d'architecture SNMP [RFC3411]).

La clé d'authentification (privée) de l'usager est normalement différente pour chaque moteur SNMP d'autorité et donc le snmpEngineID est utilisé pour choisir la clé appropriée pour le processus d'authentification.

7.2.3 Messages SNMP qui utilisent ce protocole d'authentification

Les messages qui utilisent ce protocole d'authentification portent un champ msgAuthenticationParameters au titre des msgSecurityParameters. Pour ce protocole, le champ msgAuthenticationParameters est la CHAINE D'OCTETS mise en série qui représente les douze premiers octets du résultat du HMAC-SHA-96 effectué sur le wholeMsg.

Le résumé est calculé sur le wholeMsg de sorte que si un message est authentifié, cela signifie aussi que tous les champs du message sont intacts et n'ont pas été altérés.

7.2.4 Services fournis par le module d'authentification HMAC-SHA-96

Ce paragraphe décrit les entrées et sorties que le module d'authentification HMAC-SHA-96 attend et produit lorsque le module de sécurité fondé sur l'utilisateur invoque les services du module d'authentification HMAC-SHA-96.

7.2.4.1 Services pour générer un message SNMP sortant

Le protocole d'authentification HMAC-SHA-96 suppose que le choix de la authKey est fait par l'appelant et que celui-ci passe la clé secrète à utiliser.

Lorsque c'est fait, le module d'authentification retourne les statusInformation et, si le résumé de message a été correctement calculé, le wholeMsg avec le résumé inséré à l'endroit approprié. La primitive de service abstraite est :

```
statusInformation =  authenticateOutgoingMsg(    -- succès ou échec
   IN   authKey                                  -- clé secrète pour l'authentification
   IN   wholeMsg                                 -- message complet non authentifié
   OUT  authenticatedWholeMsg                    -- message authentifié complet
   )
```

Les éléments de données abstraits sont :
statusInformation : indiquent si le processus d'authentification a réussi. Sinon, c'est une indication du problème.
authKey : clé secrète à utiliser par l'algorithme d'authentification. La longueur de cette clé DOIT être 20 octets.
wholeMsg : le message à authentifier.
authenticatedWholeMsg : le message authentifié (incluant le résumé inséré) en sortie.

Noter que le champ authParameters est rempli par le module d'authentification et ce champ devrait être déjà présent dans le wholeMsg avant que soit généré le code d'authentification de message (MAC).

7.2.4.2 Services pour traiter un message SNMP entrant

Le protocole d'authentification HMAC-SHA-96 suppose que le choix de la authKey est fait par l'appelant et que celui-ci passe la clé secrète à utiliser.

Lorsque c'est fait, le module d'authentification retourne les statusInformation et, si le résumé de message a été correctement calculé, le wholeMsg comme il a été traité. La primitive de service abstraite est :

```
statusInformation =   authenticateIncomingMsg(    -- succès ou échec
  IN   authKey                                    -- clé secrète pour l'authentification
  IN   authParameters                             -- comme reçu du réseau
  IN   wholeMsg                                   -- comme reçu du réseau
 OUT  authenticatedWholeMsg                       -- message authentifié complet
  )
```

Les éléments de données abstraits sont :
statusInformation : indiquent si le processus d'authentification a réussi. Sinon, c'est l'indication du problème.

authKey : la clé secrète à utiliser par l'algorithme d'authentification. La longueur de cette clé DOIT être 20 octets.
authParameters : les authParameters provenant du message entrant.
wholeMsg : le message à authentifier en entrée et le message authentifié en sortie.
authenticatedWholeMsg : le message entier après l'achèvement de la vérification d'authentification.

7.3 Éléments de procédure

Ce paragraphe décrit les procédures pour le protocole d'authentification HMAC-SHA-96.

7.3.1 Traitement d'un message sortant

Ce paragraphe décrit la procédure suivie par un moteur SNMP chaque fois qu'il doit authentifier un message sortant en utilisant le usmHMACSHAAuthProtocol.

1) Le champ msgAuthenticationParameters est réglé à une CHAINE D'OCTETS mise en série, conformément aux règles de la [RFC3417], contenant 12 octets à zéro.
2) Deux clés K1 et K2 sont déduites de la authKey secrète :
   a) extension de authKey à 64 octets par l'ajout de 44 octets de zéros ; sauvegardée comme extendedAuthKey ;
   b) obtenir IPAD en dupliquant l'octet 0x36 64 fois ;
   c) obtenir K1 en OUixant extendedAuthKey avec IPAD ;
   d) obtenir OPAD en dupliquant l'octet 0x5C 64 fois ;
   e) obtenir K2 en OUixant extendedAuthKey avec OPAD.
3) Ajouter K1 au wholeMsg et calculer le résumé SHA sur lui conformément à [SHA-NIST].
4) Ajouter K2 au résultat de l'étape 4 et calculer le résumé SHA sur lui conformément à [SHA-NIST]. Prendre les 12 premiers octets du résumé final – c'est le code d'authentification de message (MAC).
5) Remplacer le champ msgAuthenticationParameters par le MAC obtenu à l'étape 4.
6) Le authenticatedWholeMsg est alors retourné à l'appelant avec les statusInformation qui indiquent le succès.

7.3.2 Traitement d'un message entrant

Ce paragraphe décrit la procédure suivie par un moteur SNMP chaque fois qu'il doit authentifier un message entrant en utilisant le usmHMACSHAAuthProtocol.

1) Si le résumé reçu dans le champ msgAuthenticationParameters ne fait pas 12 octets, c'est un échec et une errorIndication (authenticationError) est retournée au module appelant.
2) Le MAC reçu dans le champ msgAuthenticationParameters est sauvegardé.
3) Le résumé dans le champ msgAuthenticationParameters est remplacé par les 12 octets de zéros.
4) Deux clés K1 et K2 sont déduites de la authKey secrète :
   a) extension de authKey à 64 octets par l'ajout de 44 octets de zéros ; sauvegardée comme extendedAuthKey ;
   b) obtenir IPAD en dupliquant l'octet 0x36 64 fois ;
   c) obtenir K1 en OUixant extendedAuthKey avec IPAD ;
   d) obtenir OPAD en dupliquant l'octet 0x5C 64 fois ;
   e) obtenir K2 en OUixant extendedAuthKey avec OPAD.
5) Le MAC est calculé sur le wholeMsg :
   a) ajouter K1 au wholeMsg et calculer le résumé SHA sur lui ;
   b) ajouter K2 au résultat de l'étape 5.a et calculer le résumé SHA sur lui ;
   c) les 12 premiers octets du résultat de l'étape 5.b sont le MAC.
   Le champ msgAuthenticationParameters est remplacé par la valeur du MAC qui a été sauvegardée à l'étape 2.
6) Le nouveau MAC calculé est comparé au MAC sauvegardé à l'étape 2. Si ils ne correspondent pas, c'est un échec et une errorIndication (authenticationFailure) est retournée au module appelant.
7) Le authenticatedWholeMsg et les statusInformation qui indiquent le succès sont alors retournés à l'appelant.

8. Protocole CBC-DES de chiffrement symétrique

Cette Section décrit le protocole de chiffrement symétrique CBC-DES. Ce protocole est le premier protocole de confidentialité défini pour le modèle de sécurité fondé sur l'utilisateur.

Ce protocole est identifié par usmDESPrivProtocol. Avec le temps, d'autres protocoles de confidentialité pourraient être définis soit en remplacement, soit en complément de ce protocole.

8.1 Mécanismes

- Pour prendre en charge la confidentialité des données, un algorithme de chiffrement est nécessaire. Une portion appropriée du message est chiffrée avant qu'il soit transmis. Le modèle de sécurité fondé sur l'utilisateur spécifie que la scopedPDU est la portion du message qui doit être chiffrée.
- Une valeur secrète en combinaison avec une valeur de délai est utilisée pour créer la clé de chiffrement/déchiffrement et la valeur d'initialisation. La valeur secrète est partagée par tous les moteurs SNMP autorisés à générer des messages au nom de l'usager approprié.

8.1.1 Protocole de chiffrement symétrique

Le protocole de chiffrement symétrique défini dans le présent mémoire fournit la prise en charge de la confidentialité des données. La portion désignée d'un message SNMP est chiffrée et incluse comme partie du message envoyé au receveur.

Deux organisations ont publié des spécifications qui définissent le DES : l'Institut National des Normes et Technologies (NIST) [DES-NIST] et l'Institut Américain de normalisation [DES-ANSI]. Il y a une spécification d'accompagnement sur les modes de fonctionnement pour chaque définition ([DESO-NIST] et [DESO-ANSI], respectivement).

Le NIST a publié trois documents supplémentaires que les développeurs pourraient trouver utiles.
- Il y a un document de lignes directrices pour la mise en œuvre et l'utilisation du DES, incluant des spécifications fonctionnelles pour le DES et ses modes de fonctionnement [DESG-NIST].
- Il y a une spécification d'une suite d'essais de validation pour le [DEST-NIST]. La suite est conçue pour essayer tous les aspects du DES et elle est utile pour épingler les problèmes spécifiques.
- Il y a une spécification d'un essai de maintenance pour le DES [DESM-NIST]. L'essai utilise une quantité minimale de données et de traitement pour tester tous les composants du DES. Elle fournit une simple indication binaire de fonctionnement direct et est utile au titre des étapes d'initialisation, par exemple lorsque un ordinateur se réamorce.

8.1.1.1 Clé DES et valeur d'initialisation

Les huit premiers octets des 16 octets du secret (clé privée de confidentialité) sont utilisés comme clé DES. Comme DES utilise seulement 56 bits, le bit de moindre poids de chaque octet n'est pas pris en considération.

La valeur d'initialisation (IV) pour le chiffrement est obtenue en respectant la procédure suivante .

Les huit derniers octets des 16 octets du secret (clé privée de confidentialité) sont utilisés comme pre-IV.

Afin d'assurer que l'IV pour deux paquets différents chiffrés par la même clé ne sont pas les mêmes (c'est-à-dire, que l'IV ne se répète pas) on a besoin de "saler" la pré IV avec quelque chose d'unique par paquet. Une chaîne de 8 octets est utilisée comme "sel". L'enchaînement des 32 bits du snmpEngineBoots du moteur SNMP générateur et d'un entier local de 32 bits, que le moteur de chiffrement entretient, est entré dans le "sel". L'entier de 32 bits est initialisé à une valeur arbitraire au moment de l'amorçage.

Les 32 bits de snmpEngineBoots sont convertis dans les quatre premiers octets (octet de poids fort en premier) de notre "sel". L'entier de 32 bits est ensuite converti dans les quatre derniers octets (octet de poids fort en premier) de notre "sel". Le "sel" résultant est ensuite OU-ixé avec la pré IV pour obtenir l'IV. Le "sel" de 8 octets est mis ensuite dans le champ privParameters codé comme une CHAINE D'OCTETS. Le "sel" entier est alors modifié. On recommande qu'il soit incrémenté de un et revienne à zéro lorsque il atteint la valeur maximum.

Comment varie exactement la valeur du "sel" (et donc de l'IV) est une question de mise en œuvre, pour autant que des mesures soient prises pour éviter de produire une duplication d'IV.

Le "sel" doit être placé dans le champ privParameters pour permettre à l'entité receveuse de calculer l'IV correcte et déchiffrer le message.

8.1.1.2 Chiffrement des données

Les données à chiffrer sont traitées comme une séquence d'octets. Sa longueur devrait être un entier multiple de 8 - et si elle ne l'est pas, les données seront bourrées en fin autant que nécessaire. La valeur réelle du bourrage est sans importance.

Les données sont chiffrées en mode de chaînage de bloc de chiffrement (CBC, *Cipher Block Chaining*).

Le teste source (plaintext) est divisé en blocs de 64 bits.

Chaque bloc du texte source est OUixé avec le texte chiffré (ciphertext) du bloc précédent, le résultat est chiffré et le résultat du chiffrement est le texte chiffré pour le bloc. Cette procédure est répétée jusqu'à ce qu'il n'y ait plus de bloc de texte source.

Pour le tout premier bloc, la valeur d'initialisation est utilisée à la place du texte chiffré du bloc précédent.

8.1.1.3 Déchiffrement des données

Avant le déchiffrement, la longueur des données chiffrées est vérifiée. Si la longueur de la CHAINE D'OCTETS à déchiffrer n'est pas un multiple entier de 8 octets, le processus de déchiffrement est arrêté et une exception appropriée est notée. Lors du déchiffrement, le bourrage est ignoré.

Le premier bloc de texte chiffré est déchiffré, le résultat du déchiffrement est OUixé avec la valeur d'initialisation, et le résultat est le premier bloc de texte source.

Pour chaque bloc suivant, le bloc de texte chiffré est déchiffré, le résultat du déchiffrement est OUixé avec le bloc de texte chiffré précédent et le résultat est le bloc de texte source.

8.2 Éléments du protocole de confidentialité de DES

Ce paragraphe contient les définitions nécessaires pour réaliser le module de confidentialité défini dans le présent mémoire.

8.2.1 Utilisateurs

Le chiffrement/déchiffrement des données qui utilise le protocole de chiffrement symétrique emploie un ensemble défini de userNames. Pour tout utilisateur au nom duquel un message doit être chiffré/déchiffré à un certain moteur SNMP, ce moteur SNMP doit avoir connaissance de cet utilisateur. Un moteur SNMP qui souhaite communiquer avec un autre moteur SNMP doit aussi avoir connaissance d'un utilisateur connu de ce moteur SNMP, incluant la connaissance des attributs applicables de cet usager.

Un usager et ses attributs sont définis comme suit :
<userName> : chaîne d'octets qui représente le nom de l'usager.
<privKey> : clé secrète d'un usager à utiliser comme entrée pour la clé et l'IV DES. La longueur de cette clé DOIT être de 16 octets.

8.2.2 msgAuthoritativeEngineID

La valeur msgAuthoritativeEngineID contenue dans un message authentifié spécifie le moteur SNMP d'autorité pour ce message particulier (voir la définition de SnmpEngineID dans le document "Architecture SNMP" [RFC3411]).

La clé de confidentialité (privée) de l'usager est normalement différente pour chaque moteur SNMP d'autorité et donc le snmpEngineID est utilisé pour choisir la clé appropriée pour le processus de chiffrement/déchiffrement.

8.2.3 Messages SNMP qui utilisent ce protocole de confidentialité

Les messages qui utilisent ce protocole de confidentialité portent un champ msgPrivacyParameters au titre des msgSecurityParameters. Pour ce protocole, le champ msgPrivacyParameters est la CHAINE D'OCTETS mise en série qui représente le "sel" qui a été utilisé pour créer la IV.

8.2.4 Services fournis par le module de confidentialité DES

Ce paragraphe décrit les entrées et les résultats qu'attend le module de confidentialité DES et qu'il produit lorsque le module de sécurité fondé sur l'utilisateur invoque les services du module de confidentialité DES.

8.2.4.1 Services pour chiffrer les données sortantes

Le protocole de confidentialité DES suppose que le choix de la privKey est fait par l'appelant et que celui-ci passe la clé

secrète à utiliser.

Lorsque c'est fait, le module de confidentialité retourne les statusInformation et, si le processus de chiffrement a réussi, la encryptedPDU et les msgPrivacyParameters codés comme CHAINE D'OCTETS. La primitive de service abstraite est :

```
statusInformation =   encryptData(    -- succès ou échec
    IN    encryptKey                  -- clé secrète pour chiffrement
    IN    dataToEncrypt               -- données à chiffrer (scopedPDU)
    OUT   encryptedData               -- données chiffrées (encryptedPDU)
    OUT   privParameters              -- rempli par le fournisseur de
                                      service
     )
```

Les éléments de données abstraits sont :
statusInformation : indication du succès ou échec du processus de chiffrement. En cas d'échec, cela indique l'erreur.
encryptKey : clé secrète à utiliser par l'algorithme de chiffrement. La longueur de cette clé DOIT être 16 octets.
dataToEncrypt : les données à chiffrer.
encryptedData : les données chiffrées après achèvement réussi.
privParameters : les paramètres de confidentialité codés comme CHAINE D'OCTETS.

8.2.4.2 Services pour déchiffrer les données entrantes

Ce protocole de confidentialité DES suppose que le choix de la privKey est fait par l'appelant et que celui-ci passe la clé secrète à utiliser.

Lorsque c'est fait, le module de confidentialité retourne les statusInformation et, si le processus de déchiffrement a réussi, la scopedPDU en texte source. La primitive de service abstraite est :

```
statusInformation = decryptData(
    IN    decryptKey          clé secrète pour le déchiffrement
    IN    privParameters      comme reçu du réseau
    IN    encryptedData       données chiffrées (encryptedPDU)
   OUT    decryptedData       donnée déchiffrées (scopedPDU)
     )
```

Les éléments de données abstraits sont :
statusInformation : indiquent si les données ont été déchiffrées avec succès et sinon l'indication de l'erreur.
decryptKey : clé secrète à utiliser par l'algorithme de déchiffrement. La longueur de cette clé DOIT être 16 octets.
privParameters : le "sel" à utiliser pour calculer l'IV.
encryptedData : données à déchiffrer.
decryptedData : données déchiffrées.

8.3 Éléments de procédure

Ce paragraphe décrit les procédures pour le protocole de confidentialité DES.

8.3.1 Traitement d'un message sortant

Ce paragraphe décrit la procédure suivie par un moteur SNMP chaque fois qu'il doit chiffrer une partie d'un message sortant en utilisant le usmDESPrivProtocol.

1) La cryptKey secrète est utilisée pour construire la clé de chiffrement DES, le "sel" et la pré IV DES (à partir de laquelle est calculée la IV comme décrit au paragraphe 8.1.1.1).

2) Le champ privParameters est réglé à la mise en série conformément aux règles de la [RFC3417] d'une CHAINE D'OCTETS représentant la chaîne "sel".

3) La scopedPDU est chiffrée (comme décrit au paragraphe 8.1.1.2) et les données chiffrées sont mises en série conformément aux règles de la [RFC3417] comme CHAINE D'OCTETS.

4) La CHAINE D'OCTETS mise en série qui représente la scopedPDU chiffrée avec les privParameters et les

statusInformation indiquant le succès sont retournées au module appelant.

8.3.2 Traitement d'un message entrant

Ce paragraphe décrit la procédure suivie par un moteur SNMP chaque fois qu'il doit déchiffrer une partie d'un message entrant en utilisant le usmDESPrivProtocol.

1) Si le champ privParameters n'est pas une CHAINE D'OCTETS de huit octets, une indication d'erreur(decryptionError) est retournée au module appelant.

2) Le "sel" est extrait du champ privParameters.

3) La cryptKey secrète et le "sel" sont alors utilisés pour construire la clé de déchiffrement DES et la pré IV (à partir de laquelle la IV est calculée comme décrit au paragraphe 8.1.1.1).

4) La encryptedPDU est ensuite déchiffrée (comme décrit au paragraphe 8.1.1.3).

5) Si la encryptedPDU ne peut pas être déchiffrée, une indication d'erreur (decryptionError) est retournée au module appelant.

6) La scopedPDU déchiffrée et les statusInformation indiquant le succès sont retournées au module appelant.

9. Propriété intellectuelle

L'IETF ne prend pas position sur la validité et la portée de tout droit de propriété intellectuelle ou autres droits qui pourrait être revendiqués au titre de la mise en œuvre ou l'utilisation de la technologie décrite dans le présent document ou sur la mesure dans laquelle toute licence sur de tels droits pourrait être ou n'être pas disponible ; pas plus qu'elle ne prétend avoir accompli aucun effort pour identifier de tels droits. Les informations sur les procédures de l'ISOC au sujet des droits dans les documents de l'ISOC figurent dans les BCP 78 et BCP 79.

Des copies des dépôts d'IPR faites au secrétariat de l'IETF et toutes assurances de disponibilité de licences, ou le résultat de tentatives faites pour obtenir une licence ou permission générale d'utilisation de tels droits de propriété par ceux qui mettent en œuvre ou utilisent la présente spécification peuvent être obtenues sur répertoire en ligne des IPR de l'IETF à http://www.ietf.org/ipr.

L'IETF invite toute partie intéressée à porter son attention sur tous copyrights, licences ou applications de licence, ou autres droits de propriété qui pourraient couvrir les technologies qui peuvent être nécessaires pour mettre en œuvre la présente norme. Prière d'adresser les informations à l'IETF à ietf-ipr@ietf.org.

10. Remerciements

Le présent document est le résultat des efforts du groupe de travail SNMPv3. Des remerciements particuliers sont adressés par ordre alphabétique aux membres suivants du GT SNMPv3 : Harald Tveit Alvestrand (Maxware), Dave Battle (SNMP Research, Inc.), Alan Beard (Disney Worldwide Services), Paul Berrevoets (SWI Systemware/Halcyon Inc.), Martin Bjorklund (Ericsson), Uri Blumenthal (IBM T.J. Watson Research Center), Jeff Case (SNMP Research, Inc.), John Curran (BBN), Mike Daniele (Compaq Computer Corporation), T. Max Devlin (Eltrax Systems), John Flick (Hewlett Packard), Rob Frye (MCI), Wes Hardaker (U.C.Davis, Information Technology - D.C.A.S.), David Harrington (Cabletron Systems Inc.), Lauren Heintz (BMC Software, Inc.), N.C. Hien (IBM T.J. Watson Research Center), Michael Kirkham (InterWorking Labs, Inc.), Dave Levi (SNMP Research, Inc.), Louis A Mamakos (UUNET Technologies Inc.), Joe Marzot (Nortel Networks), Paul Meyer (Secure Computing Corporation), Keith McCloghrie (Cisco Systems), Bob Moore (IBM), Russ Mundy (TIS Labs at Network Associates), Bob Natale (ACE*COMM Corporation), Mike O'Dell (UUNET Technologies Inc.), Dave Perkins (DeskTalk), Peter Polkinghorne (Brunel University), Randy Presuhn (BMC Software, Inc.), David Reeder (TIS Labs at Network Associates), David Reid (SNMP Research, Inc.), Aleksey Romanov (Quality Quorum), Shawn Routhier (Epilogue), Juergen Schoenwaelder (TU Braunschweig), Bob Stewart (Cisco Systems), Mike Thatcher (Independent Consultant), Bert Wijnen (IBM T.J. Watson Research Center).

Le document se fonde sur les recommandations de l'équipe conseil Évolution du cadre administratif et de sécurité pour SNMP de l'IETF. Les membres de cette équipe conseil étaient : David Harrington (Cabletron Systems Inc.), Jeff Johnson (Cisco Systems), David Levi (SNMP Research Inc.), John Linn (Openvision), Russ Mundy (Trusted Information Systems)

président, Shawn Routhier (Epilogue), Glenn Waters (Nortel), Bert Wijnen (IBM T. J. Watson Research Center).

Comme recommandé par l'équipe conseil et la charte du groupe de travail SNMPv3, la conception a incorporé autant que faire s'est pu des précédentes RFC et projets. Il en résulte que des remerciement particuliers sont dus aux auteurs des projets précédents connus sous les noms de SNMPv2u et de SNMPv2* : Jeff Case (SNMP Research, Inc.), David Harrington (Cabletron Systems Inc.), David Levi (SNMP Research, Inc.), Keith McCloghrie (Cisco Systems), Brian O'Keefe (Hewlett Packard), Marshall T. Rose (Dover Beach Consulting), Jon Saperia (BGS Systems Inc.), Steve Waldbusser (International Network Services), Glenn W. Waters (Bell-Northern Research Ltd.).

11. Considérations pour la sécurité

11.1 Pratiques recommandées

Cette Section décrit les pratiques qui contribuent au fonctionnement sûr et efficace des mécanismes définis dans le présent mémoire.

- Un moteur SNMP doit éliminer les messages Response SNMP qui ne correspondent à aucun message de demande actuellement en instance. Il est de la responsabilité du module de traitement de message de veiller à cela. Par exemple, il peut utiliser un msgID pour cela.
  Une application de générateur de commandes SNMP doit éliminer toute PDU de classe Response pour laquelle il n'y a pas actuellement de PDU de classe Confirmée en instance ; par exemple, pour les PDU SNMPv2 [RFC3416], le composant request-id dans la PDU peut être utilisé pour corréler les réponses aux demandes en instance.
  Bien qu'il soit normal pour un moteur SNMP et une application SNMP de générateur de commandes de faire cela de façon naturelle, lorsque on utilise ces protocoles de sécurité, cela devient significatif à cause de la possibilité de duplication de messages (malveillante ou autre).

- Si un moteur SNMP utilise un msgID pour corréler les messages de réponse aux messages de demande en instance, il DOIT alors utiliser des msgID différents dans tous ces messages de demande qu'il envoie durant une fenêtre horaire (150 secondes).
  Une application de générateur de commandes ou de notifications DOIT utiliser des identifiants de demande différents dans toutes les PDU Request qu'elle envoie durant une fenêtre horaire (150 secondes).
  Ceci doit être fait pour protéger de la possibilité de duplication de messages (malveillante ou autre).
  Par exemple, commencer les opérations avec une valeur de msgID et/ou de request-id de zéro n'est pas une bonne idée. Les initialiser avec un nombre imprévisible (afin qu'elles ne démarrent pas avec le même après chaque réamorçage) et les incrémenter ensuite de un serait acceptable.

- Un moteur SNMP devrait effectuer la synchronisation horaire en utilisant des messages authentifiés afin de protéger de la possibilité de duplication de messages (malveillante ou autre).

- Lors de l'envoi de messages qui altèrent l'état à un moteur SNMP d'autorité géré, une application de générateur de commandes devrait retarder l'envoi des messages successifs à ce moteur SNMP géré jusqu'à ce qu'un accusé de réception positif soit reçu pour le message précédent ou jusqu'à ce que le message précédent arrive à expiration.
  Aucun ordre des messages n'est imposé par SNMP. Les messages peuvent être reçus dans n'importe quel ordre par rapport à l'heure de leur génération et chacun sera traité dans l'ordre de réception. Noter que lorsque un message authentifié est envoyé à un moteur SNMP géré, il sera valide pendant une durée approximative de 150 secondes dans des circonstances normales, et il est sujet à répétition durant cette période. Bien sûr, un moteur SNMP et les applications de générateur de commandes SNMP doivent compter avec les pertes et le réarrangement des messages qui résultent des anomalies du réseau.
  Cependant, un objet géré, snmpSetSerialNo [RFC3418], est spécifiquement défini pour être utilisé avec les opérations SET SNMP afin de fournir un mécanisme qui assure que le traitement des messages SNMP se fait dans un ordre spécifique.

- La fréquence avec laquelle les secrets d'un utilisateur du modèle de sécurité fondé sur l'utilisateur devraient être changés est indirectement en rapport avec la fréquence de leur utilisation.
  Protéger les secrets de la divulgation est critique pour la sécurité globale des protocoles. Une utilisation fréquente d'un secret fournit une source continuelle de données qui peut être utile à un cryptanalyste qui exploite les faiblesse connues ou perçues dans un algorithme. Un fréquent changement du secret évite cette vulnérabilité. Changer un secret après chaque usage est généralement considéré comme la pratique la plus sûre, mais une quantité significative de frais de gestion peut être associée à cette approche.

Noter aussi que dans un environnement local, la menace de divulgation peut être moins significative, et que le changement

des secrets peut donc être moins fréquent. Cependant, lorsque des réseaux publics de données sont utilisés comme chemins de communication, il est prudent de prendre plus de précautions.

11.2 Définir les utilisateurs

Les mécanismes définis dans le présent document emploient la notion d'usagers au nom desquels les messages sont envoyés. Comment les "usagers" sont définis est soumis à la politique de sécurité de l'administration du réseau. Par exemple, les usagers pourraient être des individus (par exemple, "joe" ou "jane"), ou un rôle particulier (par exemple, "opérateur" ou "administrateur") ou une combinaison (par exemple, "joe-opérateur", "jane-opérateur" ou "joe-admin"). De plus, un utilisateur peut être une entité logique, comme une application SNMP ou en ensemble d'applications SNMP, agissant au nom d'un individu ou rôle, ou un ensemble d'individus, ou un ensemble de rôles, incluant des combinaisons.

L'Appendice A décrit un algorithme pour transposer un "mot de passe" d'utilisateur en une valeur de 16/20 octets à utiliser comme clé d'authentification d'un utilisateur ou comme clé de confidentialité (ou les deux). Noter cependant, que l'utilisation du même mot de passe (et donc de la même clé) pour l'authentification et la confidentialité est une très mauvaise pratique de sécurité et devrait être fortement déconseillée. Les mots de passe sont souvent générés, mémorisés et entrés par une personne. Les mots de passe générés par des personnes peuvent faire moins que les 16/20 octets exiges par les protocoles d'authentification et de confidentialité, et les attaques en force brute peuvent être assez faciles sur un jeu de caractères ASCII relativement court. Donc, l'algorithme de l'Appendice A effectue une transformation sur le mot de passe. Si l'algorithme de l'Appendice A est utilisé, les mises en œuvre de SNMP (et les applications de configuration SNMP) doivent s'assurer que les mots de passe font au moins 8 caractères de long. Prière de noter que des mots de passe plus longs avec des chaînes répétitives peuvent résulter en exactement la même clé. Par exemple, un mot de passe 'bertbert' va résulter en exactement la même clé que le mot de passe 'bertbertbert'.

Parce que l'algorithme de l'Appendice A utilise (presque) directement de tels mots de passe, il est très important qu'on ne puisse les deviner facilement. Il est suggéré qu'ils soient composés de caractères alphanumériques et de ponctuation de casse mixte qui ne forment pas de mots ou de phrases qu'on pourrait trouver dans un dictionnaire. Les mots de passe plus longs améliorent la sécurité du système. Les utilisateurs peuvent souhaiter entrer des phrases de plusieurs mots pour rendre leur chaîne de mot de passe plus longues tout en s'assurant qu'elle est mémorisable.

Comme il est impossible aux utilisateurs humains de se souvenir des différents mots de passe pour chaque moteur SNMP, mais que les exigences de sécurité déconseillent fortement d'avoir la même clé pour plus d'un moteur SNMP, le modèle de sécurité fondé sur l'utilisateur emploie un compromis proposé dans [Localized-key]. Il déduit les clés d'usager pour les moteurs SNMP à partir d'un mot de passe d'utilisateur de telle façon qu'il est pratiquement impossible ni de déterminer le mot de passe de l'usager, ni la clé de l'usager pour un autre moteur SNMP à partir de toute combinaison des clés d'usager sur les moteurs SNMP.

Noter cependant, que si le mot de passe de l'usager est divulgué, la localisation de clé ne servira à rien et la sécurité du réseau peut être compromise dans ce cas. Donc le mot de passe d'un utilisateur ou une clé non localisée NE DOIT PAS être mémorisé sur un appareil/nœud géré. À la place, la clé localisée DEVRA être mémorisée (si elle l'est) afin que, en cas de compromission d'un appareil, aucun autre appareil géré ou gérant ne soit compromis à son tour.

11.3 Conformité

Pour être appelée une "mise en œuvre SNMP sûre" sur la base du modèle de sécurité fondé sur l'utilisateur, une mise en œuvre SNMP DOIT :

- mettre en œuvre un ou plusieurs protocoles d'authentification. Les protocoles d'authentification HMAC-MD5-96 et HMAC-SHA-96 définis dans le présent mémoire sont des exemples de tels protocoles.
- Dans la mesure maximum du possible, interdire l'accès aux secrets de chaque usager sur lequel elle détient des informations dans un magasin de données de configuration locale (LCD) dans toutes les circonstances sauf comme exigé pour générer et/ou valider les messages SNMP à l'égard de cet usager.
- mettre en œuvre le mécanisme de localisation de clé.
- mettre en œuvre la MIB SNMP-USER-BASED-SM-MIB.

De plus, un moteur SNMP d'autorité DEVRAIT fournir la configuration initiale conformément à l'Appendice A.1.

La mise en œuvre d'un protocole de confidentialité (le protocole de chiffrement symétrique DES défini dans le présent mémoire est un de ces protocoles) est facultative.

11.4 Utilisation des rapports

L'utilisation de rapports non sûrs (c'est-à-dire, leur envoi avec un niveau de sécurité de noAuthNoPriv) expose potentiellement un moteur SNMP non d'autorité à diverses formes d'attaques. Certains craignent ces attaques de déni de service, d'autres pas. Une installation devrait évaluer le risque impliqué avant de déployer des PDU Report non sûres.

11.5 Accès à la MIB SNMP-USER-BASED-SM

Les objets de cette MIB peuvent être considérés comme sensibles dans de nombreux environnements. Précisément, les objets du tableau usmUserTable contiennent des informations sur les usagers et leurs protocoles d'authentification et de confidentialité. Il est important de contrôler étroitement l'accès (à la fois en lecture et en écriture) à ces objets de MIB en utilisant des modèles de contrôle d'accès configurés de façon appropriée (par exemple le modèle de contrôle d'accès fondé sur la vue spécifié dans la [RFC3415]).

12. Références

12.1 Références normatives

[RFC1321] R. Rivest, "Algorithme de résumé de message MD5", avril 1992. *(Information)*

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

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

[RFC2578] K. McCloghrie, D. Perkins, J. Schoenwaelder, "Structure des informations de gestion, version 2 (SMIv2)", avril 1999. (STD0058)

[RFC2579] K. McCloghrie, D. Perkins, J. Schoenwaelder, "Conventions textuelles pour SMIv2", avril 1999. (STD0058)

[RFC2580] K. McCloghrie, D. Perkins, J. Schoenwaelder, "Déclarations de conformité pour SMIv2", avril 1999. (STD0058)

[RFC3411] D. Harrington, R. Presuhn, B. Wijnen, "Architecture de description des cadres de gestion du protocole simple de gestion de réseau (SNMP)", décembre 2002. (*MàJ par* RFC5343) (STD0062)

[RFC3412] J. Case et autres, "Traitement et distribution de message pour le protocole simple de gestion de réseau (SNMP)", décembre 2002. (STD0062)

[RFC3415] B. Wijnen, R. Presuhn, K. McCloghrie, "Modèle de contrôle d'accès fondé sur la vue (VACM) pour le protocole simple de gestion de réseau (SNMP)", décembre 2002. (STD0062)

[RFC3416] R. Presuhn, éd., "Version 2 des opérations de protocole pour le protocole simple de gestion de réseau (SNMP)", décembre 2002. (STD0062)

[RFC3417] R. Presuhn, éd. "Transpositions de transport pour le protocole simple de gestion de réseau (SNMP)", décembre 2002. (*MàJ par* RFC4789) (STD0062)

[RFC3418] R. Presuhn, éd., "Base de données d'informations de gestion (MIB) pour le protocole simple de gestion de réseau (SNMP)", décembre 2002. (STD0062)

[DES-NIST] National Institute of Standards and Technology. "Data Encryption Standard", Federal Information Processing Standard (FIPS) Publication 46-1. Remplace la publication FIPS 46, (janvier 1977; republiée en janvier 1988).

[DESO-NIST] National Institute of Standards and Technology. "DES Modes of Operation", Federal Information Processing Standard (FIPS) Publication 81, (décembre 1980).

[SHA-NIST] NIST FIPS 180-1, "Secure Hash Algorithm". (avril 1995) http://csrc.nist.gov/fips/fip180-1.txt (ASCII) http://csrc.nist.gov/fips/fip180-1.ps (Postscript)

12.1 Références pour information

[Localized-Key] U. Blumenthal, N. C. Hien, B. Wijnen "Key Derivation for Network Management Applications" IEEE Network Magazine, avril/mai 1997.

[DES-ANSI] Data Encryption Algorithm, American National Standards Institute. ANSI X3.92-1981, (décembre 1980).

[DESO-ANSI] Data Encryption Algorithm - Modes of Operation, American National Standards Institute. ANSI X3.106-1983, (mai 1983).

[DESG-NIST] Guidelines for Implementing and Using the NBS Data Encryption Standard, National Institute of Standards and Technology. Federal Information Processing Standard (FIPS) Publication 74, (avril, 1981).

[DEST-NIST] Validating the Correctness of Hardware Implementations of the NBS Data Encryption Standard, National Institute of Standards and Technology. Special Publication 500-20.

[DESM-NIST] Maintenance Testing for the Data Encryption Standard, National Institute of Standards and Technology. Special Publication 500-61, (août 1980).

[RFC3174] D. Eastlake 3 et P. Jones, "Algorithme US de hachage sécurisé n° 1 (SHA1)", sept. 2001. *(Info, MàJ par* 4634 *et* 6234*)*

Appendice A Installation

A.1 Paramètres d'installation du moteur SNMP

Durant l'installation, un moteur SNMP d'autorité DEVRAIT (au sens défini dans la [RFC2119]) être configuré avec plusieurs paramètres initiaux. Cela inclut :

1) Une posture de sécurité
Le choix de la posture de sécurité détermine si la configuration initiale est mise en œuvre et comment elle l'est. Un des trois choix possibles est retenu : sécurité minimum, semi sécurité, très sûr (c'est-à-dire, pas de configuration initiale).
Dans le cas d'une posture très sûre, il n'y a pas de configuration initiale, et donc les étapes suivantes ne sont pas pertinentes.

2) Un ou plusieurs secrets
Ce sont les secrets d'authentification/confidentialité pour le premier usager à configurer.
Une façon de réaliser cela est que l'installateur entre un "mot de passe" pour chaque secret nécessaire. Le mot de passe est alors converti algorithmiquement en le secret requis en :

- formant une chaîne de 1 048 576 octets de long en répétant la valeur du mot de passe aussi souvent que nécessaire, en tronquant en conséquence, et en utilisant la chaîne résultante comme entrée de l'algorithme MD5 [RFC1321]. Le résumé résultant, appelé "digest1", est utilisé dans l'étape suivante.
- une seconde chaîne est formée en enchaînant digest1, la valeur snmpEngineID du moteur SNMP, et digest1. Cette chaîne est utilisée comme entrée à l'algorithme MD5 [RFC1321].

Le résumé résultant est le secret requis (voir l'Appendice A.2).

Avec des paramètres configurés, le moteur SNMP instancie l'entrée usmUserEntry suivante dans usmUserTable :

	sans confidentialité	**avec confidentialité**
usmUserEngineID	localEngineID	localEngineID
usmUserName	"initial"	"initial"
usmUserSecurityName	"initial"	"initial"
usmUserCloneFrom	ZeroDotZero	ZeroDotZero
usmUserAuthProtocol	usmHMACMD5AuthProtocol	usmHMACMD5AuthProtocol
usmUserAuthKeyChange	""	""
usmUserOwnAuthKeyChange	""	""
usmUserPrivProtocol	aucune	usmDESPrivProtocol
usmUserPrivKeyChange	""	""
usmUserOwnPrivKeyChange	""	""
usmUserPublic	""	""
usmUserStorageType	anyValidStorageType	anyValidStorageType
usmUserStatus	active	active

Il est recommandé d'instancier aussi un ensemble de gabarits de usmUserEntries qui peuvent être utilisés comme usagers dupliqués pour les nouvelles créations de usmUserEntries. Voici les deux entrées suggérées :

	sans confidentialité	**avec confidentialité**
usmUserEngineID	localEngineID	localEngineID
usmUserName	"templateMD5"	"templateMD5"
usmUserSecurityName	"templateMD5"	"templateMD5"
usmUserCloneFrom	ZeroDotZero	ZeroDotZero
usmUserAuthProtocol	usmHMACMD5AuthProtocol	usmHMACMD5AuthProtocol
usmUserAuthKeyChange	""	""
usmUserOwnAuthKeyChange	""	""
usmUserPrivProtocol	aucune	usmDESPrivProtocol
usmUserPrivKeyChange	""	""
usmUserOwnPrivKeyChange	""	""
usmUserPublic	""	""
usmUserStorageType	permanent	permanent
usmUserStatus	active	active

	sans confidentialité	**avec confidentialité**
usmUserEngineID	localEngineID	localEngineID
usmUserName	"templateSHA"	"templateSHA"
usmUserSecurityName	"templateSHA"	"templateSHA"
usmUserCloneFrom	ZeroDotZero	ZeroDotZero
usmUserAuthProtocol	usmHMACSHAAuthProtocol	usmHMACSHAAuthProtocol
usmUserAuthKeyChange	""	""
usmUserOwnAuthKeyChange	""	""
usmUserPrivProtocol	aucune	usmDESPrivProtocol
usmUserPrivKeyChange	""	""
usmUserOwnPrivKeyChange	""	""
usmUserPublic	""	""
usmUserStorageType	permanente	permanente
usmUserStatus	active	active

A.2 Mot de passe pour l'algorithme de clé

Un fragment d'échantillon de code (au paragraphe A.2.1) montre l'algorithme de transformation de mot de passe en clé qui peut être utilisé lors de la transposition d'un mot de passe en clé d'authentification ou de confidentialité utilisant MD5. Le code source de référence de MD5 est disponible dans la [RFC1321].

Un autre fragment d'échantillon de code (au paragraphe A.2.2) montre l'algorithme de transformation de mot de passe en clé qui peut être utilisé lors de la transposition d'un mot de passe en clé d'authentification ou de confidentialité utilisant SHA (documenté dans [SHA-NIST]).

Un exemple des résultats d'une mise en œuvre correcte est fourni (au paragraphe A.3) que peut utiliser une mise en œuvre pour vérifier si elle produit le même résultat.

A.2.1 Mot de passe de code d'échantillon de clé pour MD5

```
void password_to_key_md5(
  u_char *password,    /* IN */
  u_int   passwordlen, /* IN */
  u_char *engineID,    /* IN     - pointeur sur snmpEngineID  */
  u_int   engineLength,/* IN     - longueur de snmpEngineID */
  u_char *key)         /* OUT    - pointeur sur l'antémémoire de 16 octets de l'appelant */
{
  MD5_CTX     MD;
  u_char     *cp, password_buf[64];
  u_long      password_index = 0;
  u_long      count = 0, i;

  MD5Init (&MD);                /* initialise MD5 */
```

```
/**********************************************/
/* Utiliser en boucle jusqu'à avoir fait 1 Méga octets */
/**********************************************/
    while (count < 1048576) {
      cp = password_buf;
      for (i = 0; i < 64; i) {

/***************************************************/
/* Prendre le prochain octet du mot de passe, en revenant au début du mot de passe si nécessaire.*/
/***************************************************/
        *cp = password[password_index % passwordlen];
    }
    MD5Update (&MD, password_buf, 64);
    count = 64;
  }
  MD5Final (key, &MD);         /* dit à MD5 que c'est fini */

/*******************************************************/
/* Maintenant localiser la clé avec l'identifiant de moteur et passer par MD5 pour produire la clé finale.*/
/* On peut vouloir s'assurer que engineLength ≤ 32,  autrement on a besoin d'une antémémoire de plus de 64 */
/*******************************************************/
  memcpy(password_buf, key, 16);
  memcpy(password_buf+16, engineID, engineLength);
  memcpy(password_buf+16+password, key, 16);

  MD5Init(&MD);
  MD5Update(&MD, mot de passe_buf, 32 □□□□);
  MD5Final(key, &MD);
  return;
}
```

A.2.2 Mot de passe de code d'échantillon de clé pour SHA

```
void password_to_key_sha(
  u_char *password,    /* IN */
  u_int   passwordlen, /* IN */
  u_char *engineID,    /* IN  - pointeur sur snmpEngineID  */
  u_int   engineLength,/* IN  - longueur de snmpEngineID */
  u_char *key)         /* OUT - pointeur sur la mémoire tampon de 20 octets de l'appelant */
{
  SHA_CTX     SH;
  u_char     *cp, password_buf[72];
  u_long      password_index = 0;
  u_long      count = 0, i;

  SHAInit (&SH);   /* initialise SHA */

/**********************************************/
/* Utiliser en boucle jusqu'à avoir fait 1 Méga octets */
/**********************************************/
  while (count < 1048576) {
    cp = password_buf;
    for (i = 0; i < 64; i) {
/***************************************************/
/* Prendre le prochain octet du mot de passe, en revenant au début du mot de passe si nécessaire.*/
/***************************************************/
      *cp = password[password_index % passwordlen];
    }
    SHAUpdate (&SH, password_buf, 64);
    count = 64;
  }
  SHAFinal (key, &SH);             /* dit à SHA que c'est fini */
```

```
/*************************************************/
/* On localise maintenant la clé avec le engineID et on passe par SHA pour produire la clé finale */
/* On peut vouloir s'assurer que engineLength ≤ 32, autrement on doit utiliser une antémémoire de plus de 72  */
/*************************************************/
memcpy(password_buf, key, 20);
  memcpy(password_buf+20, engineID, engineLength);
  memcpy(password_buf+20+engineLength, key, 20);

  SHAInit(&SH);
  SHAUpdate(&SH, password_buf, 40+engineLength);
  SHAFinal(key, &SH);
  return;
}
```

A.3 Mot de passe pour les résultats d'échantillon de clé

A.3.1 Mot de passe de résultat d'échantillon de clé en utilisant MD5

Voici un exemple de résultat de l'algorithme de mot de passe à clé pour une clé de 16 octets avec MD5.

Avec un mot de passe de "maplesyrup" le résultat de l'algorithme de mot de passe à clé avant que la clé soit localisée avec le snmpEngineID du moteur SNMP est : '9f af 32 83 88 4e 92 83 4e bc 98 47 d8 ed d9 63'H

Après la localisation de la clé intermédiaire (montrés ci dessus) avec la valeur de snmpEngineID de :
'00 00 00 00 00 00 00 00 00 00 00 02'H
le résultat final de l'algorithme de mot de passe à clé est : '52 6f 5e ed 9f cc e2 6f 89 64 c2 93 07 87 d8 2b'H

A.3.2 Mot de passe de résultat d'échantillon de clé en utilisant SHA

Voici un exemple de résultat de l'algorithme de mot de passe à clé pour une clé de 20 octets en utilisant SHA.

Avec un mot de passe de "maplesyrup", le résultat de l'algorithme de mot de passe à clé avant que la clé soit localisée avec le snmpEngineID du moteur SNMP est : '9f b5 cc 03 81 49 7b 37 93 52 89 39 ff 78 8d 5d 79 14 52 11'H

Après la localisation de la clé intermédiaire (montrée ci-dessus) avec la valeur de snmpEngineID de :
'00 00 00 00 00 00 00 00 00 00 00 02'H
le résultat final de l'algorithme de mot de passe en clé est :
'66 95 fe bc 92 88 e3 62 82 23 5f c7 15 1f 12 84 97 b3 8f 3f'H

A.4 Exemple de codage de msgSecurityParameters

Les msgSecurityParameters dans un message SNMP sont représentés comme une CHAINE D'OCTETS. Cette CHAINE D'OCTETS devrait être considérée comme opaque en dehors d'un modèle de sécurité spécifique.

Le modèle de sécurité fondé sur l'utilisateur définit le contenu de la CHAINE D'OCTETS comme une SEQUENCE (voir au paragraphe 2.4).

Étant données ces deux propriétés, ce qui suit est un exemple des msgSecurityParameters pour le modèle de sécurité fondé sur l'utilisateur, codé comme une CHAINE D'OCTETS:

```
04 <longueur>
30 <longueur>
04 <longueur>   <msgAuthoritativeEngineID>
02 <longueur>   <msgAuthoritativeEngineBoots>
02 <longueur>   <msgAuthoritativeEngineTime>
04 <longueur>   <msgUserName>
04 0c           <HMAC-MD5-96-digest>
04 08           <sel>
```

Voici l'exemple, mais avec les valeurs réelles (sauf pour le résumé dans msgAuthenticationParameters et le sel dans

msgPrivacyParameters, qui dépendent de données variables qu'on n'a pas définies ici) :

Données hexadécimales	Description	
04 39	CHAINE D'OCTETS,	longueur 57
30 37	SEQUENCE,	longueur 55
04 0c 80000002	msgAuthoritativeEngineID :	IBM
01		adresse IPv4
09840301		9.132.3.1
02 01 01	msgAuthoritativeEngineBoots :	1
02 02 0101	msgAuthoritativeEngineTime :	257
04 04 62657274	msgUserName:	bert
04 0c 01234567	msgAuthenticationParameters :	valeur d'échantillon
89abcdef		
fedcba98		
04 08 01234567	msgPrivacyParameters:	valeur d'échantillon
89abcdef		

A.5 Exemple de résultats de keyChange

A.5.1 Exemple de résultats de keyChange avec MD5

Supposons qu'un utilisateur ait un mot de passe actuel de "maplesyrup" comme au paragraphe A.3.1. et supposons aussi le snmpEngineID de 12 octets : '00 00 00 00 00 00 00 00 00 00 00 02'H
Si on veut maintenant changer le mot de passe en "newsyrup", on calcule d'abord la clé pour le nouveau mot de passe. Elle est comme suit : '01 ad d2 73 10 7c 4e 59 6b 4b 00 f8 2b 1d 42 a7'H
Si on la localise pour le snmpEngineID ci-dessus, la nouvelle clé localisée devient :
'87 02 1d 7b d9 d1 01 ba 05 ea 6e 3b f9 d9 bd 4a'H

Si on utilise alors une valeur aléatoire (pas très bonne, mais facile à tester) de :
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'H
la valeur qu'on doit alors envoyer pour keyChange est :
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 88 05 61 51 41 67 6c c9 19 61 74 e7 42 a3 25 51'H

Si c'était pour la clé de confidentialité, ce serait exactement la même.

A.5.2 Exemple de résultats de keyChange avec SHA

Supposons qu'un utilisateur ait un mot de passe actuel de "maplesyrup" comme au paragraphe A.3.2. et supposons aussi le snmpEngineID de 12 octets : '00 00 00 00 00 00 00 00 00 00 00 02'H
Si on veut maintenant changer le mot de passe en "newsyrup", on calcule d'abord la clé pour le nouveau mot de passe. Elle est comme suit : '3a 51 a6 d7 36 aa 34 7b 83 dc 4a 87 e3 e5 5e e4 d6 98 ac 71'H
Si on la localise pour le snmpEngineID ci-dessus, la nouvelle clé localisée devient :
'78 e2 dc ce 79 d5 94 03 b5 8c 1b ba a5 bf f4 63 91 f1 cd 25'H
Si on utilise alors une valeur aléatoire (pas très bonne, mais facile à tester) de :
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'H
la valeur qu'on doit alors envoyer pour keyChange est :
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 9c 10 17 f4 fd 48 3d 2d e8 d5 fa db f8 43 92 cb 06 45 70 51'

Pour la clé utilisée pour la confidentialité, la nouvelle clé non localisée serait :
'3a 51 a6 d7 36 aa 34 7b 83 dc 4a 87 e3 e5 5e e4 d6 98 ac 71'H

Pour la clé utilisée pour la confidentialité, la nouvelle clé localisée serait (noter que la clé localisée va être tronquée à 16 octets pour DES) : '78 e2 dc ce 79 d5 94 03 b5 8c 1b ba a5 bf f4 63'H

Si on utilise alors une valeur aléatoire (pas très bonne, mais facile à tester) de :
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'H
la valeur qu'on doit envoyer pour keyChange pour la clé de confidentialité est :
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 '7e f8 d8 a4 c9 cd b2 6b 47 59 1c d8 52 ff 88 b5'H

Appendice B Journal des changements

Changements depuis la RFC2574 :

- mise à jour des références
- mise à jour des informations de contact
- précisions
  - à la première contrainte élément1) page 6.
  - à la clause DESCRIPTION de usmUserCloneFrom
  - à securityName au paragraphe 2.1
- correction de "répondeur de commandes" en "générateur de commandes" au dernier paragraphe de la clause DESCRIPTION de usmUserTable.

Changements depuis la RFC2274 :

- correction de msgUserName pour permettre une taille de zéro et explication que ce peut être utilisé pour la découverte de snmpEngineID
- précision au paragraphe 3.1 étapes 4.b, 5, 6 et 8.b
- précision au paragraphe 3.2 alinéa 2
- précision au paragraphe 3.2 étape 7.dernier alinéa, étape 7.b.1 second alinéa et étape 7.b.2 troisième alinéa.
- précision au paragraphe 4 pour indique que la découverte peut utiliser un userName de longueur zéro dans les messages non authentifiés, tandis qu'un userName valide doit être utilisé dans les messages authentifiés
- ajout d'une clause REVISION à IDENTITÉ DE MODULE
- précision à la TC KeyChange par l'ajout d'une note disant que les clés localisées doivent être utilisées lors du calcul d'une valeur de KeyChange
- ajout d'une précision à la clause DESCRIPTION de usmUserTable
- ajout d'un texte décrivant une procédure recommandée pour ajouter un nouvel usager
- précision sur l'utilisation de l'objet usmUserCloneFrom
- précision sur comment et dans quelles conditions usmUserAuthProtocol et usmUserPrivProtocol peuvent être initialisés et/ou changés
- ajout d'un commentaire sur les tailles typiques pour usmUserAuthKeyChange et usmUserPrivKeyChange et aussi pour usmUserOwnAuthKeyChange et usmUserOwnPrivKeyChange
- ajout d'une clarifications à la clause DESCRIPTION de usmUserAuthKeyChange, usmUserOwnAuthKeychange, usmUserPrivKeyChange et usmUserOwnPrivKeychange
- ajout d'une clarification à la clause DESCRIPTION de usmUserStorageType
- ajout d'une clarification à la clause DESCRIPTION de usmUserStatus
- précision sur la procédure de génération de l'IV au paragraphe 8.1.1.1 et clarification au paragraphe 8.3.1 étape 1 et au paragraphe 8.3.2. étape 3
- précision au paragraphe 11.2 et ajout d'un avertissement sur les mots de passe de tailles différentes avec des chaîne répétitives qui peuvent résulter en la même clé
- ajout d'un gabarit d'usager à l'Appendice A pour les besoins du processus de clonage.
- correction des exemple de code C à l'Appendice A
- correction des exemples de clés générées à l'Appendice A
- ajout d'un exemple de valeurs de KeyChange à l'Appendice A
- utilisation des classes de PDU à la place des types de PDU de la RFC1905
- ajout d'un texte sur la sécurité au paragraphe sur les rapports et le contrôle d'accès à la MIB
- suppression d'une note incorrecte à la fin du paragraphe 3.2 étape 7.
- ajout d'une note au paragraphe 3.2 étape 3
- correction de diverses fautes de frappe.
- correction de la procédure au paragraphe 3.2 étape 2.a)
- diverses clarifications.
- correction des références aux documents nouveaux/révisés
- changement aux données en antémémoire qui ne sont pas utilisées

Adresse des éditeurs

Uri Blumenthal
Lucent Technologies
67 Whippany Rd.
Whippany, NJ 07981

Bert Wijnen
Lucent Technologies
Schagen 33
3461 GL Linschoten

USA
téléphone : 973-386-2163
mél : uri@lucent.com

Netherlands
téléphone : 348-480-685
mél : bwijnen@lucent.com

Déclaration complète de droits de reproduction

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