Groupe de travail Réseau
Request for Comments : 2660
Catégorie : Expérimentale
Traduction Claude Brière de L'Isle

E. Rescorla, RTFM, Inc.
A. Schiffman, Terisa Systems, Inc.
août 1999

Protocole de transfert HyperTexte sécurisé

Statut de ce mémoire
Le présent mémoire définit un protocole expérimental pour la communauté de l'Internet. Il ne spécifie en aucune façon une norme de l'Internet. Des discussions et des suggestions pour son amélioration sont demandées. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé
Le présent mémoire décrit une syntaxe pour sécuriser les messages envoyés en utilisant le protocole de transfert hypertexte (HTTP, *Hypertext Transfer Protocol*) qui forme la base de la toile mondiale (WWW, *World Wide Web*). HTTP sécurisé (S-HTTP) fournit des services de sécurité applicables indépendamment pour la confidentialité, l'authenticité/intégrité et la non répudiabilité d'origine des transactions.

Le protocole met l'accent sur un maximum de souplesse de choix des mécanismes de gestion de clés, des politiques de sécurité et des algorithmes de chiffrement en prenant en charge la négociation des options entre les parties pour chaque transaction.

Table des Matières

1. Introduction

La Toile mondiale (WWW, *World Wide Web*) est un système hypermédia réparti qui a conquis une large acceptation parmi les utilisateurs de l'Internet. Bien que les navigateurs de la Toile mondiale prennent en charge d'autres protocoles d'application Internet préexistants, le protocole natif et principal utilisé entre les clients et serveurs WWW est le protocole de transfert hypertexte (HTTP) [RFC2616]. La facilité d'utilisation de la Toile a incité à son large emploi comme architecture client/serveur pour de nombreuses applications. Beaucoup de ces applications exigent que le client et le serveur soient capables de s'authentifier l'un l'autre et échangent de façon confidentielle des informations sensibles. La spécification HTTP d'origine avait seulement une prise en charge modeste pour les mécanismes cryptographiques appropriés pour de telles transactions.

HTTP sécurisé (S-HTTP) fournit des mécanismes de communication sûrs entre une paire client-serveur HTTP afin de permettre des transactions commerciales spontanées pour une large gamme d'applications. Notre intention est de fournir un protocole souple qui prenne en charge plusieurs modes de fonctionnement orthogonaux, des mécanismes de gestion de clés, des modèles de confiance, des algorithmes de chiffrement et des formats d'encapsulation à travers la négociation d'options entre les parties pour chaque transaction.

1.1 Résumé des caractéristiques

HTTP sécurisé est un protocole de communications sécurisé en mode message conçu pour être utilisé en conjonction avec HTTP. Il est conçu pour coexister avec le modèle de messagerie de HTTP et pour être facilement intégré aux applications HTTP.

HTTP sécurisé fournit divers mécanismes de sécurité aux clients et serveurs HTTP, fournissant des options de service de sécurité appropriées à la large gamme d'utilisations finales potentielles possibles pour la Toile mondiale. Le protocole fournit des capacités symétriques aux clients et aux serveurs (en ce qu'un traitement égal est assuré aux demandes et aux réponses, ainsi que pour les préférences des deux parties) tout en préservant le modèle de transaction et les caractéristiques de mise en œuvre de HTTP.
Plusieurs normes de format de message cryptographique peuvent être incorporées dans les clients et serveurs S-HTTP, en particulier, mais sans s'y limiter en principe, [RFC2630] et [MOSS]. S-HTTP prend en charge l'interopération entre diverses mises en œuvre, et est compatible avec HTTP. Les clients à capacité S-HTTP peuvent communiquer avec les serveurs qui ont oublié S-HTTP et vice-versa, bien que de telles transactions aient peu de chances d'utiliser les dispositifs de sécurité de S-HTTP.

S-HTTP n'exige pas de certificat de clé publique (ou des clés publiques) du côté client car il prend en charge les modes de fonctionnement seulement en clé symétrique.

Ceci est significatif parce que cela veut dire que des transactions privées spontanées peuvent survenir sans exiger que les utilisateurs individuels aient une clé publique établie. Alors que S-HTTP est capable de tirer parti des infrastructures de certification que l'on rencontre partout, son déploiement ne les exige pas.

S-HTTP prend en charge les transactions sécurisées de bout en bout, à la différence du mécanisme d'autorisation du HTTP d'origine qui exige que le client tente l'accès et soit refusé avant que le mécanisme de sécurité soit employé. Les clients peuvent être encouragés à initier une transaction sûre (normalement en utilisant des informations fournies dans les en-têtes de message) ; cela peut être utilisé, par exemple, à l'appui du chiffrement de formulaires remplis. Avec S-HTTP, aucune donnée sensible ne doit jamais être envoyée en clair sur le réseau.

S-HTTP procure une souplesse complète des algorithmes, modes et paramètres cryptographiques .La négociation des options est utilisée pour permettre aux clients et serveurs de s'accorder sur les modes de transaction (par exemple, si la demande devrait être signée ou chiffrée ou les deux – de même que pour la réponse ?) sur les algorithmes de chiffrement (RSA ou DSA pour la signature, DES ou RC2 pour le chiffrement, etc.) et sur le choix des certificats (prière de signer avec votre "certificat de vidéo Block-buster").

S-HTTP tente d'éviter de présumer un modèle de confiance particulier, bien que ses concepteurs admettent un effort conscient pour faciliter une confiance hiérarchique multi-racines, et prévoient que les principaux puissent voir de nombreux certificats de clé publique.

S-HTTP diffère de l'authentification par résumé, décrite dans la [RFC2617] en ce qu'elle fournit la prise charge de la cryptographie à clé publique et par conséquent une capacité de signature numérique, ainsi que la confidentialité.

1.2 Changements

Le présent document décrit S-HTTP/1.4. Il diffère des mémoires précédents en ce qu'il prend en charge la syntaxe de message cryptographique (CMS, *Cryptographic Message Syntax*) [RFC2630], un successeur de PKCS-7 ; et donc prend maintenant en charge les systèmes de chiffrement Diffie-Hellman et la norme de signature numérique du NIST. La CMS utilisée en mode RSA est compatible au bit près avec PKCS-7.

1.3 Modèle de traitement

1.3.1 Préparation du message

La création d'un message S-HTTP peut être vue comme une fonction avec trois entrées :

1. Le message en clair. C'est soit un message HTTP, soit quelque autre objet de données. Noter que comme le message en clair est porté de façon transparente, en-têtes et tout, toute version de HTTP peut être portée dans une enveloppe S-HTTP.
2. Les préférences cryptographiques du receveur et le matériel de chiffrement. Ceci est soit spécifié explicitement par le receveur, soit l'objet d'un ensemble de préférences par défaut.
3. Les préférences cryptographiques de l'envoyeur et le matériel de chiffrement. Cette entrée à la fonction peut être vue comme implicite car elle n'existe que dans la mémoire de l'envoyeur.

Pour créer un message S-HTTP, l'envoyeur intègre ensuite ses préférences avec celles du receveur. Le résultat est une liste d'améliorations cryptographiques à appliquer et du matériel de chiffrement à utiliser pour les appliquer. Cela peut exiger une intervention de l'utilisateur. Par exemple, il peut y avoir plusieurs clés disponibles pour signer le message. (Voir le paragraphe 3.2.4.9.3 pour les détails.) En utilisant ces données, l'envoyeur applique les améliorations au message en clair pour créer le message S-HTTP.

Les étapes du traitement nécessaires pour transformer le message en clair en message S-HTTP sont décrites dans les Sections 2 et 3. Ces étapes de traitement nécessaires pour fusionner les préférences de l'envoyeur et du receveur sont décrites au paragraphe 3.2.

1.3.2 Récupération du message

La récupération d'un message S-HTTP peut être vue comme une fonction de quatre entrées distinctes :

1. Le message S-HTTP.
2. Les préférences cryptographiques déclarées du receveur et le matériel de chiffrement. Le receveur a l'opportunité de se souvenir des préférences cryptographiques qu'il a fournies afin de déréférencer ce document.
3. Les préférences cryptographiques actuelles du receveur et le matériel de chiffrement.
4. Les options cryptographiques précédemment déclarées de l'envoyeur. L'envoyeur peut avoir déclaré qu'il effectuerait certaines opérations cryptographiques dans ce message. (Voir les Sections 4 et 5 pour les détails.)

Pour récupérer un message S-HTTP, le receveur doit lire les en-têtes pour découvrir quelles transformations cryptographiques ont été effectuées sur le message, puis retirer les transformations en utilisant une certaine combinaison du matériel de chiffrement de l'envoyeur et du receveur, tout en prenant note des améliorations qui ont été appliquées.

Le receveur peut aussi choisir de vérifier que les améliorations appliquées correspondent à la fois aux améliorations dont l'envoyeur avait dit qu'elles s'appliqueraient (entrée 4 ci-dessus) et que le receveur demandait (entrée 2 ci-dessus) ainsi que les préférences actuelles pour voir si le message S-HTTP a été transformé de façon appropriée. Ce processus peut exiger une interaction avec l'usager pour vérifier que les améliorations sont acceptables pour l'usager. (Voir le paragraphe 6.4 pour les

détails.)

1.4 Modes de fonctionnement

La protection du message peut être assurée sur trois axes orthogonaux : signature, authentification, et chiffrement. Tout message peut être signé, authentifié, chiffré, ou avoir toute combinaison des trois (incluant pas de protection).

Plusieurs mécanismes de gestion de clé sont pris en charge, incluant les secrets partagés manuellement de style mot de passe, et l'échange de clé à clé publique. En particulier, des dispositions ont été prises pour les clés de session symétriques (dans une transaction antérieure ou hors bande) afin d'envoyer des messages confidentiels à ceux qui n'ont pas de paire de clés publiques.

De plus, un mécanisme de défi-réponse ("nom occasionnel") est fourni pour permettre aux parties de s'assurer elles-mêmes de la fraîcheur des transactions.

1.4.1 Signature

Si l'amélioration de signature numérique est appliquée, un certificat approprié peut soit être attaché au message (éventuellement avec une chaîne de certificats) soit l'envoyeur peut s'attendre à ce que le receveur obtienne le certificat (la chaîne de certificats) requis de façon indépendante.

1.4.2 Échange de clés et chiffrement

Pour la prise en charge du chiffrement en vrac, S-HTTP définit deux mécanismes de transfert de clés, l'un qui utilise l'échange de clé enveloppé à clé publique et l'autre avec des clé arrangées en externe.

Dans le premier cas, le paramètre de système de chiffrement à clé symétrique est passé chiffré sous la clé publique du receveur. Dans le second mode, on chiffre le contenu en utilisant une clé de session pré arrangée, avec des informations d'identification spécifiées dans une des lignes d'en-tête.

1.4.3 Intégrité du message et authentification de l'envoyeur

HTTP sécurisé donne le moyen de vérifier l'intégrité du message et l'authenticité de l'envoyeur pour un message via le calcul d'un code d'authentification de message (MAC, *Message Authentication Code*) calculé comme un hachage chiffré sur le document en utilisant un secret partagé – qui pourrait éventuellement avoir été arrangé d'un certain nombre de façons, par exemple, un arrangement manuel ou une gestion de clé 'dans la bande'. Cette technique n'exige ni l'utilisation d'un chiffrement à clé publique ni le chiffrement.

Ce mécanisme est aussi utile pour les cas où il est approprié de permettre aux parties de s'identifier l'une l'autre de façon fiable dans une transaction sans fournir la non répudiabilité (via un tiers) pour les transactions elles-mêmes. La fourniture de ce mécanisme est motivée par le biais que l'action de "signer" une transaction devrait être explicite et consciente pour l'utilisateur, tandis que de nombreux besoins d'authentification (c'est-à-dire, de contrôle d'accès) peuvent être satisfaits avec un mécanisme moins lourd qui conserve les avantages d'adaptabilité de la cryptographie à clé publique pour l'échange de clés.

1.4.4 Fraîcheur

Le protocole fournit un simple mécanisme de mise au défi-réponse qui permet aux deux parties de s'assurer de la fraîcheur des transmissions. De plus, la protection de l'intégrité fournie aux en-têtes HTTP permet aux mises en œuvre de considérer l'en-tête Date: permis dans les messages HTTP comme un indicateur de fraîcheur, lorsque approprié (bien que cela exige que les mises en œuvre aient des tolérances pour l'écart maximum d'horloge entre les parties, ce que nous avons choisi de ne pas spécifier).

1.5 Options de mise en œuvre

Pour encourager une large adoption de documents sécurisés pour la Toile mondiale face à la large portée des exigences d'application, de la variabilité de la sophistication des utilisateurs, et des contraintes disparates de mise en œuvre, HTTP sécurisé fournit délibérément une grande variété d'options de mise en œuvre. Voir à la Section 8 les recommandations et exigences de mise en œuvre.

2. Format de message

Syntaxiquement, les messages HTTP sécurisés sont les mêmes que ceux de HTTP, consistant en une ligne de commande ou d'état suivie par des en-têtes et d'un corps. Cependant, la gamme d'en-têtes est différente et les corps sont normalement améliorés cryptographiquement.

2.1 Conventions de notation

Le présent document utilise le BNF augmenté de HTTP [RFC2616]. Se référer à ce document pour une description de la syntaxe.

2.2 Ligne de demande

Afin de différencier les messages S-HTTP des messages HTTP et permettre un traitement particulier, la ligne de demande devrait utiliser la méthode spéciale "Secure" et utiliser la désignation de protocole "Secure-HTTP/1.4". Par conséquent, le traitement de S-HTTP et de HTTP peut être mélangé sur le même accès TCP, par exemple l'accès 80. Afin d'empêcher des fuites d'informations potentiellement sensibles, les URI de demande devraient être "*". Par exemple,

Secure * Secure-HTTP/1.4

Lorsque on communique via un mandataire, l'URI de demande devrait consister en l'URI absolu. Normalement, la section de chemin réel devrait être remplacée par "*" pour minimiser les informations passées en clair, (par exemple, http://www.terisa.com/*) ; les mandataires devraient retirer la quantité appropriée de ces informations pour minimiser la menace d'analyse de trafic. Voir au paragraphe 7.2.2.1 une situation où il est approprié de fournir plus d'informations.

2.3 Ligne d'état

Les réponses S-HTTP devraient utiliser la désignation de protocole "Secure-HTTP/1.4". Par exemple :

Secure-HTTP/1.4 200 OK

Noter que l'état dans la ligne de réponse S-HTTP n'indique rien sur le succès ou l'échec de la demande HTTP non enveloppée. Les serveurs devraient toujours utiliser 200 OK lorsque le traitement S-HTTP est réussi. Cela empêche les analyses de la réussite ou de l'échec pour toute demande, ce que le receveur correct peut déterminer à partir des données encapsulées. Toutes les variantes devraient être acceptées.

2.4 Lignes d'en-tête de HTTP sécurisé

Les lignes d'en-tête décrites dans ce paragraphe vont dans l'en-tête du message S-HTTP. Toutes sauf 'Content-Type' et 'Content-Privacy-Domain' sont facultatives. Le corps du message devra être séparé du bloc d'en-tête par deux CRLF successifs.

Toutes les données et tous les champs dans les lignes d'en-tête devraient être traités comme insensibles à la casse sauf spécification contraire. Les espaces blanches linéaires [RFC0822] devraient être utilisées seulement comme jeton séparateur sauf mention contraire. Les longues lignes d'en-tête peuvent passer à la ligne dans le style de la [RFC0822].

Le présent document se réfère collectivement au bloc d'en-tête suivant la ligne de demande/réponse S-HTTP et précédant les CRLF successifs comme "en-têtes S-HTTP".

2.4.1 Content-Privacy-Domain

Les deux valeurs définies par le présent document sont 'MOSS' et 'CMS'. CMS se réfère à l'amélioration de confidentialité spécifiée au paragraphe 2.6.1. MOSS se réfère au format défini dans la [RFC1847] et la [RFC1848].

2.4.2 Content-Type pour la CMS

Dans les conditions normales, le contenu terminal encapsulé (après que toutes les améliorations de confidentialité ont été retirées) sera un message HTTP. Dans ce cas, il devra y avoir une ligne Content-Type disant :

Content-Type: message/http

Le type de contenu message/http est défini dans la RFC2616.

Si le message interne est un message S-HTTP, le type de contenu devra être 'application/s-http'. (Voir sa définition à l'Appendice A.)

Il est prévu que ces types soient enregistrés auprès de l'IANA comme types de contenu MIME.

Le contenu terminal peut être d'un autre type pourvu que le type soit indiqué de façon appropriée par l'utilisation d'une ligne d'en-tête Content-Type appropriée. Dans ce cas, les champs d'en-tête pour l'encapsulation du contenu terminal s'appliquent au contenu terminal (les 'en-têtes finaux'). Mais dans tous les cas, les en-têtes finaux devraient eux-mêmes toujours être encapsulés dans S-HTTP, afin que les en-têtes S-HTTP/HTTP applicables ne soient jamais passés sans améliorations.

L'encapsulation S-HTTP de données non HTTP est un mécanisme utile pour passer des données pré améliorées (en particulier des données pré signées) sans exiger que les en-têtes HTTP eux-mêmes soient pré améliorés.

2.4.3 Content-Type pour MOSS

Le type de contenu pour MOSS devra être un type de contenu MIME acceptable décrivant le traitement cryptographique à appliquer (par exemple multipart/signed). Le type de contenu du contenu interne est décrit dans la ligne de type de contenu correspondant à ce contenu interne, et pour les messages HTTP, il devra être 'message/http'.

2.4.4 Prearranged-Key-Info

Cette ligne d'en-tête est destinée à porter des informations sur une clé qui a été arrangée en dehors du format cryptographique interne. Une utilisation de ceci est de permettre la communication dans la bande de clés de session pour le chiffrement de retour dans le cas où une des parties n'a pas une paire de clés. Cependant, ceci devrait aussi être utile dans le cas où les parties choisissent d'utiliser un autre mécanisme, par exemple, une liste de clés à usage unique.

La présente spécification définit deux méthodes pour échanger les clés désignées, Inband *(dans la bande)*, Outband *(hors bande)*. Inband indique que la clé de session a été échangée précédemment, en utilisant un en-tête Key-Assign de la méthode correspondante. Les arrangements Outband impliquent que les agents ont un accès externe aux matériaux de clés correspondant à un certain nom, probablement via un accès à une base de données ou peut-être fournis immédiatement par un usager à partir d'une entrée au clavier. La syntaxe de la ligne d'en-tête est :

Prearranged-Key-Info = "Prearranged-Key-Info" ":" Hdr-Cipher "," CoveredDEK "," CoverKey-ID
CoverKey-ID = method ":" key-name
CoveredDEK = *HEX
method = "inband" | "outband"

Bien que les chiffrements à enchaînement exigent une valeur d'initialisation (IV, *Initialization Vector*) [FIPS-81] pour commencer l'enchaînement, cette information n'est pas portée par ce champ. Elle devrait plutôt être passée en interne au format cryptographique utilisé. De même, le chiffrement en vrac utilisé est spécifié de cette façon.

<Hdr-Cipher> devrait être le nom du chiffrement de bloc utilisé pour chiffrer la clé de session (voir le paragraphe 3.2.4.7).

<CoveredDEK> est la clé de chiffrement des données protégées (autrement dit la clé de transaction) sous laquelle le message encapsulé a été chiffré. Elle devrait être générée de façon appropriée (au hasard) par l'agent d'envoi, puis chiffrée sous le couvert de la clé négociée (autrement dit, la clé de session) en utilisant le chiffrement d'en-tête indiqué, et ensuite convertie en hexadécimal.

Pour éviter les collisions de noms, les espaces de noms de clé couverte doivent être tenus séparément par l'hôte et par l'accès.

Noter que certains Content-Privacy-Domains, notamment de probables futures révisions de MOSS et CMS, pourraient prendre en charge la gestion de clés symétriques.

Le champ Prearranged-Key-Info n'a pas besoin d'être utilisé dans de telles circonstances. La syntaxe native est préférée. Les

clés échangées avec Key-Assign, peuvent cependant être utilisées dans cette situation.

2.4.5 MAC-Info

Cet en-tête est utilisé pour fournir une vérification d'authenticité du message (MAC), assurant à la fois l'authentification et l'intégrité du message, calculée à partir du texte du message, de l'heure (facultatif -- pour empêcher l'attaque de répétition) et d'un secret partagé entre le client et le serveur. La MAC devrait être calculée sur le contenu encapsulé du message S-HTTP. S-HTTP/1.1 définissait que les MAC devraient être calculées en utilisant l'algorithme suivant ('||' signifie l'enchaînement) :

MAC = hex(H(Message||[<heure>]||<clé partagée>))

L'heure devrait être représentée par une quantité de 32 bits non signée représentant les secondes depuis le 1er janvier 1970 à 00:00:00 GMT (l'époque UNIX) dans l'ordre des octets du réseau. Le format de la clé partagée est une affaire locale.

Des recherches récentes [VANO95] ont démontré quelques faiblesses de cette approche, et le présent mémoire introduit une nouvelle construction, déduite de la [RFC2104]. Au nom de la rétro compatibilité, on conserve les constructions précédentes avec les mêmes noms qu'auparavant ; on introduit aussi une nouvelle série de noms (voir les noms au paragraphe 3.2.4.8) qui obéissent à une construction différente (qu'on espère plus solide).

HMAC = hex(H(K' ^ pad2 || H(K' ^ pad1 ||[<heure>]|| Message)))

(^ signifie OUX au bit près)
pad1 = l'octet 0x36 répété assez de fois pour remplir un bloc d'entrée de hachage (c'est-à-dire,. 64 fois pour MD5 et SHA-1).
pad2 = l'octet 0x5c répété assez de fois pour remplir un bloc d'entrée de hachage.
K' = H(<clé partagée>)
La construction HMAC originale est à utiliser avec une clé d'une longueur égale à celle du résultat du hachage. Bien qu'il soit considéré comme sûr d'utiliser une clé d'une longueur différente (noter que la force ne peut pas être augmentée au delà de la longueur de la fonction de hachage elle-même, mais peut être réduite par l'utilisation d'une clé plus courte [KRAW96b]) on hache la clé originale pour permettre l'utilisation de clés partagées (par exemple des phrases de passe) plus longues que le hachage. On notera (bien que ce soit évident) que cette technique n'augmente pas la force des clés courtes.

Le format de la ligne MAC-Info est :

MAC-Info = "MAC-Info" ":" [hex-time], hash-alg, hex-hash-data, key-spec
hex-time = <nombre de secondes non signé depuis l'époque Unix représentée en HEX>
hash-alg = <algorithmes de hachage du paragraphe 3.2.4.8>
hex-hash-data = <calcul comme décrit ci-dessus représenté en HEX>
Key-Spec = "null" | "dek" | Key-ID

Les Key-ID peuvent se référer soit aux limites de clés en utilisant la ligne d'en-tête Key-Assign, soit à ces limites de la même façon que dans la méthode Outband décrite plus loin. L'utilisation d'une spécification de clé 'Null' implique qu'une clé de longueur zéro a été utilisée, et que donc la MAC représente simplement un hachage du texte du message et (facultativement) de l'heure. La spécification de la clé spéciale 'dek' se réfère à la clé d'échange de données (DEK, *Data Exchange Key*) utilisée pour chiffrer le corps du message qui suit (c'est une erreur d'utiliser la spécification de clé DEK dans des situations où le corps du message qui suit n'est pas chiffré).

Si l'heure est omise de la ligne MAC-Info, elle ne devrait simplement pas être incluse dans le hachage. Noter que cette ligne d'en-tête peut être utilisée pour fournir un équivalent plus évolué du mode d'authentification de base de l'HTTP original en ce qu'il peut être demandé à l'usager de fournir un nom d'utilisateur et un mot de passe. Cependant, le mot de passe reste confidentiel et l'intégrité du message peut être assurée. De plus, ceci peut être accompli sans aucune espèce de chiffrement.

En plus, MAC-Info permet une vérification rapide de l'intégrité du message (moyennant la perte de la non répudiabilité) pour les messages, pourvu que les participants partagent une clé (éventuellement passée en utilisant Key-Assign dans un message antérieur). Noter que certains Content-Privacy-Domains, notamment dans de probables futures révisions de MOSS et CMS pourraient prendre en charge la protection symétrique de l'intégrité. Le champ MAC-Info n'a pas besoin d'être utilisé dans de telles circonstances. La syntaxe native est préférée. Les clés échangées avec Key-Assign, peuvent cependant être utilisées dans cette situation.

2.5 Contenu

Le contenu du message dépend en grande partie des valeurs des champs Content-Privacy-Domain *(domaine de confidentialité*

du contenu) et Content-Transfer-Encoding *(codage de transfert du contenu)*.

Pour un message CMS avec un Content-Transfer-Encoding de '8BIT', le contenu devrait simplement être le message CMS lui-même.

Si le Content-Privacy-Domain est MOSS, le contenu devrait consister en une multipartie Sécurité MOSS comme décrit dans la RFC1847.

Il est prévu qu'une fois les améliorations de confidentialité retirées, le contenu résultant (éventuellement protégé) sera une demande HTTP normale. Autrement, le contenu peut être un autre message S-HTTP, et dans ce cas les améliorations de confidentialité devraient être non enveloppées jusqu'à ce que soit obtenu le contenu en clair, sinon les améliorations de confidentialité ne peuvent plus être retirées. (Cela permet d'incorporer des améliorations, comme celles de signature et d'enveloppement séquentiels.) Pourvu que toutes les améliorations puissent être retirées, le contenu final dépouillé des améliorations devrait être une demande (ou réponse) HTTP valide sauf spécification contraire par la ligne Content-Type.

Noter que cette encapsulation récurrente de messages permet une application (ou une suppression) potentielle d'améliorations de sécurité pour le bénéfice des intermédiaires qui peuvent être intégrés à la transaction entre un client et un serveur (par exemple, un mandataire qui exige l'authentification du client). Le paragraphe 7.2.1 décrit comment de tels intermédiaires devraient indiquer un tel traitement.

2.6 Options de format d'encapsulation

2.6.1 Content-Privacy-Domain: CMS

Content-Privacy-Domain 'CMS' suit le format de la norme de CMS (voir l'Appendice A).

La protection du message peut se faire selon deux axes orthogonaux : la signature et le chiffrement. Tout message peut être signé, chiffré, les deux, ou ni l'un ni l'autre. Noter que le mode de protection 'auth' de S-HTTP est fourni indépendamment du codage CMS via l'en-tête MAC-Info du paragraphe 2.3.6 car la CMS ne prend pas en charge un type 'KeyDigestedData', bien qu'elle prenne en charge un type 'DigestedData'.

2.6.1.1 Signature

Cette amélioration utilise le type 'SignedData' de CMS. Lorsque des signatures numériques sont utilisées, un certificat approprié peut être attaché au message (éventuellement avec une chaîne de certificat) comme spécifié dans la CMS ou bien l'envoyeur peut attendre que le receveur obtienne son certificat (et/ou chaîne) de façon indépendante. Noter qu'une instance explicitement permise de cela est un certificat signé avec le composant privé correspondant au composant public attesté. On va appeler cela un certificat auto signé. La valeur, s'il en est, à accorder à un tel certificat est une affaire purement locale. En tous cas, un message purement signé est précisément conforme à la CMS.

2.6.1.2 Chiffrement

2.6.1.2.1 Chiffrement -- normal, clé publique

Cette amélioration est effectuée précisément comme un enveloppement (utilisant l'un ou l'autre type de ' EnvelopedData') sous CMS. Un message chiffré de cette façon, signé ou autrement, est conforme à la CMS. Pour avoir un message qui soit à la fois signé et chiffré, on crée simplement la production CMS SignedData et on l'encapsule dans EnvelopedData comme décrit dans la CMS.

2.6.1.2.2 Chiffrement – clé pré arrangée

Ceci utilise le type 'EncryptedData' de CMS. Dans ce mode, on chiffre le contenu en utilisant une DEK chiffrée sous couvert d'une clé de session pré arrangée (on expose plus loin comment cette clé peut être échangée) avec les informations d'identification de clé spécifiées sur une des lignes d'en-tête. La IV est dans le type EncryptedContentInfo de l'élément EncryptedData. Pour avoir un message qui soit à la fois signé et chiffré, on crée simplement la production CMS SignedData et on l'encapsule dans EncryptedData comme décrit dans la CMS.

2.6.2 Content-Privacy-Domain: MOSS

Le corps du message devrait être conforme à MIME avec un type de contenu qui corresponde à la ligne Content-Type dans les

en-têtes S-HTTP. Les messages chiffrés devraient utiliser multipart/encrypted. Les messages signés devraient utiliser multipart/signed. Cependant, comme multipart/signed ne porte pas de matériel de chiffrement, il est acceptable d'utiliser multipart/mixed où la première partie est application/mosskey-data et la seconde partie est multipart/mixed afin de porter les certificats à utiliser pour vérifier la signature.

Note de mise en œuvre : lorsque le chiffrement et la signature sont tous deux appliqués par le même agent, la signature devrait en général être appliquée avant le chiffrement.

2.6.3 En-têtes HTTP permis

2.6.3.1 Généralités

En général, les en-têtes HTTP [RFC2616] devraient apparaître dans le contenu interne (c'est-à-dire, le message/http) d'un message S-HTTP mais ne devraient pas apparaître dans l'enveloppe du message S-HTTP pour des raisons de sécurité. Cependant, certains en-têtes ont besoin d'être visibles aux agents qui n'ont pas accès aux données encapsulées. Ces en-têtes peuvent apparaître aussi dans les en-têtes S-HTTP.

Prière de noter que bien qu'une brève description de l'objet général de ces en-têtes soit fournie pour la clarté de l'exposé, la référence définitive est la [RFC2616].

2.6.3.2 Host

L'en-tête host spécifie l'hôte Internet et le numéro d'accès de la ressource demandée. Cet en-tête devrait être utilisé pour ôter toute ambiguïté entre les contextes de sécurité potentiellement multiples au sein desquels ce message pourrait être interprété. Noter que le message HTTP non enveloppé aura son propre champ Host (en supposant que c'est un message HTTP/1.1). Si ces champs ne correspondent pas, le serveur devrait répondre avec un code d'état 400.

2.6.3.3 Connection

Le champ Connection a précisément la même sémantique dans les en-têtes S-HTTP que dans les en-têtes HTTP. Cela permet que des connexions persistantes soient utilisées avec S-HTTP.

3. Paramètres cryptographiques

3.1 En-têtes d'options

Comme décrit au paragraphe 1.3.2, chaque demande S-HTTP est (au moins conceptuellement) pré conditionnée par les options de négociation fournies par le receveur potentiel. Les deux principales localisations pour ces options sont :

1. dans les en-têtes d'une demande/réponse HTTP,
2. dans le HTML qui contient l'ancre déréférencée.

Il y a deux sortes d'options cryptographiques qui peuvent être fournies : les options de négociation, exposées au paragraphe 3.2 portent les préférences cryptographiques du receveur potentiel du message ; les options de chiffrement, comme discutées au paragraphe 3.3 fournissent le matériel de chiffrement (ou des pointeurs sur ce matériel de chiffrement) qui peuvent être utiles à l'envoyeur lors de l'amélioration d'un message.

Lier des options cryptographiques à des ancres en utilisant les extensions HTML est le sujet du document d'accompagnement [RFC2659] et ne sera pas traité ici.

3.2 Options de négociation

3.2.1 Vue d'ensemble de la négociation

Les deux parties sont capables d'exprimer leurs exigences et préférences concernant les améliorations cryptographiques qu'elles vont permettre/exiger que fournisse l'autre partie. Le choix des options appropriées dépend des capacités d'améliorations et des exigences des applications.

Un en-tête de négociation est une séquence de spécifications dont chacune se conforme à un schéma en quatre parties qui précise :

Propriété – l'option qui est négociée, comme un algorithme de chiffrement en vrac.
Valeur – la valeur qui est discutée pour la propriété, comme DES-CBC
Direction – la direction qui va être affectée, à savoir, durant la réception ou la génération (du point de vue de l'origine).
Force – la force de la préférence, à savoir, exigée, facultative, refusée.

Par exemple, la ligne d'en-tête

SHTTP-Symmetric-Content-Algorithms: recv-optional=DES-CBC,RC2

pourrait être vue comme disant : "Vous êtes libres d'utiliser DES-CBC ou RC2 pour le chiffrement en vrac pour le chiffrement des messages qui me sont destinés."

On définit un nouvel en-tête (à utiliser dans l'en-tête HTTP encapsulé, pas dans l'en-tête S-HTTP) pour permettre la négociation de ces questions.

3.2.2 Format de l'option de négociation

Le format général pour les options de négociation est :
Option = Champ ":" Valeur-de-clé ";" *(Valeur-de-clé)
Valeur-de-clé = Clé "=" Valeur *("," Valeur)
Clé = Mode"-"Action ; Ceci est représenté comme un jeton sans espace blanche
Mode = "origine" | "receveur"
Action = "optionnel" | "exigé" | "refusé"

La valeur <Mode> indique si cette <Valeur-de-clé> se réfère à ce que les actions de l'agent concernent sur l'envoi de messages à confidentialité améliorée par opposition à leur réception. Pour toute paire mode-action donnée, l'interprétation à faire sur les améliorations (<Valeur>s) énumérées est :

'recv-optionnel' : l'agent va traiter l'amélioration si l'autre partie les utilise, mais va aussi traiter les messages sans amélioration.
'recv-exigé' : l'agent ne va pas traiter les messages sans cette amélioration.
'recv-refusé' : l'agent ne va pas traiter les messages avec cette amélioration.
'orig-optionnel' : lorsque il rencontre un agent qui refuse cette amélioration, l'agent ne la fournira pas, et quant il rencontre un agent qui l'exige, cet agent la fournira.
'orig-exigé' : l'agent va toujours générer l'amélioration.
'orig-refusé' : l'agent ne va jamais générer l'amélioration.

Le comportement des agents qui découvrent qu'ils communiquent avec un agent incompatible est à la discrétion des agents. Il n'est pas approprié de persister aveuglément dans un comportement dont on sait qu'il est inacceptable pour l'autre partie. Les réponses plausibles incluent de simplement terminer la connexion, ou, dans le cas d'une réponse de serveur, de retourner '501, Non mis en œuvre'.

Les valeurs optionnelles sont considérées être énumérées en ordre décroissant de préférence. Les agents sont cependant libres de choisir tout membre de l'intersection de la liste des options (ou aucune).

Si une <Valeur-de-clé> est laissée indéfinie, on devrait supposer qu'elle est réglée à la valeur par défaut. Toute clé qui est spécifiée par un agent devra supplanter toute apparition de cette clé dans toute <Valeur-de-clé> dans la valeur par défaut pour ce champ.

3.2.3 Paramètres pour les chiffrement à longueur de clé variable

Pour les chiffrements avec des longueurs de clé variables, les valeurs peuvent être paramétrées en utilisant la syntaxe <chiffrement>'['<longueur>']'

Par exemple, 'RSA[1024]' représente une clé de 1024 bits pour RSA. Les gammes peuvent être représentées comme

<chiffrement>'['<limite1>'-'<limite2>']'

Pour les besoins des préférences, cette notation devrait être traitée comme si elle disait (x et y étant des entiers)

<chiffrement>[x], <chiffrement>[x+1],...<chiffrement>[y] (si x<y)
et
<chiffrement>[x], <chiffrement>[x-1],...<chiffrement>[y] (si x>y)

La valeur particulière 'inf' peut être utilisée pour noter une longueur infinie.

Utiliser simplement <chiffrement> pour un tel chiffrement devra être lu comme la gamme maximum possible avec le chiffrement en question.

3.2.4 Syntaxe de négociation

3.2.4.1 SHTTP-Privacy-Domains

Cet en-tête se réfère au type Content-Privacy-Domain du paragraphe 2.3.1. Les valeurs acceptables sont énumérées ici. Par exemple,

SHTTP-Privacy-Domains: orig-required=cms;
recv-optional=cms,MOSS

indiquerait que l'agent génère toujours des messages conformes à la CMS, mais peut lire CMS ou MOSS (ou, des messages non améliorés).

3.2.4.2 SHTTP-Certificate-Types

Cet en-tête indique quelles sortes de certificats de clé publique va accepter l'agent. Les valeurs actuellement définies sont 'X.509' et 'X.509v3'.

3.2.4.3 SHTTP-Key-Exchange-Algorithms

Cet en-tête indique quels algorithmes peuvent être utilisés pour l'échange de clés. Les valeurs définies sont 'DH', 'RSA', 'Outband' et 'Inband'. DH se réfère à l'enveloppe de style Diffie-Hellman X9.42. RSA se réfère à l'enveloppe RSA. Outband se réfère à une sorte d'accord de clé externe. Inband se réfère au paragraphe 3.3.3.1.

La configuration commune attendue des clients qui n'ont pas de certificats et de serveurs ayant des certificats ressemblerait à ceci (dans un message envoyé par le serveur) :

SHTTP-Key-Exchange-Algorithms: orig-optional=Inband, DH;
recv-required=DH

3.2.4.4 SHTTP-Signature-Algorithms

Cet en-tête indique quels algorithmes de signature numérique peuvent être utilisés. Les valeurs définies sont 'RSA' [PKCS-1] et 'NIST-DSS' [FIPS-186]. Comme NIST-DSS et RSA utilisent des modules de longueur variable, la syntaxe de paramétrisation du paragraphe 3.2.3 devrait être utilisée. Noter qu'une spécification de longueur de clé peut interagir avec l'acceptabilité d'un certain certificat, car les clés (et leur longueur) sont spécifiées dans des certificats de clé publique.

3.2.4.5 SHTTP-Message-Digest-Algorithms

Cet en-tête indique quels algorithmes de résumé de message peuvent être utilisés. Les valeurs définies jusqu'ici sont 'RSA-MD2' [RFC1319], 'RSA-MD5' [RFC1321], 'NIST-SHS' [FIPS-180].

3.2.4.6 SHTTP-Symmetric-Content-Algorithms

Cet en-tête spécifie le chiffrement en vrac à clé symétrique utilisé pour chiffrer le contenu du message. Les valeurs définies sont :
DES-CBC -- DES en mode chaînage de bloc de chiffrement (CBC, *Cipher Block Chaining*) [FIPS-81]
DES-EDE-CBC -- 3DES à deux clés utilisant le chiffrement-déchiffrement-chiffrement en mode CBC externe
DES-EDE3-CBC -- 3DESà trois clés utilisant le chiffrement-déchiffrement-chiffrement en mode CBC externe
DESX-CBC -- DESX de RSA en mode CBC
IDEA-CBC -- IDEA en mode CBC
RC2-CBC -- RC2 de RSA en mode CBC
CDMF-CBC -- CDMF (DES à clé affaiblie) d'IBM [JOHN93] en mode CBC

Comme les clés RC2 sont de longueur variable, la syntaxe du paragraphe 3.2.3 devrait être utilisée.

3.2.4.7 SHTTP-Symmetric-Header-Algorithms

Cet en-tête spécifie le chiffrement à clé symétrique utilisé pour chiffrer les en-têtes de message.

DES-ECB -- DES en mode dictionnaire électronique (ECB, *Electronic Codebook*) [FIPS-81]
DES-EDE-ECB -- 3DES à deux clés utilisant le chiffrement-déchiffrement-chiffrement en mode ECB
DES-EDE3-ECB -- 3DES à trois clés utilisant le chiffrement-déchiffrement-chiffrement en mode ECB
DESX-ECB -- DESX de RSA en mode ECB
IDEA-ECB -- IDEA
RC2-ECB -- RC2 de RSA en mode ECB
CDMF-ECB -- CDMF d'IBM en mode ECB

Comme RC2 est de longueur variable, la syntaxe du paragraphe 3.2.3 devrait être utilisée.

3.2.4.8 SHTTP-MAC-Algorithms

Cet en-tête indique quels algorithmes sont acceptables pour fournir un MAC à clé symétrique. 'RSA-MD2', 'RSA-MD5' et 'NIST-SHS' subsistent de S-HTTP/1.1 en utilisant la vieille construction de MAC. Les jetons ' RSA-MD2-HMAC', 'RSA-MD5-HMAC' et 'NIST-SHS-HMAC' indiquent la nouvelle construction HMAC du paragraphe 2.3.6 avec respectivement les algorithmes MD2, MD5, et SHA-1.

3.2.4.9 SHTTP-Privacy-Enhancements

Cet en-tête indique les améliorations de sécurité à appliquer. Les valeurs possibles sont 'sign', 'encrypt' et 'auth' qui indiquent respectivement si les messages sont signés, chiffrés, ou authentifiés (c'est-à-dire, fournis avec un MAC).

3.2.4.10 Your-Key-Pattern

Cet en-tête a une syntaxe de correspondance de schéma généralisée pour décrire des identifiants pour un grand nombre de types de matériels de chiffrement. La syntaxe générale est :

```
Your-Key-Pattern = "Your-Key-Pattern" ":" key-use "," pattern-info
   key-use = "cover-key" | "auth-key" | "signing-key"
```

3.2.4.10.1 Schémas de clé de couverture

Cet en-tête spécifie les valeurs désirées pour les noms de clé utilisés pour le chiffrement des clés de transaction utilisant la syntaxe de Prearranged-Key-Info du paragraphe 2.3.5. La syntaxe pattern-info consiste en une série d'expressions régulières séparées par des virgules. Les virgules devraient être échappées avec des barres obliques inverses si elles apparaissent dans les regexps. On devrait supposer que le premier schéma est le préféré.

3.2.4.10.2 Schéma de clé d'authentification

Les schémas Auth-key spécifient les formes de nom désirées pour les authentifiants de MAC. La syntaxe pattern-info consiste en une série d'expressions régulières séparées par des virgules. Les virgules devraient être échappées avec des barres obliques inverses si elles apparaissent dans les regexps. On devrait supposer que le premier schéma est le préféré.

3.2.4.10.3 Schéma de clé de signature

Ce paramètre décrit un ou des schémas pour quelles clés sont acceptables pour signer l'amélioration de signature numérique. La syntaxe de pattern-info pour signing-key est :

```
pattern-info = name-domain "," pattern-data
```

Le seul nom de domaine actuellement défini est 'DN-1779'. Ce paramètre spécifie les valeurs désirées pour les champs de noms distinctifs (DN). Les DN sont considérés comme représentés selon la spécification de la RFC1779, l'ordre des champs et les espaces entre les champs sont non significatifs.

Toutes les valeurs de la RFC1779 devraient utiliser ',' comme séparateur plutôt que ';' car ';' est utilisé comme séparateur de déclarations dans S-HTTP.

Pattern-data est une chaîne modifiée de la RFC1779, avec des expressions régulières permises comme valeurs de champ. La correspondance de schéma est effectuée au niveau du champ, les champs inspécifiés correspondent à toute valeur (et donc, laisser le champ DN-Pattern entièrement inspécifié permet tout DN). Les chaînes de certificat peuvent aussi correspondre (pour permettre des certificats sans subordination de nom).Les chaînes de DN sont considérées comme ordonnées de gauche à droite avec le producteur d'un certificat donné immédiatement à sa droite, bien que les producteurs n'aient pas besoin d'être spécifiés. Un '.' en queue indique que la séquence des DN est absolue. C'est-à-dire que le plus à droite est une racine.

La syntaxe des valeurs de schéma est :
Value = DN-spec *("," Dn-spec)["."]
Dn-spec = "/" *(Field-spec) "/"
Field-spec := Attr = "Pattern"
Attr = "CN" | "L" | "ST" | "O" | "OU" | "C" | <ou comme approprié>
Pattern = <expressions régulières POSIX 1003.2>

Par exemple, pour demander que l'autre agent signe avec une clé certifiée par la CA RSA Persona (qui utilise la subordination de nom) on pourrait utiliser l'expression ci-dessous. Noter l'utilisation de la citation de la RFC1779 pour protéger la virgule (un séparateur de champ de la RFC1779) et la citation de POSIX 1003.2 pour protéger le point (une expression régulière de méta caractère).

```
   Your-Key-Pattern: signing-key, DN-1779,
            /OU=Persona Certificate, O="RSA Data Security,
 Inc\."/
```

3.2.4.11 Exemple

Voici un bloc d'en-tête représentatif pour un serveur :

SHTTP-Privacy-Domains: recv-optional=MOSS, CMS; orig-required=CMS
SHTTP-Certificate-Types: recv-optional=X.509; orig-required=X.509
SHTTP-Key-Exchange-Algorithms: recv-required=DH; orig-optional=Inband,DH
SHTTP-Signature-Algorithms: orig-required=NIST-DSS; recv-required=NIST-DSS
SHTTP-Privacy-Enhancements: orig-required=sign; orig-optional=encrypt

3.2.4.12 Valeurs par défaut

Les paramètres explicites de négociation ont priorité sur les valeurs par défaut. Pour un certain type d'option de négociation, les valeurs par défaut pour une certaine paire de mode-action (comme 'orig-required') sont implicitement fusionnées si elles ne sont pas explicitement outrepassées.

Les valeurs par défaut (qui peuvent être négociées vers l'aval ou vers l'amont) sont :
SHTTP-Privacy-Domains: orig-optional=CMS; recv-optional=CMS
SHTTP-Certificate-Types: orig-optional=X.509; recv-optional=X.509
SHTTP-Key-Exchange-Algorithms: orig-optional=DH,Inband,Outband; recv-optional=DH,Inband,Outband
SHTTP-Signature-Algorithms: orig-optional=NIST-DSS; recv-optional=NIST-DSS
SHTTP-Message-Digest-Algorithms: orig-optional=RSA-MD5; recv-optional=RSA-MD5
SHTTP-Symmetric-Content-Algorithms: orig-optional=DES-CBC; recv-optional=DES-CBC
SHTTP-Symmetric-Header-Algorithms: orig-optional=DES-ECB; recv-optional=DES-ECB
SHTTP-Privacy-Enhancements: orig-optional=sign,encrypt, auth; recv-required=encrypt; recv-optional=sign, auth

3.3 En-têtes de non négociation

Il y a un certain nombre d'options qui sont utilisées pour communiquer ou identifier le matériel de chiffrement du receveur potentiel.

3.3.1 Encryption-Identity

Cet en-tête identifie un principal potentiel pour lequel le message décrit par ces options pourrait être chiffré; Noter que ceci

permet explicitement un retour chiffré sous une clé publique (par exemple) sans que l'autre agent signe d'abord (ou sous une clé différente de celle de la signature). La syntaxe de la ligne Encryption-Identity est :

Encryption-Identity = "Encryption Identity" ":" name-class,key-sel,name-arg
name-class = "DN-1779" | formes de nom MOSS

name-class est une chaîne ASCII représentant le domaine au sein duquel le nom est à interpréter, dans l'esprit de MOSS. En plus des formes de nom MOSS de la RFC1848, on ajoute la forme de nom DN-1779 pour représenter une forme plus pratique de nom distinctif.

3.3.1.1 Classe de nom DN-1779

L'argument est un DN codé selon la RFC1779.

3.3.2 Certificate-Info

Afin de permettre les opérations de clé publique sur les DN spécifiés par les en-têtes Encryption-Identity sans que le receveur aille chercher un certificat explicite, l'envoyeur peut inclure des informations de certification dans l'option Certificate-Info. Le format de cette option est :

Certificate-Info: <Cert-Fmt>','<Cert-Group>

<Cert-Fmt> devrait être le type du <Cert-Group> présenté.

Les valeurs définies sont 'PEM' et 'CMS'. Les groupes de certificats CMS sont fournis comme un message CMS SignedData codé en base-64 contenant des séquences de certificats avec ou sans le champ SignerInfo. Un groupe de certificats de format PEM est une liste de certificats PEM codés en base64 et séparés par des virgules.

Plusieurs lignes Certificate-Info peuvent être définies.

3.3.3 Key-Assign

Cette option sert à indiquer que l'agent souhaite lier une clé à un nom symbolique pour (vraisemblablement) référence ultérieure.

La syntaxe générale de l'en-tête key-assign est :
Key-Assign = "Key-Assign" ":" Méthode "," Nom-de-clé "," Durée-de-vie "," Chiffrements ";" Arguments-de-méthode
Nom-de-clé = chaîne
Durée-de-vie = "ceci" | "réponse" | ""
Méthode ="inband"
Chiffrements = "null" | Chiffrement+
Chiffrement" = <Chiffrement d'en-tête du paragraphe 3.2.4.7>
kv = "4" | "5"

Nom-de-clé est le nom symbolique auquel cette clé doit être liée. Chiffrements est une liste de chiffrements pour lesquels cette clé est potentiellement applicable (voir la liste de chiffrements d'en-tête au paragraphe 3.2.4.7). Le mot clé 'null' devrait être utilisé pour indiquer qu'il est inapproprié pour l'utilisation avec TOUT chiffrement. Ceci peut être utile pour échanger des clés pour le calcul de MAC.

Durée-de-vie est une représentation de la plus longue période durant laquelle le receveur de ce message peut attendre que l'envoyeur accepte cette clé. 'ceci' indique qu'il n'est vraisemblablement valide que pour lire cette transmission. 'réponse' indique qu'il est utile pour une réponse à ce message. Si un Key-Assign avec la durée de vie de réponse apparaît dans un bloc CRYPTOPTS, il indique qu'il est bon pour au moins une (mais peut-être seulement une) déréférence de cette ancre. Une durée de vie inspécifiée implique que cette clé peut être réutilisée pour un nombre indéfini de transactions.

Méthode devrait être une des méthodes d'échange de clé. La seule valeur actuellement définie est 'inband' qui se réfère aux clés dans la bande (c'est-à-dire à l'allocation directe).

Cette ligne d'en-tête peut apparaître soit dans un en-tête non encapsulé, soit dans un message encapsule, bien que lorsque une clé non couverte est allouée directement, elle ne puisse apparaître que dans un contenu chiffré encapsulé. L'allouer à une clé qui existe déjà cause l'effacement de cette clé.

Les clés définies par cet en-tête sont appelées aussi dans la présente spécification des Key-ID, qui ont la syntaxe :

Key-ID = méthode ":" nom-de-clé

3.3.3.1 Allocation de clés dans la bande

Ceci se réfère à l'allocation directe d'une clé non couverte à un nom symbolique. Method-args devrait juste être la clé de session désirée codée en hexadécimal comme dans :

Key-Assign: inband,akey,reply,DES-ECB;0123456789abcdef

Les clés courtes devraient être déduites des clés longues en lisant les bits de gauche à droite.

Noter que l'allocation de clés dans la bande est particulièrement importante afin de permettre une communication confidentielle spontanée entre les agents lorsque un (mais pas les deux) des agents a des paires de clés. Cependant, ce mécanisme est aussi utile pour permettre des changements de clé sans calcul de clé publique. Les informations de clé qui sont portées dans cette ligne d'en-tête doivent être dans la demande HTTP sécurisée interne. et donc leur utilisation dans des messages non chiffrés n'est pas permise.

3.3.4 Noms occasionnels

Les noms occasionnels *(nonce)* sont des identifiants opaques, transitoires, par session, qui peuvent être utilisés pour démontrer la fraîcheur. Les. valeurs de nom occasionnel sont une affaire locale, mais elles peuvent aussi bien être simplement des nombres aléatoires générés par l'origine. La valeur est fournie simplement pour être retournée par le receveur.

3.3.4.1 Nonce

Cet en-tête est utilisé par un générateur pour spécifier quelle valeur sera retournée dans la réponse. Le champ peut avoir n'importe quelle valeur. Plusieurs noms occasionnels peuvent être fournis, chacun étant retourné indépendamment.

Le nom occasionnel devrait être retourné dans une ligne d'en-tête Nonce-Echo. Voir au paragraphe 4.1.1.

3.4 Groupement d'en-têtes avec SHTTP-Cryptopts

Pour que les serveurs lient un groupe d'en-têtes à une ancre HTML, il est possible de combiner un certain nombre d'en-têtes sur une seule ligne d'en-tête Cryptopts S-HTTP. Le nom de l'ancre à laquelle ces en-têtes s'appliquent est indiqué par un paramètre 'scope'.

3.4.1 SHTTP-Cryptopts

Cette option fournit un ensemble de cryptopts et une liste de références à celui auquel il s'applique. (Pour HTML, ces références seront désignées en utilisant l'étiquette NAME). Les noms sont fournis dans l'attribut scope *(portée)* comme une liste séparée par des virgules et séparée de la ligne d'en-tête suivante par un point-virgule. Le format de la ligne SHTTP-Cryptopts est :

SHTTP-Cryptopts = "SHTTP-Cryptopts" ":" scope ";" cryptopt-list
scope = "scope="<tag-spec> ; tout ceci est un seul jeton sans espace
tag-spec = tag *("," tag) | ""
cryptopt-list = cryptopt *(";" cryptopt)
cryptopt = <lignes cryptopt S-HTTP décrites ci-dessous>
tag = <valeur utilisée dans l'attribut d'ancre HTML NAME>

Par exemple :

SHTTP-Cryptopts: scope=tag1,tag2;
SHTTP-Privacy-Domains:
orig-required=cms; recv-optional=cms,MOSS

Si un message contient à la fois des en-têtes de négociation S-HTTP et des en-têtes groupés sur un ou des lignes SHTTP-Cryptopts, les autres en-têtes devront être pris comme s'appliquant à toutes les ancres non liées sur la ou les lignes SHTTP-Cryptopts. Noter que cette proposition est du type tout ou rien. C'est-à-dire que si un en-tête SHTTP-Cryptopts lie des options

à une référence, aucune de ces options globales ne s'applique, même si certains des en-têtes d'option n'apparaissent pas dans les options liées. C'est plutôt les valeurs S-HTTP par défaut qu'on trouve au paragraphe 3.2.4.11 qui s'appliquent.

4. Nouvelles lignes d'en-tête pour HTTP

On définit ici deux lignes d'en-tête de non négociation pour HTTP.

4.1 Security-Scheme

Tous les agents conformes à S-HTTP doivent générer l'en-tête Security-Scheme dans les en-têtes de tous les messages HTTP qu'ils génèrent. Cet en-tête permet aux autres agents de détecter qu'ils sont en train de communiquer avec un agent conforme à S-HTTP et de générer les en-têtes d'options cryptographiques appropriés.

Pour les mises en œuvre conformes à la présente spécification, la valeur doit être 'S-HTTP/1.4'.

4.1.1 Nonce-Echo

Cet en-tête est utilisé pour retourner la valeur fournie dans un champ Nonce: reçu précédemment. Il doit aller dans les en-têtes encapsulés afin qu'il puisse être cryptographiquement protégé.

5. Réessai sur rapport d'état d'erreur du serveur

On décrit ici le traitement spécial qui est approprié pour que le client réessaye en présence du retour par un serveur d'un état d'erreur.

5.1 Réessai pour renégociation d'option

Un serveur peut répondre à une demande d'un client par un code d'erreur qui indique que la demande n'a pas complètement échoué et que le client pourrait avoir satisfaction au moyen d'une autre demande. HTTP a déjà ce concept avec les codes de redirection 3XX.

Dans le cas de S-HTTP, il est concevable (et en fait probable) que le serveur s'attende à ce que le client réessaye sa demande en utilisant un autre ensemble d'options cryptographiques. Par exemple, le document qui contient l'ancre que le client déréférence est vieux et n'exige pas de signature numérique pour la demande en question, mais le serveur a maintenant une politique qui exige une signature pour déréférencer cet URL. Ces options devraient être portées dans l'en-tête du message HTTP encapsulé, précisément comme sont portées les options du client.

L'idée générale est que le client va effectuer le réessai de la manière indiquée par la combinaison de la demande d'origine et de la nature précise de l'erreur et des améliorations cryptographiques selon les options portées dans la réponse du serveur.

Le principe directeur dans la réponse du client à ces erreurs devrait être de fournir à l'utilisateur la même sorte d'informations de choix par rapport au déréférencement de ces ancres qu'au déréférencement normal d'ancre. Par exemple, dans le cas ci-dessus, il serait inapproprié que le client signe la demande sans demander la permission pour l'action.

5.2 Comportement spécifique de réessai

5.2.1 401 Non autorisé, 402 Payement exigé

Les erreurs HTTP '401 Non autorisé', '402 Payement exigé' représentent des échecs de schémas d'authentification et de payement de style HTTP. Bien que S-HTTP ne prenne pas explicitement en charge ces mécanismes, ils peuvent être effectués sous S-HTTP tout en tirant parti des services de confidentialité offerts par S-HTTP. (Il y a d'autres erreurs pour les erreurs d'authentification spécifiques de S-HTTP.)

5.2.2 420 Réessai de sécurité

Cette réponse d'état du serveur est fournie afin que le serveur puisse informer le client que bien que la demande actuelle soit

rejetée, un nouvel essai demandé avec des améliorations cryptographiques différentes vaut la peine d'être tenté. Cet en-tête devra aussi être utilisé dans le cas où une demande HTTP a été faite mais qu'une demande S-HTTP devrait avoir été faite. Évidemment, cela ne sert à rien d'autre qu'à signaler une erreur si la demande d'origine aurait dû être chiffrée, mais dans d'autres situations (par exemple, le contrôle d'accès) cela peut être utile.

5.2.2.1 Réessais de sécurité pour les demandes S-HTTP

Dans le cas d'une demande qui a été faite comme demande S-HTTP, cela indique que pour certaines raisons, les améliorations cryptographiques appliquées à la demande ont été non satisfaisantes et que la demande devrait être répétée avec les options trouvées dans l'en-tête de réponse. Noter que ceci peut être utilisé comme moyen de forcer une nouvelle négociation de clé publique si la clé de session utilisée est arrivée à expiration ou pour fournir un nom occasionnel unique afin d'assurer la fraîcheur de la demande.

5.2.2.2 Réessais de sécurité pour les demandes HTTP

Si le code 420 est retourné dans une réponse à une demande HTTP, il indique que la demande devrait être réessayée en utilisant S-HTTP et les options cryptographiques indiquées dans l'en-tête de réponse.

5.2.3 421 En-tête erroné

Ce code d'erreur indique que quelque chose ne va pas dans la demande S-HTTP. Le code d'erreur doit être suivi d'une explication appropriée, par exemple :

421 En-tête erroné, le domaine de confidentialité du contenu doit être spécifié

5.2.4 422 Authentification du mandataire SHTTP exigée

Cette réponse est analogue à la réponse 420 excepté que les options dans le message se réfèrent aux améliorations que le client doit effectuer afin de satisfaire le mandataire.

5.2.5 320 SHTTP Non modifié

Ce code de réponse est à utiliser spécifiquement avec une interaction mandataire-serveur où le mandataire a placé l'en-tête If-Modified-Since dans les en-têtes S-HTTP de sa demande. Cette réponse indique que le message S-HTTP suivant contient du matériel de chiffrement suffisant pour que le mandataire transmette le document en antémémoire pour le nouveau demandeur.

En général, cela prend la forme d'un message S-HTTP où le contenu réel amélioré manque, mais tous les en-têtes et le matériel de chiffrement sont conservés. (C'est-à-dire que la section de contenu facultatif du message de CMS a été retiré.) Ainsi, si la réponse d'origine était chiffrée, la réponse contient la DEK originale recouverte pour le nouveau receveur. (Remarquer que le serveur effectue le même traitement que si il avait été dans le cas du côté serveur qui met en antémémoire du 7.1 excepté que le corps du message est élidé.)

5.2.6 3XX Redirection

Ces en-têtes sont, là aussi, internes à HTTP, mais peuvent contenir des options de négociation S-HTTP significatives pour S-HTTP. La demande devrait être redirigée dans le sens de HTTP, avec les précautions cryptographiques appropriées.

5.3 Limitations sur les réessais automatiques

Permettre des réessais automatiques du client en réponse à cette sorte de réponse de serveur autorise plusieurs formes d'attaques. Considérons pour le moment le simple cas de la carte de crédit:

L'utilisateur regarde un document qui exige sa carte de crédit. Il vérifie que le DN du receveur prévu est acceptable et que la demande sera chiffrée et il déréférence l'ancre. L'attaquant intercepte la réponse du serveur et répond par un message chiffré sous la clé publique du client contenant l'en-tête 301 Déplacé. Si le client devait effectuer automatiquement cette redirection, cela permettrait de compromettre la carte de crédit de l'utilisateur.

5.3.1 Réessai automatique de chiffrement

Cela montre un des dangers possibles des réessais automatiques – la compromission potentielle des informations chiffrées. Alors qu'il est impossible de considérer tous les cas possibles, les clients ne devraient jamais rechiffrer automatiquement des données sauf si le serveur demande le réessai ce qui prouve qu'il a déjà les données. Ainsi, les situations dans lesquelles il serait acceptable de rechiffrer seraient :

1. si la réponse de réessai a été retournée chiffrée sous une clé dans la bande nouvellement générée pour la demande d'origine,
2. si la réponse de réessai a été signée par le receveur prévu de la demande d'origine,
3. si la demande originale a utilisé une clé hors bande et si la réponse est chiffrée sous cette clé.

Cette liste n'est pas exhaustive, cependant les auteurs de navigateurs seraient bien inspirés de bien réfléchir avant de mettre en œuvre le rechiffrement automatique dans d'autres cas. Noter qu'un comportement approprié dans les cas où le rechiffrement automatique n'est pas approprié est de demander la permission à l'utilisateur.

5.3.2 Réessai automatique de signature

Comme on déconseille la signature automatique (sans confirmation de l'utilisateur) même dans les cas courants, et étant donnés les dangers décrits ci-dessus, il est interdit de réessayer automatiquement l'amélioration de signature.

5.3.3 Réessai automatique de l'authentification de MAC

En supposant que toutes les autres conditions sont suivies, il est permis de réessayer automatiquement l'authentification de MAC.

6. Autres questions

6.1 Compatibilité des serveurs avec les vieux clients

Les serveurs qui reçoivent des demandes en clair qui pourraient être sécurisées devraient retourner '420 Réessai de sécurité' avec les lignes d'en-tête réglées à indiquer les améliorations de confidentialité requises.

6.2 Type de protocole d'URL

On définit un nouveau désignateur de protocole d'URL, 'shttp'. L'utilisation de ce désignateur au titre d'un URL d'ancre implique que le serveur cible a la capacité S-HTTP, et qu'une déréférence de cet URL devrait entreprendre le traitement S-HTTP.

Noter que les agents sans S-HTTP ne devraient pas vouloir déréférencer un URL qui a une spécification de protocole qu'ils ne connaissent pas, et donc des données sensibles ne seront pas envoyées accidentellement en clair par les utilisateurs de clients non sécurisée.

6.3 Présentation du navigateur

6.3.1 État de sécurité de transaction

Lors de la préparation d'un message sûr, le navigateur devrait fournir une indication visuelle de la sécurité de la transaction, ainsi qu'une indication de la partie qui sera capable de lire le message. Lors de la lecture d'un message signé et/ou enveloppé, le navigateur devrait indiquer cela et (si c'est applicable) l'identité du signataire. Les certificats auto signés devraient être clairement différenciés de ceux validés par une hiérarchie de certification.

6.3.2 Rapport des défaillances

L'échec de l'authentification ou du déchiffrement d'un message S-HTTP devrait être présenté différemment de l'échec de la restitution du document. Les clients conformes peuvent à leur gré afficher des documents invérifiables mais doivent clairement indiquer qu'ils étaient invérifiables d'une façon clairement distincte de la manière dont ils affichent les documents qui ne possèdent pas de signature numérique ou des documents qui ont une signature vérifiable.

6.3.3 Gestion des certificats

Les clients devront fournir une méthode pour déterminer que les demandes HTTP sont à signer et pour déterminer quel certificat (en supposant qu'ils sont nombreux) est à utiliser pour la signature. Il est suggéré qu'il soit présenté aux utilisateurs une sorte de liste de sélection à partir de laquelle ils peuvent choisir une valeur par défaut. Aucune signature ne devrait être effectuée sans une sorte d'action explicite de l'interface d'utilisateur, bien qu'une telle action puisse prendre la forme d'un réglage persistant via un mécanisme de préférences de l'utilisateur (mais ceci est déconseillé.)

6.3.4 Déréférencement d'ancre

Les clients devront fournir une méthode pour afficher le DN et la chaîne de certificats associée à une certaine ancre à déréférencer afin que les utilisateurs puissent déterminer pour qui leurs données sont chiffrées. Ceci devrait être distinct de la méthode pour afficher qui a signé le document contenant l'ancre car ce sont des éléments orthogonaux des informations de chiffrement.

7. Notes de mise en œuvre

7.1 Données pré améliorées

Alors que S-HTTP a toujours pris en charge les documents pré améliorés, dans les versions précédentes, comment les mettre réellement en œuvre n'a jamais été clair. Ce paragraphe décrit deux méthodes pour le faire : pré améliorer les demandes/réponses HTTP et pré améliorer les données sous-jacentes.

7.1.1 Motivation

Les deux motivations principales des documents pré améliorés sont la sécurité et les performances. Ces avantages proviennent principalement de la signature mais peuvent aussi dans des circonstances particulières s'appliquer à la confidentialité ou l'authentification répudiable (fondée sur le MAC).

Considérons le cas d'un serveur qui dessert de façon répétée le même contenu à plusieurs clients. Un tel exemple serait celui d'un serveur qui dessert des catalogues ou des listes de prix. Il est clair que les consommateurs aimeraient être capables de vérifier que ces prix sont réels. Cependant, comme les prix sont normalement les mêmes pour tous les arrivants, la confidentialité n'est pas un problème. (Voir au paragraphe 7.1.5 comment traiter aussi ce cas).

Par conséquent, le serveur peut souhaiter signer le document une seule fois et simplement envoyer le document signé qui est en antémémoire lorsque un client fait une nouvelle demande, évitant la surcharge d'une opération de clé privée à chaque fois. Noter qu'on peut concevoir que le document signé puisse avoir été généré par un tiers et placé dans l'antémémoire du serveur. Le serveur peut même ne pas avoir la clé de signature ! Cela illustre l'avantage de sécurité apporté par la pré signature: Les serveurs qui ne sont pas de confiance peuvent desservir les données authentifiées sans risque même si le serveur est compromis.

7.1.2 Demandes/réponses pré signées

La mise en œuvre évidente est de simplement prendre une seule demande/réponse, de la mettre en antémémoire, et de l'envoyer dans les situations où un nouveau message aurait autrement été généré.

7.1.3 Documents pré signés

Il est aussi possible en utilisant S-HTTP de signer les données sous-jacentes et de les envoyer comme message S-HTTP. Pour ce faire, on va prendre le document signé (un message CMS ou MOSS) et y attacher les deux en-têtes S-HTTP (par exemple,. la ligne demande/réponse S-HTTP et le Content-Privacy-Domain) et les en-têtes HTTP nécessaires (incluant un Content-Type qui reflète le contenu interne).

```
SECURE * Secure-HTTP/1.4
Content-Type: text/html
Content-Privacy-Domain: CMS

Un message signé aléatoire vient ici ...
```

Ce message ne peut pas être lui-même envoyé, mais a besoin d'une encapsulation récurrente, comme décrit au paragraphe

suivant.

7.1.4 Encapsulation récurrente

Comme requis au paragraphe 7.3, le résultat ci-dessus a lui-même besoin d'être encapsulé pour protéger les en-têtes HTTP. Le cas évident (et celui illustré ici) est quand la confidentialité est exigée, mais que l'amélioration d'authentification ou même la transformation nulle pourraient être appliquées à la place. C'est-à-dire, le message qu'on montre ci-dessus peut être utilisé comme contenu interne d'un nouveau message S-HTTP, comme celui-ci :

```
SECURE * Secure-HTTP/1.4
Content-Type: application/s-http
Content-Privacy-Domain: CMS

Version chiffrée du message ci-dessus...
```

Pour déplier ceci, le receveur va décoder le message S-HTTP externe, rentrer la boucle d'analyse (S-)HTTP pour traiter le nouveau message, voir que c'est aussi du S-HTTP, le décoder, et retrouver le contenu interne.

Noter que cette approche peut aussi être utilisée pour assurer la fraîcheur de l'activité du serveur (mais pas celle du document lui-même) tout en fournissant la non répudiation des données du document si un NONCE est inclus dans la demande.

7.1.5 Messages pré chiffrés

Bien que la pré amélioration fonctionne mieux avec une signature, elle peut aussi être utilisée avec un chiffrement sous certaines conditions. Considérons la situation où le même document confidentiel est à envoyer de façon répétée. Le temps passé à chiffrer peut être économisé en mettant en antémémoire le texte chiffré et en générant simplement un nouveau bloc d'échange de clé pour chaque receveur. (Noter que ceci est logiquement équivalent à un message multi receveurs comme défini à la fois dans MOSS et CMS et il faut donc veiller à utiliser le bourrage PKCS-1 approprié si RSA est utilisé car autrement, on peut ouvrir la voie à une attaque de faible exposant de chiffrement [HAST96].

7.2 Interaction de mandataire

L'utilisation de S-HTTP présente des problèmes de mise en œuvre pour l'utilisation de mandataires HTTP. Bien qu'avoir simplement un mandataire qui transmet aveuglément les réponses soit très direct, il serait préférable que les mandataires à capacité S-HTTP soient quand même capables de mettre les réponses en antémémoire au moins dans certaines circonstances. De plus, les services S-HTTP devraient être utilisables pour protéger l'authentification de client-mandataire. Le présent paragraphe décrit comment réaliser ces objectifs en utilisant les mécanismes décrits ci-dessus.

7.2.1 Authentification client-mandataire

Lorsque un mandataire à capacité S-HTTP reçoit une demande (HTTP ou S-HTTP) qui (quelles que soient les règles de contrôle d'accès qu'il utilise) exige d'être authentifié par S-HTTP (et si il ne l'est pas déjà) il devrait retourner le code de réponse 422 (5.7.4).

Lorsque le client reçoit le code de réponse 422, il devrait lire les options cryptographiques que le mandataire a envoyées et déterminer si il veut ou non appliquer cette amélioration au message. Si le client veut satisfaire ces exigences, il devrait encapsuler de façon récurrente la demande qu'il a envoyé précédemment en utilisant les options appropriées. (Noter que comme l'amélioration est appliquée de façon récurrente, même les clients qui ne veulent pas envoyer des demandes en clair aux serveurs peuvent vouloir envoyer le message déjà chiffré au mandataire sans autre chiffrement.) (Voir au paragraphe 7.1 un autre exemple de message encapsulé de façon récurrente.)

Lorsque le mandataire reçoit un tel message, il devrait supprimer l'encapsulation interne pour récupérer le message qui devrait être envoyé au serveur.

8. Recommandations et exigences de mise en œuvre

Tous les agents S-HTTP doivent prendre en charge le résumé de message MD5 et l'authentification par MAC. Selon S-HTTP/1.4, tous les agents doivent aussi prendre en charge la construction RSA-MD5-HMAC.

Tous les agents S-HTTP doivent prendre en charge l'échange de clés Outband, Inband, et DH.

Tous les agents doivent prendre en charge le chiffrement utilisant DES-CBC.

Les agents doivent prendre en charge la génération et la vérification de signature utilisant NIST-DSS.

9. Résumé de la syntaxe du protocole

On présente ci-dessous un résumé des principales caractéristiques syntaxiques de S-HTTP/1.4, à l'exclusion de l'encapsulation du message propre.

9.1 En-têtes S-HTTP (non encapsulés)

Content-Privacy-Domain: ('CMS' | 'MOSS')
Prearranged-Key-Info: <Hdr-Cipher>,<Key>,<Key-ID>
Content-Type: 'message/http'
MAC-Info: [hex(timeofday)',']<hash-alg>','hex(<hash-data>)',' <key-spec>

9.2 Options de non négociation HTTP (encapsulées)

Key-Assign: <Method>','<Key-Name>','<Lifetime>',' <Ciphers>';'<Method-args>
Encryption-Identity: <name-class>','<key-sel>','<name-args>
Certificate-Info: <Cert-Fmt>','<Cert-Group>
Nonce: <string>
Nonce-Echo: <string>

9.3 Options de négociation encapsulées

SHTTP-Cryptopts: <scope>';'<string>(,<string>)*
SHTTP-Privacy-Domains: ('CMS' | 'MOSS')
SHTTP-Certificate-Types: ('X.509')
SHTTP-Key-Exchange-Algorithms: ('DH', 'RSA' | 'Inband' | 'Outband')
SHTTP-Signature-Algorithms: ('RSA' | 'NIST-DSS')
SHTTP-Message-Digest-Algorithms: ('RSA-MD2' | 'RSA-MD5' | 'NIST-SHS' 'RSA-MD2-HMAC', 'RSA-MD5-HMAC', 'NIST-SHS-HMAC')
SHTTP-Symmetric-Content-Algorithms: ('DES-CBC' | 'DES-EDE-CBC' | 'DES-EDE3-CBC' | 'DESX-CBC' | 'CDMF-CBC' | 'IDEA-CBC' | 'RC2-CBC')
SHTTP-Symmetric-Header-Algorithms: ('DES-ECB' | 'DES-EDE-ECB' | 'DES-EDE3-EBC' | 'DESX-ECB' | 'CDMF-ECB' | 'IDEA-ECB' | 'RC2-ECB')
SHTTP-Privacy-Enhancements: ('sign' | 'encrypt' | 'auth')
Your-Key-Pattern: <key-use>','<pattern-info>

9.4 Méthodes HTTP

Secure * Secure-HTTP/1.4

9.5 Rapports d'état du serveur

Secure-HTTP/1.4 200 OK
SecurityRetry 420
BogusHeader 421 <reason>

10. Exemple étendu

On donne ici un exemple forcé d'une série de demandes et réponses S-HTTP. Des rangée de signe égal sont utilisées pour séparer la partie narrative des élément de l'exemple de message. Noter que les corps réels de message chiffré ou signé seraient normalement une suite incompréhensible de caractères binaires. Pour tenter de préserver la lisibilité tout en utilisant des messages (principalement) authentiques, les corps des demandes ont été codés en base64. Pour régénérer les messages S-HTTP réels, il est nécessaire de retirer le codage base64 du corps du message.

10.1 Demande utilisant un échange de clé RSA avec réponse de clé dans la bande

Alice, qui utilise un client à capacité S-HTTP, commence par faire une demande HTTP qui donne la page de réponse suivante :

```
============================================================
200 OK HTTP/1.0
Server-Name: Navaho-0.1.3.3alpha
Certificate-Info: CMS,MIAGCSqGSIb3DQEHAqCAMIACAQExADCABgkqh
    kiG9w0BBwEAAKCAM
    IIBrTCCAUkCAgC2MA0GCSqGSIb3DQEBAgUAME0xCzAJBgNVBAYTAlVTMSAwH
    gYDVQQKExdSU0EgRGF0YSBTZWN1cml0eSwgSW5jLjEcMBoGA1UECxMTUGVyc
    29uYSBDZXJ0aWZpY2F0ZTAeFw05NDA0MDkwMDUwMzdaFw05NDA4MDIxODM4N
    TdaMGcxCzAJBgNVBAYTAlVTMSAwHgYDVQQKExdSU0EgRGF0YSBTZWN1cml0e
    SwgSW5jLjEcMBoGA1UECxMTUGVyc29uYSBDZXJ0aWZpY2F0ZTEYMBYGA1UEA
    xMPU2V0ZWMgQXN0cm9ub215MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAMy8Q
    cW7RMrB4sTdQ8Nmb2DFmJmkWn+el+NdeamIDElX/qw9mIQu4xNj1FfepfJNx
    zPvA0OtMKhy6+bkrlyMEU8CAwEAATANBgkqhkiG9w0BAQIFAANPAAYn7jDgi
    rhiIL4wnP8nGzUisGSpsFsF4/7z2P2wqne6Qk8Cg/Dstu3RyaN78vAMGP8d8
    2H5+Ndfhi2mRp4YHiGHz0HlK6VbPfnyvS2wdjCCAccwggFRAgUCQAAAFDANB
    gkqhkiG9w0BAQIFADBfMQswCQYDVQQGEwJVUzEgMB4GA1UEChMXUlNBIERhd
    GEgU2VjdXJpdHksIEluYy4xLjAsBgNVBAsTJUxvdyBBc3N1cmFuY2UgQ2Vyd
    GlmaWNhdGlvbiBBdXRob3JpdHkwHhcNOTQwMTA3MDAwMDAwWhcNOTYwMTA3M
    jM1OTU5WjBNMQswCQYDVQQGEwJVUzEgMB4GA1UEChMXUlNBIERhdGEgU2Vjd
    XJpdHksIEluYy4xHDAaBgNVBAsTE1BlcnNvbmEgQ2VydGlmaWNhdGUwaTANB
    gkqhkiG9w0BAQEFAANYADBVAk4GqghQDa9Xi/2zAdYEqJVIcYhlLN1FpI9tX
    Q1m6zZ39PYXK8Uhoj0Es7kWRv8hC04vqkOKwndWbzVtvoHQOmP8nOkkuBi+A
    QvgFoRcgOUCAwEAATANBgkqhkiG9w0BAQIFAANhAD/5Uo7xDdp49oZm9GoNc
    PhZcW1e+nojLvHXWAU/CBkwfcR+FSf4hQ5eFu1AjYv6Wqf430Xe9Et5+jgnM
    Tiq4LnwgTdA8xQX4elJz9QzQobkE3XVOjVAtCFcmiin80RB8AAAMYAAAAAAA
    AAAAA==

Encryption-Identity: DN-1779, null, CN=Setec Astronomy, OU=Persona
    Certificate,O="RSA Data Security, Inc.", C=US;
SHTTP-Privacy-Enhancements: recv-required=encrypt

<A name=tag1 HREF="shttp://www.setec.com/secret">
Ne pas lire ceci. </A>
============================================================
```

Une demande HTTP appropriée pour déréférencer cet URL serait :

```
============================================================
GET /secret HTTP/1.0
Security-Scheme: S-HTTP/1.4
User-Agent: Web-O-Vision 1.2beta
Accept: *.*
Key-Assign: Inband,1,reply,des-ecb;7878787878787878

============================================================
```

La ligne ajoutée Key-Assign qui ne serait pas dans une demande HTTP ordinaire permet à Bob (le serveur) de chiffrer sa réponse à Alice, bien qu'Alice n'ait pas de clé publique, car ils partageraient une clé après que la demande est reçue par Bob. Cette demande a l'encapsulation S-HTTP suivante :

```
================================================================
Secure * Secure-HTTP/1.4
Content-Type: message/http
Content-Privacy-Domain: CMS

MIAGCSqGSIb3DQEHA6CAMIACAQAxgDCBqQIBADBTME0xCzAJBgNVBAYTAlVTMSAw
HgYDVQQKExdSU0EgRGF0YSBTZWN1cml0eSwgSW5jLjEcMBoGA1UECxMTUGVyc29u
YSBDZXJ0aWZpY2F0ZQICALYwDQYJKoZIhvcNAQEBBQAEQCU/R+YCJSUsV6XLilHG
cNVzwqKcWzmT/rZ+duOv8Ggb7oO/d8H3xUVGQ2LsX4kYGq2szwj8Q6eWhsmhf4oz
lvMAADCABgkqhkiG9w0BBwEwEQYFKw4DAgcECFif7BadXlw3oIAEgZBNcMexKe16
+mNxx8YQPukBCL0bWqS86lvws/AgRkKPELmysBi5lco8MBCsWK/fCyrnxIRHs1oK
BXBVlsAhKkkusk1kCf/GbXSAphdSgG+d6LxrNZwHbBFOX6A2hYS63Iczd5bOVDDW
Op2gcgUtMJq6k2LFrs4L7HHqRPPlqNJ6j5mFP4xkzOCNIQynpD1rV6EECMIk/T7k
1JLSAAAAAAAAAAAAAA==
================================================================
```

Les données entre les délimiteurs sont un message CMS, enveloppé en RSA pour Setec Astronomy.

Bob déchiffre la demande, trouve le document en question, et est prêt à le reservir à Alice.

Une réponse appropriée du serveur HTTP serait :

```
================================================================
HTTP/1.0 200 OK
Security-Scheme: S-HTTP/1.4
Content-Type: text/html

Félicitations, vous avez gagné.
<A href="/prize.html"
CRYPTOPTS="Key-Assign: Inband,alice1,reply,des-ecb;020406080a0c0e0f;
SHTTP-Privacy-Enhancements: recv-required=auth">Cliquer ici pour demander votre prix</A>
================================================================
```

Cette réponse HTTP, encapsulée comme un message S-HTTP devient :

```
================================================================
Secure * Secure-HTTP/1.4
Content-Type: message/http
Prearranged-Key-Info: des-ecb,697fa820df8a6e53,inband:1
Content-Privacy-Domain: CMS

MIAGCSqGSIb3DQEHBqCAMIACAQAwgAYJKoZIhvcNAQcBMBEGBSsOAwIHBAifqtdy
x6uIMYCCARgvFzJtOZBn773DtmXlx037ck3giqnV0WC0QAx5f+fesAiGaxMqWcir
r9XvT0nT0LgSQ/8tiLCDBEKdyCNgdcJAduy3D0r2sb5sNTT0TyL9uydG3w55vTnW
aPbCPCWLudArI1UHDZbnoJICrVehxG/sYX069M8v6VO8PsJS7//hh1yM+0nekzQ5
l1p0j7uWKu4W0csrlGqhLvEJanj6dQAGSTNCOoH3jzEXGQXntgesk8poFPfHdtj0
5RH4MuJRajDmoEjlrNcnGl/BdHAd2JaCo6uZWGcnGAgVJ/TVfSVSwN5nlCK87tXl
nL7DJwaPRYwxb3mnPKNq7ATiJPf5u162MbwxrddmiE7e3sST7naSN+GS0ateY5X7
AAAAAAAAAAA=
================================================================
```

Les données entre les délimiteurs sont un message CMS chiffré sous une DEK choisie au hasard qui peut être retrouvée en calculant :

DES-DECRYPT(inband:1,697fa820df8a6e53)

où 'inband:1' est la clé échangée dans la ligne Key-Assign dans la demande d'origine.

10.2 Demande utilisant l'amélioration auth

Il y a un lien sur la page HTML qui vient d'être retournée, qu'Alice déréference, créant le message HTTP :

```
================================================================
GET /prize.html HTTP/1.0
Security-Scheme: S-HTTP/1.4
User-Agent: Web-O-Vision 1.1beta
Accept: *.*

================================================================
```

Qui, lorsque encapsulé comme un message S-HTTP, devient :

```
================================================================
Secure * Secure-HTTP/1.4
Content-Type: message/http
MAC-Info:31ff8122,rsa-md5,b3ca4575b841b5fc7553e69b0896c416,inband:alice1
Content-Privacy-Domain: CMS

MIAGCSqGSIb3DQEHAaCABGNHRVQgL3ByaXplLmh0bWwgSFRUUC8xLjAKU2VjdXJp
dHktU2NoZW1lOiBTLUhUVFAvMS4xClVzZXItQWdlbnQ6IFdlYi1PLVZpc2lvbiAx
LjFiZXRhCkFjY2VwdDogKi4qCgoAAAAA
================================================================
```

Les données entre les délimiteurs sont une représentation CMS 'Data' de la demande.

Appendice A Rappel de CMS

CMS (norme de "syntaxe de message cryptographique") est un format d'encapsulation de message cryptographique, similaire à PEM, fondé sur la syntaxe de message cryptographique PKCS-7 de RSA.

CMS est seulement un des deux formats d'encapsulation pris en charge par S-HTTP, mais il doit être préféré car il permet l'ensemble le moins restreint d'options négociables, et permet le codage binaire. Pour rendre la présente spécification plus autonome, on résumé ici la CMS.

La CMS est définie en termes de notation de syntaxe abstraite n° 1 de l'OSI (ASN.1, définie dans X.208) et est concrètement représentée en utilisant les règles de codage de base (BER, *Basic Encoding Rules*) de l'ASN.1 définies dans X.209. Un message CMS est une séquence de parties de contenu typées. Il y a six types de contenu, composables de façon récurrente :

Data – quelques octets, sans amélioration.

SignedData – une partie de contenu, avec zéro, un ou plusieurs blocs de signature, et le matériel de chiffrement associé. Les matériaux de chiffrement peuvent être transportés via le cas dégradé de pas de bloc de signature et pas de données.

EnvelopedData -- un ou plusieurs (par receveur) blocs d'échange de clés et une partie de contenu chiffré.

DigestedData – une partie de contenu avec un seul bloc de résumé.

EncryptedData – une partie de contenu chiffré, avec les matériaux de chiffrement fournis en externe.

On ne respecte pas ici les conventions par égard pour les lecteurs qui ne sont pas familiarisé avec la lecture de l'ASN.1, et on présente une syntaxe pour CMS en BNF informel (avec beaucoup de glose). Dans le codage réel, la plupart des productions ont une étiquette explicite et des champs de longueur.

```
Message = *Content
Content = Data | SignedData | EnvelopedData | DigestedData | EncryptedData
Data = Bytes
SignedData = *DigestAlg Content *Certificates *CRLs SignerInfo*
EnvelopedData = *RecipientInfo BulkCryptAlg Encrypted(Content)
```

DigestedData = DigestAlg Content DigestBytes
EncryptedData = BulkCryptAlg Encrypted(Bytes)
SignerInfo = CertID ... Encrypted(DigestBytes) ...
RecipientInfo = CertID KeyCryptAlg Encrypted(DEK)

Appendice B Type de support Internet message/s-http

En plus de la définition du protocole S-HTTP/1.4, le présent document sert de spécification pour le type de support Internet "message/s-http". Ce qui suit est à enregistrer par l'IANA.

Nom de type de support :	message
Nom de sous-type de support :	s-http
Paramètres exigés :	aucun
Paramètres facultatifs :	version, msgtype
Version :	numéro de version S-HTTP du message inclus (par exemple, "1.4"). Si il n'est pas présent, la version peut être déterminée d'après la première ligne du corps.
msgtype :	type du message -- "demande" ou "réponse". S'il n'est pas présent, le type peut être déterminé d'après la première ligne du corps.
Considérations de codage :	seuls "7bit", "8bit", ou "binary" sont permis.
Considérations de sécurité :	c'est un protocole de sécurité.

Bibliographie et références

[FIPS-46-1] Federal Information Processing Standards Publication 46-1, "Data Encryption Standard", modifiée le 22 janvier 1988 (remplace FIPS PUB 46, 15 janvier 1977).

[FIPS-81] Federal Information Processing Standards Publication 81, "DES Modes of Operation", 2 décembre 1980.

[FIPS-180] Federal Information Processing Standards Publication 180-1, "Secure Hash Standard", 17 avril 1995.

[FIPS-186] Federal Information Processing Standards Publication 186, "Digital Signature Standard", 19 mai 1994.

[HAST86] Hastad, J., "On Using RSA With Low Exponents in a Public Key Network," Advances in Cryptology – CRYPTO 95 Proceedings, Springer-Verlag, 1986.

[JOHN93] Johnson, D.B., Matyas, S.M., Le, A.V., Wilkins, J.D., "Design of the Commercial Data Masking Facility Data Privacy Algorithm," Proceedings 1st ACM Conference on Computer & Communications Security, novembre 1993, Fairfax, VA., pp. 93-96.

[KRAW96b] Krawczyk, H. communication personnelle.

[LAI92] Lai, X. "On the Design and Security of Block Ciphers," ETH Series in Information Processing, v. 1, Konstanz: Hartung-Gorre Verlag, 1992.

[PKCS-6] RSA Data Security, Inc. "Extended Certificate Syntax Standard", PKCS-6, 1 novembre 1993.

[RFC0822] D. Crocker, "Norme pour le format des messages de texte de l'ARPA-Internet", STD 11, août 1982. (*Obsolète, remplacée par la RFC5322*)

[RFC1319] B. Kaliski, "Algorithme de résumé de message MD2", avril 1992. *(Historique, Information)*

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

[RFC1421] J. Linn, "Amélioration de la confidentialité pour la messagerie électronique Internet : Partie I : Chiffrement de message et procédures d'authentification", février 1993. *(Historique)*

[RFC1422] S. Kent, "Amélioration de la confidentialité pour la messagerie électronique Internet : Partie II – Gestion de clés fondée sur le certificat", février 1993. *(Historique)*

[RFC1738] T. Berners-Lee et autres, "Localisateurs uniformes de ressource (URL)", décembre 1994. *(P.S., Obsolète, voir les*

RFC4248 et 4266)

[RFC1779] S. Kille, "Représentation de chaîne des noms distinctifs", mars 1995. *(Obsolète, voir* RFC2253, RFC3494) *(Historique)*

[RFC1847] J. Galvin, S. Murphy, S. Crocker, N. Freed, "Sécurité multiparties pour MIME : multipartie/signée et multipartie/chiffrée", octobre 1995. *(P.S.)*

[RFC1848] S. Crocker, N. Freed, J. Galvin et S. Murphy, "Services de sécurité d'objet MIME", octobre 1995. *(Historique)*

[RFC1864] J. Myers, M. Rose, "Champ d'en-tête Contenu-MD5", octobre 1995. *(D.S.)*

[RFC2045] N. Freed et N. Borenstein, "Extensions de messagerie Internet multi-objets (MIME) Partie 1 : Format des corps de message Internet", novembre 1996. *(D. S., MàJ par* 2184, 2231, 5335.)

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

[RFC2615] A. Malis, W. Simpson, "PPP sur SONET/SDH", juin 1999. *(P.S.)*

[RFC2616] R. Fielding et autres, "Protocole de transfert hypertexte -- HTTP/1.1", juin 1999. *(D.S., MàJ par* 2817, 6585)

[RFC2630] R. Housley, "Syntaxe de message cryptographique", juin 1999. *(Obsolète, voir* 3369, 3370) *(P.S.)*

[RFC2659] E. Rescorla, A. Schiffman, "Extensions de sécurité pour HTML", août 1999. *(Expérimentale)*

[VANO95] B. Prennel and P. van Oorschot, "On the security of two MAC algorithms", à parraître dans Eurocrypt'96.

[X509] Recommandation UIT-T X.509, "L'annuaire – cadre d'authentification", (1988).

Considérations sur la sécurité

Ce document tout entier traite de la sécurité.

Adresse des auteurs

Eric Rescorla
RTFM, Inc.
30 Newell Road, #16
East Palo Alto, CA 94303
téléphone : (650) 328-8631
mél : ekr@rtfm.com

Allan M. Schiffman
SPYRUS/Terisa
5303 Betsy Ross Drive
Santa Clara, CA 95054
téléphone : (408) 327-1901
mél : ams@terisa.com

15. Déclaration complète de droits de reproduction

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