Groupe de travail Réseau
Request for Comments : 5116
Catégorie : Sur la voie de la normalisation

D. McGrew, Cisco Systems, Inc.
janvier 2008
Traduction Claude Brière de L'Isle

Interface et algorithmes pour chiffrement authentifié

Résumé
Le présent document définit des algorithmes pour le chiffrement authentifié avec données associées (AEAD, *Authenticated Encryption with Associated Data*) et définit une interface uniforme et un registre pour ces algorithmes. L'interface et le registre peuvent être utilisés comme un ensemble de suites crypto algorithmiques indépendant de l'application. Cette approche donne des avantages en efficacité et en sécurité, et promeut la réutilisation des mises en œuvre de chiffrement.

Table des Matières

1. Introduction

Le chiffrement authentifié [BN00] est une forme de chiffrement qui, en plus de fournir la confidentialité pour le texte source qui est chiffré, donne un moyen de vérifier son intégrité et son authenticité. Le chiffrement authentifié avec données associées (AEAD, *Authenticated Encryption with Associated Data*) [R02], ajoute la capacité de vérifier l'intégrité et l'authenticité de certaines des données associées (AD, *Associated Data*) aussi appelées "données authentifiées supplémentaires", qui ne sont pas chiffrées.

1.1 Fondements

De nombreuses applications cryptographiques exigent à la fois la confidentialité et l'authentification du message. La confidentialité est un service de sécurité qui assure que les données ne sont disponibles qu'à ceux qui sont autorisés à les obtenir ; elle est généralement réalisée par le chiffrement. L'authentification de message est le service qui assure que les données n'ont pas été altérées ou falsifiées par des entités non autorisées ; cela peut être réalisé en utilisant un code d'authentification de message (MAC, *Message Authentication Code*). Ce service est aussi appelé intégrité des données. De nombreuses applications utilisent ensemble une méthode de chiffrement et un MAC pour fournir ces deux services de sécurité, chaque algorithme utilisant une clé indépendante. Plus récemment, l'idée de fournir ces deux services de sécurité en utilisant un seul algorithme cryptographique a été acceptée. Dans ce concept, le chiffrement et le MAC sont remplacés par un algorithme de chiffrement authentifié avec données associées (AEAD, *Authenticated Encryption with Associated Data*).

Plusieurs algorithmes de chiffrement qui mettent en œuvre des algorithmes AEAD ont été définis, incluant des modes de fonctionnement de chiffrement de bloc et des algorithmes dédiés. Certains de ces algorithmes ont été adoptés et se sont révélés utiles en pratique. De plus, AEAD est proche d'une vision "idéale" du chiffrement, comme celui utilisé dans l'analyse automatisée des protocoles cryptographiques (voir, par exemple, le paragraphe 2.5 de [BOYD]).

Les avantages des algorithmes AEAD, et cette interface, sont décrits au paragraphe 1.3.

1.2 Portée

Dans ce document, on définit un algorithme AEAD comme une abstraction, en spécifiant une interface à un AEAD et en définissant un registre IANA pour les algorithmes AEAD. On remplit ce registre avec quatre algorithmes AEAD fondés sur la norme de chiffrement évoluée (AES, *Advanced Encryption Standard*) en mode Galois/compteur [GCM] avec des clés de 128 et 256 bits, et AES en mode compteur et MAC CBC [CCM] avec des clés de 128 et 256 bits.

Ensuite, on définit l'interface AEAD (Section 2) et on donne des directives sur l'utilisation des algorithmes AEAD (Section 3) en mentionnant les exigences que chaque algorithme AEAD doit satisfaire (Section 4). On définit ensuite plusieurs algorithmes AEAD (Section 5) et on établit un registre IANA pour les algorithmes AEAD (Section 6). Enfin, on discute quelques autres considérations (Section 7).

La spécification d'interface AEAD ne traite pas des questions de protocole de sécurité comme les services anti-répétition ou les décisions de contrôle d'accès qui sont prises sur des données authentifiées. La spécification vise plutôt à abstraire la cryptographie de ces problèmes. L'interface, et les directives sur la façon de l'utiliser, sont cohérentes avec les recommandations de [EEM04].

1.3 Avantages

L'approche AEAD permet aux applications qui ont besoin de services de sécurité cryptographiques d'adopter plus facilement ces services. Elle bénéficie au concepteur d'application en lui permettant de se concentrer sur les questions importantes comme les services de sécurité, la canonisation et la surveillance des données, et de se libérer du besoin de concevoir les mécanismes de chiffrement qui satisfont à leurs objectifs de sécurité. Il est important que la sécurité d'un algorithme AEAD puisse être analysée indépendamment de son utilisation dans une application particulière. Cette propriété libère l'utilisateur de l'AEAD du besoin de considérer des aspects de sécurité tels que l'ordre relatif de l'authentification et du chiffrement et la sécurité de la combinaison particulière de chiffrement et de MAC, comme la perte potentielle de la confidentialité à travers le MAC. Le concepteur d'application qui utilise l'interface AEAD n'a pas besoin de choisir un algorithme AEAD particulier durant le stade de conception. De plus, l'interface à l'AEAD est relativement simple, car elle exige seulement une clé en entrée et requiert seulement un identifiant pour indiquer l'algorithme utilisé dans un cas particulier.

L'approche de AEAD bénéficie à la mise en œuvre des algorithmes de chiffrement en rendant disponibles des optimisations qui ne seraient autrement pas possibles pour réduire la quantité de calcul, le coût de mise en œuvre, et/ou les exigences de mémorisation. Plus l'interface est simple, plus les essais sont faciles ; c'est un avantage considérable pour une mise en œuvre d'algorithme de chiffrement. En fournissant une interface uniforme pour accéder aux services cryptographiques, l'approche de l'AEAD permet à une seule mise en œuvre de chiffrement de prendre plus facilement en charge plusieurs applications. Par exemple, un module de matériel qui prend en charge l'interface AEAD peut facilement fournir l'accélération du chiffrement à toute application qui utilise cette interface, même si l'application n'a pas été conçue lors de la construction du module.

1.4 Conventions utilisées dans ce document

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

2. Interface AEAD

Un algorithme AEAD a deux opérations, le chiffrement authentifié et le déchiffrement authentifié. Les entrées et les résultats de ces algorithmes sont définis ci-dessous en termes de chaînes d'octets.

Une mise en œuvre PEUT accepter des entrées supplémentaires. Par exemple, une entrée pourrait être fournie pour permettre à l'utilisateur de choisir entre différentes stratégies de mise en œuvre. Cependant, de telles extensions NE DOIVENT PAS affecter l'interopérabilité avec les autres mises en œuvre.

2.1 Chiffrement authentifié

L'opération de chiffrement authentifié a quatre entrées, dont chacune est une chaîne d'octets :

- Une clé secrète K, qui DOIT être générée de façon uniformément aléatoire ou pseudo aléatoire.

- Un nom occcasionnel *(nom occasionnel)* N. Chaque nom occasionnel fourni à des invocations distinctes de l'opération de chiffrement authentifié DOIT être distinct, pour toute valeur particulière de la clé, sauf si tous les noms occasionnels sont de longueur zéro. Les applications qui peuvent générer des noms occasionnels distincts DEVRAIENT utiliser la méthode de formation de noms occasionnels définie au paragraphe 3.2, et PEUVENT utiliser toute autre méthode qui satisfait à l'exigence d'unicité. Les autres applications DEVRAIENT utiliser des noms occasionnels de longueur zéro.

- Un texte source P, qui contient les données à chiffrer et authentifier.

- Les données associées A, qui contiennent les données à authentifier, mais pas à chiffrer.

Il y a une seule sortie, un texte chiffré C, qui est au moins aussi long que le texte source, ou l'indication que l'opération de chiffrement demandée n'a pas pu être effectuée.

Toutes les entrées et sorties sont des chaînes d'octets de longueur variable, dont la longueur est soumise aux restrictions suivantes :

Le nombre d'octets dans la clé K est entre 1 et 255. Pour chaque algorithme AEAD, la longueur de K DOIT être fixe.

Pour toute valeur particulière de la clé, soit 1) chaque nom occasionnel fourni à des invocations distinctes d'opération de chiffrement authentifié DOIT être distinct, soit 2) chaque nom occasionnel DOIT être de longueur zéro. Si des noms occasionnels de longueur zéro sont utilisés avec une clé particulière, alors chaque nom occasionnel utilisé avec cette clé DOIT avoir une longueur de zéro. autrement, le nombre d'octets dans le nom occasionnel DEVRAIT être douze (12). Les noms occasionnels avec des longueurs différentes PEUVENT être utilisés avec une clé particulière. Certains algorithmes ne peuvent pas être utilisés avec des noms occasionnels de longueur zéro, mais d'autres le peuvent ; voir à la Section 4. Les applications qui se conforment à la longueur de nom occasionnel recommandée éviteront d'avoir à construire des noms occasionnels de longueurs différentes, selon l'algorithme qu'elles utilisent. Cette directive aide à garder la logique spécifique de l'algorithme en dehors des applications.

Le nombre d'octets dans le texte source P PEUT être zéro.

Le nombre d'octets dans les données associées A PEUT être zéro.

Le nombre d'octets dans le texte chiffré C PEUT être zéro.

Le présente spécification ne fixe pas de longueur maximum au nom occasionnel, au texte source, au texte chiffré, ou aux données authentifiés supplémentaires. Cependant, un algorithme AEAD particulier PEUT restreindre encore plus la longueur de ces entrées et sorties. Une mise en œuvre particulière de AEAD PEUT encore restreindre les longueurs de ses entrées et sorties. Si il est demandé à une mise en œuvre particulière d'un algorithme AEAD de traiter une entrée qui sort de la gamme des longueurs admissibles, ou une entrée qui sort de la gamme des longueurs acceptées par cette mise en œuvre,

elle DOIT retourner un code d'erreur et NE DOIT PAS sortir d'autre information. En particulier, des données partiellement chiffrées ou partiellement déchiffrées NE DOIVENT PAS être retournées

La confidentialité et l'authentification de message sont toutes deux fournies sur le texte source P. Quand la longueur de P est zéro, l'algorithme AEAD agit comme un code d'authentification de message sur l'entrée A.

Les données associées A sont utilisées pour protéger les informations qui ont besoin d'être authentifiées, mais n'ont pas besoin de rester confidentielles. Quand on utilise un AEAD pour sécuriser un protocole réseau, par exemple, cette entrée pourrait inclure les adresses, accès, numéros de séquence, numéros de version de protocole, et autres champs qui indiquent comment le texte source ou le texte chiffré devrait être traité, ou transmis. Dans de nomabreuses situations, il est désirable d'authentifier ces champs, bien qu'ils doivent rester en clair pour permettre au réseau ou système de fonctionner correctement. Quand ces données sont incluses dans l'entrée A, l'authentification est fournie sans copier les données dans le texte source.

La clé secrète K NE DOIT PAS être incluse dans une autre entrée (N, P, et A). (Cette restriction ne signifie pas que les valeurs de ces entrées doivent être vérifiées pour s'assurer qu'elles n'incluent pas de sous chaînes correspondant à la clé ; cela signifie plutôt que la clé ne doit pas être explicitement copiée dans ces entrées.)

Le nom occasionnel est authentifié en interne à l'algorithme, et il n'est pas nécessaire de l'inclure dans l'entrée d'AD. Le nom occasionnel PEUT être inclus dans P ou A si cela convient à l'application.

Le nom occasionnel PEUT être mémorisé ou transporté avec le texte chiffré, ou il PEUT être reconstruit immédiatement avant l'opération de déchiffrement authentifié. Il est suffisant de fournir au module de déchiffrement assez d'informations pour lui permettre de construire le nom occasionnel. (Par exemple, un système pourrait utiliser un nom occasionnel consistant en un numéro de séquence dans un format particulier, auquel cas il pourrait être déduit de l'ordre des textes chiffrés.) Parce que le processus de déchiffrement authentifié détecte les valeurs incorrectes de nom occasionnel, aucun échec de la sécurité ne va résulter de la reconstruction incorrecte d'un nom occasionnel et de son injection dans une opération de déchiffrement authentifié. Toute méthode de reconstruction de nom occasionnel va devoir tenir compte de la possibilité de perte ou de réarrangement des textes chiffrés entre les processus de chiffrement et de déchiffrement.

Les applications NE DOIVENT PAS supposer de structure ou format particulier du texte chiffré.

2.2 Déchiffrement authentifié

L'opération de déchiffrement authentifié a quatre entrées : K, N, A, et C, comme défini ci-dessus. Elle n'a qu'un seul résultat, soit une valeur de texte source P, soit un symbole spécial FAIL qui indique que les entrées ne sont pas authentiques. Un texte chiffré C, un nom occasionnel N, et des données associées A sont authentiques pour la clé K quand C est généré par l'opération de chiffrement avec les entrées K, N, P, et A, pour certaines valeurs de N, P, et A. L'opération de déchiffrement authentifié va, avec une forte probabilité, retourner FAIL chaque fois que les entrées N, P, et A ont été élaborées par un adversaire qui respecte le nom occasionnel mais ne connaît pas la clé secrète (en supposant que l'algorithme AEAD est sûr).

2.3 Formattage des données

Le présent document ne spécifie pas de codage particulier pour les entrées et sorties de AEAD, car le codage n'affecte pas les services de sécurité fournis par un algorithme AEAD.

Quand elle choisit le format des données d'application, une application DEVRAIT positionner le texte chiffré C de façon telle qu'il apparaisse après toutes les autres données qui sont nécessaires pour construire les autres entrées de l'opération de déchiffrement authentifié. Par exemple, si le nom occasionnel et le texte chiffré apparaissent tous deux dans un paquet, la valeur du nom occasionnel devrait précéder le texte chiffré. Cette règle facilite la mise en œuvre efficace et simple des algorithmes AEAD dans le matériel.

3. Lignes directrices sur l'utilisation des algorithmes AEAD

Cette section donne des directives qui doivent être suivies afin d'utiliser en toute sécurité un algorithme AEAD.

Si une application n'est pas capable de respecter l'exigence d'unicité de la génération du nom occasionnel, elle DOIT alors utiliser un nom occasionnel de longueur zéro. Les algorithmes aléatoires ou à états pleins, qui sont définis plus loin,

conviennent pour l'usage de telles applications. Autrement, une application DEVRAIT utiliser des noms occasionnels d'une longueur de douze octets. Comme les algorithmes sont encouragés à prendre en charge cette longueur, les applications devraient l'utiliser pour faciliter l'interopérabilité.

3.1 Exigences pour la génération de nom occasionnel

Il est essentiel pour la sécurité que les noms occasionnels soient construits d'une manière qui respecte l'exigence que chaque valeur de nom occasionnel soit distincte pour chaque invocation de l'opération de chiffrement authentifié, pour toute valeur fixée de la clé. Dans ce paragraphe, on attire l'attention sur certaines conséquences de cette exigence dans différents scénarios.

Quand plusieurs appareils effectuent le chiffrement en utilisant une seule clé, ces appareils doivent se coordonner pour s'assurer que les noms occasionnels sont uniques. Une façon simple de le faire est d'utiliser un format de nom occasionnel qui contient un champ distinct pour chacun des appareils, comme décrit au paragraphe 3.2. Noter qu'il n'est pas nécessaire de coordonner les détails du format de nom occasionnel entre le chiffreur et le déchiffreur, tant que le nom occasionnel entier est envoyé ou mémorisé avec le texte chiffré et est donc disponible au déchiffreur. Si le nom occasionnel complet n'est pas disponible pour le déchiffreur, celui-ci va alors avoir besoin de savoir comment le nom occasionnel est structuré afin de pouvoir le reconstruire. Les applications DEVRAIENT fournir des moteurs de chiffrement avec une certaine liberté de choix de leurs noms occasionnels ; par exemple, un nom occasionnel pourrait contenir à la fois un compteur et un champ réglé par le chiffreur mais pas traité par le receveur. Cette liberté permet à un ensemble d'appareils de chiffrement de se coordonner plus directement pour s'assurer de l'unicité de leurs noms occasionnels.

Si une clé secrète va être utilisée pendant longtemps, par exemple, à travers plusieurs réamorçages, alors le nom occasionnel va devoir être mémorisé dans une mémoire non volatile. Dans ce cas, il est essentiel d'utiliser un point de contrôle du nom occasionnel ; c'est-à-dire que la valeur courante du nom occasionnel devrait être mémorisée pour fournir les informations d'état nécessaires pour reprendre le chiffrement en cas de défaillance inattendue. Une façon simple d'avoir une assurance forte qu'une valeur de nom occasionnel ne va pas être utilisée de façon répétée est d'attendre jusqu'à ce que le processus de chiffrement reçoive confirmation de la part du processus de mémorisation du succès de la mémorisation de la valeur de nom occasionnel. Parce que cette méthode peut ajouter une latence significative, il peut être désirable de mémoriser une valeur de nom occasionnel qui soit plusieurs fois au delà de celle qui suit. Par exemple, le nom occasionnel 100 pourrait être mémorisé, après quoi les noms occasionnels de 1 à 99 pourraient être utilisés pour le chiffrement. La valeur de nom occasionnel 200 pourrait être mémorisée en même temps que les noms occasionnels de 1 à 99 sont utilisés, et ainsi de suite.

De nombreux problèmes de réutilisation de nom occasionnel peuvent être évités en changeant une clé dans une situation où la coordination de nom occasionnel est difficile.

Chaque algorithme AEAD DEVRAIT décrire quelle dégradation de la sécurité résulterait d'une réutilisation invlontaire d'une valeur de nom occasionnel.

3.2 Formation recommandée de nom occasionnel

La méthode suivante pour construire des noms occasionnels est RECOMMANDÉE. Le nom occasionnel est formaté comme illustré à la Figure 1, avec les octets initiaux consistant en un champ fixé, et les octets finaux consistant en un champ Compteur. Pour chaque clé fixée, la longueur de chacun de ces champs, et donc la longueur du nom occasionnel, est fixe. Les mises en œuvre DEVRAIENT accepter des noms occasionnels de 12 octets dans lesquels le champ Compteur est de quatre octets.

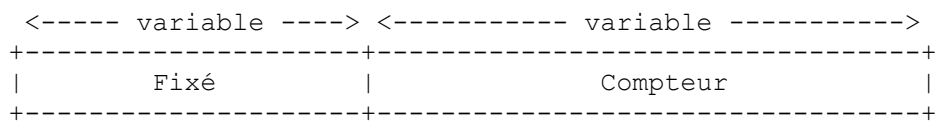

Figure 1 : Format recommandé de nom occasionnel

Les champs Compteur des noms occasionnels successifs forment un séquence d'accroissement monotone, quand ces champs sont considérés comme des entiers non signés dans l'ordre des octets du réseau. La longueur du champ Compteur DOIT rester constante pour tous les noms occasionnels générés pour un certain appareil de chiffrement. La partie Compteur DEVRAIT être égale à zéro pour le premier nom occasionnel, et être incrémentée de un pour chaque nom occasionnel suivant généré. Cependant, toute valeur particulière de compteur PEUT être sautée, et laissée en dehors de la séquence des valeurs utilisées, si cela convient. Par exemple, une application pourrait choisir de sauter la valeur de compteur initial =0, et

régler le champ Compteur au nom occasionnel initial de 1. Donc, au plus 2^(8*C) noms occasionnels peuvent être générés quand le champ Compteur est long de C octets.

Le champ Fixé DOIT rester constant pour tous les noms occasionnels générés pour un appareil de chiffrement donné. Si des appareils différents effectuent le chiffrement avec une seule clé, alors chaque appareil distinct DOIT utiliser un champ Fixé distinct, pour assurer l'unicité des noms occasionnels. Donc, au plus 2^(8*F) chiffreurs distincts peuvent partager une clé quand le champ Fixé est long de F octets.

3.2.1 Noms occasionnels partiellement implicites

Dans certains cas, il est désirable de ne pas transmettre ou mémoriser un nom occasionnel entier, mais plutôt de reconstruire cette valeur à partir d'informations du contexte immédiatement avant le déchiffrement. Par exemple, les textes chiffrés pourraient être mémorisés à la suite sur un disque, et le nom occasionnel pour un texte chiffré particulier pourrait être déduit de sa localisation, pour autant que la règle pour la génération des noms occasionnels soit connue du déchiffreur. On appelle la portion de nom occasionnel qui est mémorisée ou envoyée avec le texte chiffré la partie explicite. On appelle la portion du nom occasionnel qui est déduite la partie implicite. Quand une partie du nom occasionnel est implicite, la spécialisation suivante du format ci-dessus est RECOMMANDÉE. Le champ Fixé est divisé en deux sous champs : un champ Fixé commun et un champs Fixé distinct. Ce format est montré à la Figure 2. Si des appareils différents effectuent le chiffrement avec une seule clé, alors chaque appareil distinct DOIT utiliser un champ Fixé distinct différent. Le champ Fixé commun est commun à tous les noms occasionnels. Le champ Fixé distinct et le champ Compteur DOIVENT être dans la partie explicite du nom occasionnel. Le champ Fixé commun PEUT être dans la partie implicite du nom occasionnel. Ces conventions assurent que le nom occasionnel est facile à reconstruire à partir des données explicites.

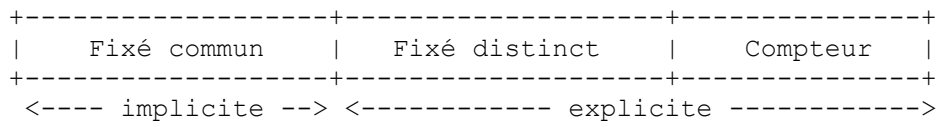

Figure 2 : Format partiellement implicite de nom occasionnel

La raison du format partiellement implicite du nom occasionnel est la suivante. Cette méthode de construction de nom occasionnel incorpore les meilleurs pratiques connues ; elle est utilisée par l'encapsulation de charge utile de sécurité (ESP, *Encapuslating Security Payload*) GCM [RFC4106] et CCM [RFC4309], dans lesquelles le champ Fixé contient la valeur du sel et les huit octets de moindre poids du nom occasionnel sont explicitement portés dans la paquet ESP. Dans ESP GCM, le champ Fixé doit aussi être d'au moins quatre octets, afin qu'il puisse contenir la valeur du sel. Dans ESP CCM, le champ Fixé doit être d'au moins trois octets pour la même raison. Cette méthode de génération de nom occasionnel est aussi utilisée par plusieurs variantes de mode compteur incluant ESP CTR.

3.3 Construction des entrées d'AEAD

Si l'entrée d'AD est construite à partir de plusieurs éléments de données, il est alors essentiel qu'elle soit analysable sans ambiguïté dans ses éléments constitutifs, sans utiliser de données non authentifiés dans le processus d'analyse. (En termes mathématiques, l'entrée d'AD doit être une fonction injective des éléments de données.) Si une application construit son entrée d'AD de telle façon qu'il y ait deux ensembles distincts d'éléments de données qui résultent en la même valeur d'AD, alors un attaquant pourrait causer l'acceptation par un receveur d'un ensemble bogué en substituant un ensemble à un autre. L'exigence que les AD soient analysables de façon univoque assure que cette attaque n'est pas possible. Cette exigence est satisfaite de façon triviale si les AD sont composées d'éléments de longueur fixe. Si les AD contiennent une chaîne de longueur variable, par exemple, cette exigence peut aussi être satisfaite en incluant la longueur de la chaîne dans les AD.

De même, si le texte source est construit à partir de plusieurs éléments de données, il est alors essentiel qu'il soit analysable sans ambiguïté dans ses éléments constitutifs, sans utiliser de données non authentifiés dans le processus d'analyse. Noter que les données incluses dans les AD peuvent être utilisées dans l'analyse du texte source, bien que comme les AD ne sont pas chiffrées, il y ait une perte potentielle de confidentialité quand les informations sur le texte source sont incluses dans les AD.

3.4 Exemple d'utilisation

Pour utiliser un algorithme AEAD, une application doit définir comment les entrées de l'algorithme de chiffrement sont définies en termes de données d'application, et comment le texte chiffré et le nom occasionnel sont portés. La façon la plus claire de faire cela est d'exprimer chaque entrée dans les termes des données qui la forment, puis d'exprimer les données d'application en termes de résultats de l'opération de chiffrement AEAD.

Par exemple, AES-GCM ESP [RFC4106] peut être exprimé comme suit. Les entrées d'AEAD sont :
P = ResteDesDonnéesDeChargeUtile || BourrageTFC || Bourrage || LongueurDeBourrage || ProchainEn-tête
N = Sel || IV
A = SPI || Numéro de séquence

où le symbole "||" note l'opération d'enchaînement, et les champs ResteDesDonnéesDeChargeUtile, BourrageTFC, Bourrage, LongueurDeBourrage, ProchainEn-tête, SPI, et Numéro de séquence sont comme défini dans la [RFC4303], et les champs Sel et IV sont comme défini dans la [RFC4106]. Le champ ResteDesDonnéesDeChargeUtile contient les données du texte source qui sont décrites par le champ ProchainEn-tête, et aucune autre donnée. (On rappelle que le champ DonnéesDeChargeUtile contient l'IV et le ResteDesDonnéesDeChargeUtile ; voir la Figure 2 de la [RFC4303] pour une illustration.)

Le format du paquet ESP peut être exprimé par :

ESP = SPI || Numéro de séquence || IV || C

où C est le texte chiffré AEAD (qui dans ce cas incorpore l'étiquette d'authentification). Noter qu'ici on n'a pas décrit l'utilisation du numéro de séquence étendu ESP.

4. Exigences pour les spécifications d'algorithme AEAD

Chaque algorithme AEAD DOIT seulement accepter des clés d'une longueur K_LEN fixée, et NE DOIT PAS exiger de format de données particulier pour les clés fournies en entrée. Un algorithme qui exige une telle structure (par exemple, une avec des sous clés dans un format de vérification de parité particulier) va devoir le fournir en interne.

Chaque algorithme AEAD DOIT accepter tout texte source d'une longueur entre zéro et P_MAX octets, inclus, où la valeur P_MAX est spécifique de cet algorithme. La valeur de P_MAX DOIT être supérieure à zéro, et DEVRAIT être au moins de 65 536 (2^16) octets. Cette taille est une limite supérieure normale pour les paquets de données sur le réseau. D'autres applications peuvent utiliser des valeurs encore plus grandes de P_MAX, donc il est souhaitable que les algorithmes d'utilisation générale prennent en charge des valeurs supérieures.

Chaque algorithme AEAD DOIT accepter toutes les données associées d'une longueur entre zéro et A_MAX octets, inclus, où la valeur A_MAX est spécifique de cet algorithme. La valeur de A_MAX DOIT être supérieure à zéro, et DEVRAIT être au moins de 65 536 (2^16) octets. D'autres applications peuvent utiliser des valeurs encore plus grandes de A_MAX, donc il est souhaitable que les algorithmes d'utilisation générale prennent en charge des valeurs supérieures.

Chaque algorithme AEAD DOIT accepter tout nom occasionnel d'une longueur entre N_MIN et N_MAX octets, inclus, où les valeurs de N_MIN et N_MAX sont spécifiques de cet algorithme. Les valeurs de N_MAX et N_MIN PEUVENT être égales. Chaque algorithme DEVRAIT accepter un nom occasionnel d'une longueur de douze (12) octets. Les algorithmes aléatoires ou à états pleins, qui sont décrits plus loin, PEUVENT avoir une valeur N_MAX de zéro.

Un algorithme AEAD PEUT structurer son résultat de texte chiffré de toutes façons ; par exemple, le texte chiffré peut incorporer une étiquette d'authentification. Chaque algorithme DEVRAIT choisir une structure convenable pour un traitement efficace.

Un algorithme de chiffrement authentifié PEUT incorporer ou utiliser une source aléatoire, par exemple, pour la génération d'une valeur d'initialisation interne qui est incorporée dans le résultat du texte chiffré. Un algorithme AEAD de cette sorte est dit aléatoire ; noter cependant que seul le chiffrement est aléatoire et le déchiffrement est toujours déterministe. Un algorithme aléatoire PEUT avoir une valeur de N_MAX égale à zéro.

Un algorithme de chiffrement authentifié PEUT incorporer des informations d'état internes qui sont conservées entre les invocations de l'opération de chiffrement, par exemple, pour permettre la construction de valeurs distinctes utilisées comme noms occasionnels internes par l'algorithme. Un algorithme AEAD de cette sorte est dit à états pleins. Cette méthode pourrait être utilisée par un algorithme pour fournir une bonne sécurité même quand l'application entre des noms occasionnels de longueur zéro. Un algorithme à états pleins PEUT avoir une valeur de N_MAX égale à zéro.

La spécification d'un algorithme AEAD DOIT inclure les valeurs de K_LEN, P_MAX, A_MAX, N_MIN, et N_MAX définies ci-dessus. De plus, elle DOIT spécifier le nombre d'octets dans le plus grand texte chiffré possible, qu'on note C_MAX.

Chaque algorithme AEAD DOIT donner une description de la longueur du texte source par rapport au texte chiffré. Cette relation NE DOIT PAS dépendre de paramètres externes, comme un paramètre de force d'authentification (par exemple, la longueur de l'étiquette d'authentification). Cette sorte de dépendance compliquerait l'utilisation de l'algorithme en créant une situation dans laquelle les informations provenant du registre AEAD ne seraient pas suffisantes pour assurer l'interopérabilité.

Chaque spécification d'algorithme AEAD DEVRAIT décrire quelle dégradation de la sécurité résulterait d'une réutilisation involontaire d'une valeur de nom occasionnel.

Chaque spécification d'algorithme AEAD DEVRAIT fournir une référence à une analyse de sécurité détaillée. Le présent document ne spécifie pas un modèle de sécurité particulier, parce que plusieurs modèles différents ont été utilisés dans la littérature. L'analyse de sécurité DEVRAIT définir ou faire référence à un modèle de sécurité.

Un algorithme qui est aléatoire ou à états pleins, comme défini ci-dessus, DEVRAIT se décrire en utilisant ces termes.

5. Algorithmes AEAD

Cette Section définit quatre algorithmes AEAD ; deux se fondent sur AES GCM, deux se fondent sur AES CCM. Chaque paire inclut un algorithme avec une taille de clé de 128 bits et un d'une taille de clé de 256 bits.

5.1 AEAD_AES_128_GCM

L'algorithme de chiffrement authentifié AEAD_AES_128_GCM fonctionne comme spécifié dans [GCM], en utilisant AES-128 comme chiffrement de bloc, en fournissant la clé, le nom occasionnel, le texte source, et les données associées à ce mode de fonctionnement. Une étiquette d'authentification d'une longueur de 16 octets (128 bits) est utilisée. Le texte chiffré AEAD_AES_128_GCM est formé en ajoutant l'étiquette d'authentification fournie comme résultat de l'opération de chiffrement GCM au texte chiffré qui est le résultat de cette opération. Des cas d'essai sont fournis dans l'appendice à [GCM]. Les longueurs d'entrée et de sortie sont les suivantes :

K_LEN fait 16 octets,
P_MAX fait 2^36 - 31 octets,
A_MAX fait 2^61 - 1 octets,
N_MIN et N_MAX font tous deux 12 octets, et
C_MAX fait 2^36 - 15 octets.

Un texte chiffré AEAD_AES_128_GCM fait exactement 16 octets de plus que son texte source correspondant.

Une analyse de la sécurité de GCM est disponible dans [MV04].

5.1.1 Réutilisation de nom occasionnel

La réutilisation involontaire du même nom occasionnel par deux invocations de l'opération de chiffement GCM, avec la même clé, mais des valeurs distinctes de texte source, met en jeu la confidentialité du texte source protégé dans ces deux invocations, et met en danger la protection de l'authenticité et de l'intégrité fournies par cette clé. Pour cette raison, GCM devraient seulement être utilisé quand l'unicité du nom occasionnel peut être assurée. La caractéristique de conception qu'utilise GCM pour réaliser une latence minimale cause des vulnérabilités sur les utilisations suivantes de la clé. Noter qu'il est acceptable d'entrer la même valeur de nom occasionnel plusieurs fois dans l'opération de déchiffrement.

Les conséquences pour la sécurité sont assez sérieuses si un attaquant observe deux textes chiffrés qui ont été créés en utilisant les mêmes valeurs de nom occasionnel et de clé, sauf si le texte source et les valeurs des AD dans les deux invocations de l'opération de chiffrement étaient identiques. D'abord, une perte de confidentialité s'ensuit parce que il va être capable de reconstruire l'opération OU exclusif au bit près des deux valeurs de texte source. Ensuite, une perte d'intégrité s'ensuit parce que l'attaquant va être capable de récupérer la clé de hachage interne utilisée pour assurer l'intégrité des données. La connaissance de cette clé rend triviales les falsifications suivantes.

5.2 AEAD_AES_256_GCM

Cet algorithme est identique à AEAD_AES_128_GCM, mais avec les différences suivantes :
K_LEN fait 32 octets, au lieu de 16 octets, et

AES-256 GCM est utilisé à la place de AES-128 GCM.

5.3 AEAD_AES_128_CCM

L'algorithme de chiffrement authentifié AEAD_AES_128_CCM fonctionne comme spécifié dans [CCM], en utilisant AES-128 comme chiffrement de bloc, en fournissant la clé, un nom occasionnel, les données associées, et le texte source à ce mode de fonctionnement. Le formatage et la fonction de génération de compteur sont comme spécifieé à l'Appendice A de cette référence, et les valeurs des paramètres identifiées dans cet appendice sont les suivantes :

la longueur du nom occasionnel n est 12,
la longueur de l'étiquette t est 16, et
la valeur de q est 3.

Une étiquette d'authentification d'une longueur de 16 octets (128 bits) est utilisée. Le texte chiffré AEAD_AES_128_CCM est formé en ajoutant l'étiquette d'authentification fournie comme résultat de l'opération de chiffrement CCM au texte chiffré qui est le résultat de cette opération. Des cas d'essai sont fournis dans [CCM]. Les longueurs d'entrée et de sortie sont comme suit :

K_LEN fait 16 octets,
P_MAX fait 2^24 - 1 octets,
A_MAX fait 2^64 - 1 octets,
N_MIN et N_MAX font tous deux 12 octets, et
C_MAX fait 2^24 + 15 octets.

Un texte chiffré AEAD_AES_128_CCM est exactement plus long de 16 octets que son texte source correspondant.

Une analyse de la sécurité de AES CCM est disponible dans [J02].

5.3.1 Réutilisation de nom occasionnel

La réutilisation involontaire du même nom occasionnel par deux invocations de l'opération de chiffement CCM, avec la même clé, met en danger la sécurité des messages traités avec ces invocations. Une perte de confidentialité s'ensuit parce que un adversaire va être capable de reconstruire le OU exclusif au bit près des deux valeurs de texte source.

5.4 AEAD_AES_256_CCM

Cet algorithme est identique à AEAD_AES_128_CCM, mais avec les différences suivantes :

K_LEN fait 32 octets, au lieu de 16, et
AES-256 CCM est utilisé à la place de AES-128 CCM.

6. Considérations relatives à l'IANA

L'Autorité d'allocation des numéros de l'Internet (IANA, *Internet Assigned Numbers Authority*) a défini le "Registre AEAD" décrit ci-dessous. Un concepteur d'algorithmes PEUT enregistrer un algorithme afin de faciliter son utilisation. Les ajouts au registre AEAD exigent qu'une spécification soit documentée dans une RFC ou autre référence permanente et directement disponible, en détails suffisants pour que l'interopérabilité entre des mises en œuvre indépendantes soit possible. Chaque entrée dans le registre contient les éléments suivants :

un nom bref, comme "AEAD_AES_128_GCM", qui commence par la chaîne "AEAD",
un numéro positif, et
une référence à une spécification qui définit complètement un algorithme AEAD et fournit des cas d'essai qui peuvent être utilisés pour vérifier la correction de la mise en œuvre.

Les demandes d'ajout d'une entrée au registre DOIVENT inclure le nom et la référence. Le numéro est alloué par l'IANA. Ces allocations de numéro DEVRAIENT utiliser le plus petit numéro positif disponible. Les soumettants DEVRAIENT avoir leurs demandes revues par le groupe de recherche Crypto Forum de l'IRTF (CFRG) à cfrg@ietf.org. Les demandeurs intéressés qui ne sont pas familiarisés avec les processus de l'IANA devraient visiter le site http://www.iana.org.

Les numéros entre 32 768 (binaire 1000000000000000) et 65 535 (binaire 1111111111111111) inclus, ne seront pas alloués par l'IANA, et sont réservés pour utilisation privée ; aucune tentative ne sera faite pour empêcher plusieurs sites d'utiliser la même valeur de façons différentes (et incompatibles) [RFC2434].

L'IANA a ajouté les entrées suivantes au registre AEAD :

Nom	Référence	Identifiant numérique
AEAD_AES_128_GCM	paragraphe 5.1	1
AEAD_AES_256_GCM	paragraphe 5.2	2
AEAD_AES_128_CCM	paragraphe 5.3	3
AEAD_AES_256_CCM	paragraphe 5.4	4

Un enregistrement par l'IANA d'un AEAD ne constitue pas une approbation de cet algorithme ou de sa sécurité.

7. Autres considérations

Faire des essais directs d'un algorithme de chiffrement AEAD aléatoire en utilisant des cas d'essai avec des entrées et sorties fixes n'est pas possible, car le processus de chiffrement est non déterministe. Cependant, il est possible de faire des essais sur un algorithme AEAD aléatoire en utilisant la technique suivante. L'algorithme de déchiffrement authentifié est déterministe, et il peut être essayé directement. L'algorithme de chiffrement authentifié peut être essayé en chiffrant un texte source, en déchiffrant le texte chiffré résultant, et en comparant le texte source original au texte source post-déchiffrement. La combinaison de ces deux essais couvre les deux algorithmes de chiffrement et de déchiffrement.

Les algorithmes AEAD choisis reflètent ceux qui ont déjà été adoptés par des normes. La question est ouverte de savoir quels autres algorithmes AEAD devraient être ajoutés. De nombreuses variantes des algorithmes de base sont possibles, chacune avec ses propres avantages. Bien qu'il soit souhaitable d'admettre tous les algorithmes qui se trouvent être utiles en pratique, il est aussi souhaitable de limiter le nombre total d'algorithmes enregistrés. La spécification actuelle exige qu'un algorithme enregistré fournisse une spécification complète et un ensemble de données de validation ; il est espéré que ces exigences établissent de façon appropriée les critères d'admission.

Il peut être désirable de définir un algorithme AEAD qui utilise la composition générique avec la méthode "chiffrement puis MAC" [BN00], combinant un algorithme de chiffrement commun, comme CBC [MODES], avec un code d'authentification de message, comme HMAC-SHA1 [RFC2104] ou AES CMAC [CMAC]. Un algorithme AEAD de cette sorte reflèterait la meilleure pratique actuelle, et pourrait être plus facilement supporté par les crypto modules qui manquent de soutien pour les autres algorithmes AEAD.

8. Considérations sur la sécurité

Le présent document décrit des algorithmes de chiffrement authentifié, et donne des directives sur leur utilisation. Bien que ces algorithmes rendent plus facile, dans une certaine mesure, de concevoir une application cryptographique, on devrait se souvenir qu'une sécurité cryptographique forte est difficile à réaliser. Bien que les algorithmes AEAD soient assez utiles, ils ne font rien pour traiter les questions de génération de clé [RFC4086] et de gestion de clé [RFC4107].

Les algorithmes AEAD qui s'appuient sur des noms occasionnels distincts peuvent être inappropriés pour certaines applications ou pour certains scénarios. Les concepteurs d'applications devraient comprendre les exigences mentionnées au paragraphe 3.1.

Une mise en œuvre logicielle de l'opération de chiffrement AEAD dans un environnement de machine virtuelle (VM, *Virtual Machine*) pourrait involontairement réutiliser un nom occasionnel du fait d'un "repli" de la VM à un état antérieur [GR05]. Les applications sont invitées à documenter les problèmes potentiels pour aider l'utilisateur de l'application et la VM à éviter des fautes non intentionnelles de cette sorte. La possibilité existe qu'un attaquant puisse causer un repli de VM ; les menaces et les contremesures dans ce scénario font l'objet de recherches actives. Pour mémoire, on note qu'un attaquant qui peut déclencher un tel repli peut avoir déjà réussi à subvertir la sécurité du système, par exemple, en causant une erreur comptable.

L'enregistrement par l'IANA d'un algorithme AEAD NE DOIT PAS être regardé comme l'approbation de sa sécurité. De plus, le niveau de sécurité perçu d'un algorithme peut se dégrader au fils du temps, du fait des avancées de la cryptanalyse ou de la "Loi de Moore", c'est-à-dire, la diminution du coût des ressources de calcul au fil du temps.

9. Remerciements

De nombreux relecteurs ont fourni des commentaires précieux sur les projets antérieurs de ce document. Des discussions fructueuses ont eu lieu sur la liste de diffusion du groupe de travail Crypto Forum Research en 2006.

10. Références

10.1 Références normatives

[CCM] Dworkin, M., "NIST Special Publication 800-38C: The CCM Mode for Authentication and Confidentiality", U.S. National Institute of Standards and Technology, <http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C.pdf>.

[GCM] Dworkin, M., "NIST Special Publication 800-38D: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC.", U.S. National Institute of Standards and Technology, novembre 2007, <http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf>.

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

10.2 Références pour information

[BN00] Bellare, M. and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm", Proceedings of ASIACRYPT 2000, Springer-Verlag, LNCS 1976, pp. 531-545, 2002.

[BOYD] Boyd, C. and A. Mathuria, "Protocols for Authentication and Key Establishment", Springer 2003.

[CMAC] "NIST Special Publication 800-38B", <http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf>.

[EEM04] Bellare, M., Namprempre, C., and T. Kohno, "Breaking and provably repairing the SSH Authenticated encryption scheme: A case study of the Encode-then-Encrypt-and-MAC paradigm", ACM Transactions on Information and System Security, <http://www-cse.ucsd.edu/users/tkohno/papers/TISSEC04/>.

[GR05] Garfinkel, T. and M. Rosenblum, "When Virtual is Harder than Real: Security Challenges in Virtual Machine Based Computing Environments", Proceedings of the 10th Workshop on Hot Topics in Operating Systems, <http://www.stanford.edu/~talg/papers/HOTOS05/virtual-harder-hotos05.pdf>.

[J02] Jonsson, J., "On the Security of CTR + CBC-MAC", Proceedings of the 9th Annual Workshop on Selected Areas on Cryptography, 2002, <http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ccm/ccm-ad1.pdf>.

[MODES] Dworkin, M., "NIST Special Publication 800-38: Recommendation for Block Cipher Modes of Operation", U.S. National Institute of Standards et Technology, <http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf>.

[MV04] McGrew, D. and J. Viega, "The Security and Performance of the Galois/Counter Mode (GCM)", Proceedings of INDOCRYPT '04, décembre 2004, <http://eprint.iacr.org/2004/193>.

[R02] Rogaway, P., "Authenticated encryption with Associated-Data", ACM Conference on Computer and Communication Security (CCS'02), pp. 98-107, ACM Press, 2002, <http://www.cs.ucdavis.edu/~rogaway/papers/ad.html>.

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

[RFC2434] T. Narten et H. Alvestrand, "Lignes directrices pour la rédaction d'une section Considérations relatives à l'IANA dans les RFC", BCP 26, octobre 1998. *(Rendue obsolète par la* RFC5226)

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

[RFC4106] J. Viega, D. McGrew, "Utilisation du mode Galois/Compteur (GCM) dans une encapsulation IPsec de charge utile de sécurité (ESP)", juin 2005. *(P.S.)*

[RFC4107] S. Bellovin, R. Housley, "Lignes directrices pour la gestion des clés de chiffrement", juin 2005. (BCP0107)

[RFC4303] S. Kent, "Encapsulation de charge utile de sécurité dans IP (ESP)", décembre 2005. (*Remplace* RFC2406) *(P.S.)*

[RFC4309] R. Housley, "Utilisation du mode CCM de la norme de chiffrement évolué (AES) avec l'encapsulation de charge utile de sécurité (ESP) dans IPsec", décembre 2005. *(P.S.)*

Adresse de l'auteur

David A. McGrew
Cisco Systems, Inc.
510 McCarthy Blvd.
Milpitas, CA 95035
US
téléphone : (408) 525 8651
mél : mcgrew@cisco.com
URI : http://www.mindspring.com/~dmcgrew/dam.htm

Déclaration complète de droits de reproduction