Internet Engineering Task Force (IETF)
Request for Comments : 6376
STD 76
RFC rendues obsolètes : 4871, 5672
Catégorie : Norme
ISSN : 2070-1721

D. Crocker, éd., Brandenburg InternetWorking
T. Hansen, éd., AT&T Laboratories
M. Kucherawy, éd., Cloudmark
septembre 2011

Traduction Claude Brière de L'Isle

Signatures de messagerie identifiées par clés de domaine (DKIM)

Résumé
La messagerie identifiée par clés de domaine (DKIM, *clés de domaine Identified Mail*) permet à une personne, rôle, ou organisation qui possède le domaine signant de revendiquer la responsabilité d'un message en associant le domaine au message. Ce peut être l'organisation d'un auteur, un relais de fonctionnement, ou un de leurs agents. DKIM sépare la question de l'identité du signataire du message de l'auteur prétendu du message. L'assertion de responsabilité est validée par une signature cryptographique et en interrogeant directement le domaine du signataire pour restituer la clé publique appropriée. Le message transite de l'auteur au receveur par des relais qui ne font normalement aucun changement substantiel au contenu du message et donc préserve la signature DKIM.

Le présent mémoire rend obsolètes les RFC 4871 et RFC 5672.

Statut de ce mémoire
Ceci est un document de l'Internet sur la voie de la normalisation.

Le présent document a été produit par l'équipe d'ingénierie de l'Internet (IETF). Il représente le consensus de la communauté de l'IETF. Il a subi une révision publique et sa publication a été approuvée par le groupe de pilotage de l'ingénierie de l'Internet (IESG). Tous les documents approuvés par l'IESG ne sont pas candidats à devenir une norme de l'Internet ; voir la Section 2 de la RFC5741.

Les informations sur le statut actuel du présent document, tout errata, et comment fournir des réactions sur lui peuvent être obtenues à http://www.rfc-editor.org/info/rfc6376

Table des matières

1. Introduction

La messagerie identifiée par clés de domaine (DKIM, *Domain Keys Identified Mail*) permet à une personne, rôle, ou organisation de revendiquer une certaine responsabilité pour un message en associant un nom de domaine [RFC1034] au message [RFC5322], qu'ils sont autorisés à utiliser. Cela peut être l'organisation d'un auteur, un relais opérationnel, ou un de leurs agents. L'affirmation de responsabilité est validée par une signature cryptographique et en interrogeant directement le domaine du signataire pour restituer la clé publique appropriée. Le transit du message de l'auteur au receveur est effectué par des relais qui ne font normalement pas de changement de substance au contenu du message et donc préservent la signature DKIM. Un message peut contenir plusieurs signatures, de la même ou d'organisations différentes impliquées dans le message.

L'approche de DKIM diffère des précédentes approches de la signature de message (par exemple, les extensions de messagerie Internet sécurisées/multi-objets (S/MIME, *Secure/Multipurpose Internet Mail Extensions*) [RFC5751], OpenPGP [RFC4880]) en ce que :

- la signature de message est écrite comme un champ d'en-tête de message afin que ni les receveurs humains ni le logiciel d'agent d'utilisateur de messagerie (MUA, *Mail User Agent*) ne soient troublés par un contenu en rapport avec la signature apparaissant dans le corps de message ;
- il n'y a pas de dépendance aux paires de clés publique et privée produites par des autorités de certification bien connues et de confiance ;
- il n'y a pas de dépendance au déploiement de nouveaux protocoles ou services de l'Internet pour la distribution ou révocation de clé publique ;
- l'échec de vérification de signature ne force pas le rejet du message ;
- aucune tentative n'est faite d'inclure le chiffrement au titre du mécanisme ; et
- l'archivage de message n'est pas un objectif de conception.

DKIM :

- est compatible avec l'infrastructure existante de messagerie et transparent dans la plus grande mesure possible ;
- exige une nouvelle infrastructure minimale ;
- peut être mis en œuvre indépendamment des clients afin de réduire le temps de déploiement ;
- peut être déployé de façon incrémentaire ; et
- permet la délégation de signature à des tiers.

1.1 Documents de l'architecture DKIM

Il est conseillé au lecteur d'être familiarisé avec le matériel des [RFC4686], [RFC5585], et [RFC5863], qui fournissent, respectivement, les fondements du développement de DKIM, une vue d'ensemble du service, des lignes directrices pour le déploiement et le fonctionnement, et des conseils.

1.2 Identité signante

DKIM sépare la question de l'identité du signataire du message de l'auteur prétendu du message. En particulier, une signature inclut l'identité du signataire. Les vérificateurs peuvent utiliser les informations signantes pour décider comment ils veulent traiter le message. L'identité signante est incluse au titre du champ d'en-tête Signature.

Raison (pour information) : l'identité signate spécifiée par une signature DKIM n'est pas obligée de correspondre à une adresse dans un champ d'en-tête particulier à cause de la diversité des méthodes d'interprétation par les systèmes de messagerie receveurs, incluant les MUA.

1.3 Adaptabilité

DKIM est conçu pour prendre en charge les exigences extrêmes d'adaptabilité qui caractérisent le problème de l'identification dans la messagerie électronique. Il y a des millions de domaines et un nombre bien plus grand d'adresses individuelles.

DKIM cherche à préserver les aspects positifs de l'infrastructure actuelle de messagerie électronique, comme la capacité pour chacun de communiquer avec n'importe qui d'autre sans introduction.

1.4 Simple gestion de clé

DKIM diffère des systèmes traditionnels de clé publique hiérarchique en ce que aucune infrastructure d'autorité de certificats n'est requise ; le vérificateur demande la clé publique à un répertoire dans le domaine du signataire prétendu directement plutôt qu'à un tiers.

Le DNS est proposé comme mécanisme initial pour les clés publiques. Donc, DKIM dépend actuellement de l'administration du DNS et de la sécurité du système DNS. DKIM est conçu pour être extensible aux autres services qui vont chercher les clés lorsque ils deviennent disponibles.

1.5 Intégrité des données

Une signature DKIM associe le nom "d=" au hachage calculé de tout ou partie du message (voir au paragraphe 3.7) afin d'empêcher la réutilisation de la signature avec des messages différents. La vérification de signature affirme que le contenu haché n'a pas changé depuis qu'il a été signé et n'affirme rien d'autre sur la "protection" de l'intégrité de bout en bout du message.

2. Terminologie et définitions

Cette Section définit les termes utilisés dans le reste du document.

DKIM est conçu pour opérer dans le service de messagerie de l'Internet, comme défini dans la [RFC5598]. La terminologie de base de la messagerie électronique est tirée de cette spécification.

Les descriptions de syntaxe utilisent le BNF augmenté (ABNF) [RFC5234].

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], [RFC8174] quand, et seulement quand ils apparaissent tout en majuscules, comme montré ci-dessus.

2.1 Signataires

Les éléments du système de messagerie qui signent les messages au nom d'un domaine sont appelés les signataires. Ce peuvent être des agents d'utilisateur de messagerie (MUA, *Mail User Agent*) des agents de soumission de messagerie (MSA, *Mail Submission Agent*) des agents de transfert de messagerie (MTA, *Mail Transfer Agent*) ou d'autres agents comme des éclateurs de liste de diffusion. En général, tout signataire va être impliqué dans l'injection d'un message dans le système de messages d'une certaine façon. La question clé est qu'un message doit être signé avant qu'il quitte le domaine administratif du signataire.

2.2 Vérificateurs

Les éléments du système de messagerie qui vérifient les signatures sont appelés les vérificateurs. Ce peuvent être des MTA, des agents de livraison de messagerie (MDA, *Mail Delivery Agent*) ou des MUA. Dans la plupart des cas, on s'attend à ce que les vérificateurs soient proches d'un utilisateur final (lecteur) du message ou un agent consommateur comme un éclateur de liste de diffusion.

2.3 Identité

Une personne, un rôle, ou une organisation. Dans le contexte de DKIM, les exemples incluent l'auteur, l'organisation de l'auteur, un FAI le long du chemin de traitement, un service indépendant d'attestation de confiance, et un opérateur de liste de diffusion.

2.4 Identifiant

Étiquette qui se réfère à une identité.

2.5 Identifiant de domaine signant (SDID)

Un seul nom de domaine qui est le résultat de charge utile obligatoire de DKIM et qui se réfère à l'identité qui revendique une certaine responsabilité sur le message en le signant. Il est spécifié au paragraphe 3.5.

2.6 Identifiant d'agent ou d'utilisateur (AUID)

Un seul identifiant qui se réfère à l'agent ou utilisateur au nom duquel l'identifiant de domaine signant (SDID, *Signing Domain Identifier*) a pris la responsabilité. Le AUID comporte un nom de domaine et une <partie-locale> facultative. Le nom de domaine est le même que celui utilisé pour le SDID ou en est un sous domaine. Pour le traitement DKIM, la portion nom de domaine de l'AUID a seulement la sémantique de nom de domaine de base ; toute sémantique possible spécifique du possesseur sort du domaine d'application de DKIM. Il est spécifié au paragraphe 3.5.

Noter que les valeurs acceptables pour l'AUID peuvent être contraintes via un fanion dans l'enregistrement de clé publique. (Voir le paragraphe 3.6.1.)

2.7 Témoin d'identité

Élément dans le système de messagerie qui consomme la charge utile DKIM, qui est l'identifiant de domaine signant (SDID, *Signing Domain Identifier*) responsable. Le témoin d'identité est dédié à l'attestation de l'identifiant livré.

D'autres valeurs DKIM (et non DKIM) peuvent aussi être utilisées par le témoin d'identité (si elles sont disponibles) pour fournir un moteur plus général de filtrage d'évaluation de message. Cependant, cette activité supplémentaire sort du domaine d'application de la présente spécification.

2.8 Espace

Il y a trois formes d'espace :

- WSP représente l'espace simple, c'est-à-dire, un caractère espace ou une tabulation (la définition formelle est dans la [RFC5234]).

- o LWSP est une espace linéaire, définie comme WSP plus CRLF (la définition formelle est dans la [RFC5234])..
- o FWS est une espace de saut à la ligne. Elle permet que plusieurs lignes séparées par un CRLF suivi par au moins une espace, soient jointes.

L'ABNF formel pour elles est (WSP et LWSP sont données pour information seulement) :

```
WSP = SP / HTAB
LWSP = *(WSP / CRLF WSP)
FWS = [*WSP CRLF] 1*WSP
```

La définition de FWS est identique à celle de la [RFC5322] sauf pour l'exclusion de obs-FWS.

2.9 Jetons importés de l'ABNF

Les jetons suivants sont importés d'autres RFC comme noté. Ces RFC devraient être considérées comme d'autorité.

Les jetons suivants sont importés de la [RFC5321]:
- o "local-part" (avertissement de mise en œuvre : cela permet des chaînes entre guillemets)
- o "sub-domain"

Les jetons suivants sont importés de la [RFC5322]:
- o "field-name" (nom d'un champ d'en-tête)
- o "dot-atom-text" (dans la partie locale d'une adresse de messagerie électronique)

Les jetons suivants sont importés de la [RFC2045]:
- o "qp-section" (une seule ligne de texte quoted-printable-encoded)
- o "hex-octet" (octet codé en quoted-printable)

Note pour information : il faut savoir que l'ABNF de la [RFC2045] n'obéit pas aux règles de la [RFC5234] et doit être interprété en conséquence, en particulier en ce qui concerne le repli de casse.

D'autres jetons non définis ici sont importés de la [RFC5234]. Ce sont des primitives intuitives comme SP, HTAB, WSP, ALPHA, DIGIT, CRLF, etc.

2.10 Jetons ABNF communs

Les jetons ABNF suivants sont utilisés ailleurs dans ce document :

```
hyphenated-word = ALPHA [ *(ALPHA / DIGIT / "-") (ALPHA / DIGIT) ]
ALPHADIGITPS =  (ALPHA / DIGIT / "+" / "/")
base64string = ALPHADIGITPS *([FWS] ALPHADIGITPS) [ [FWS] "=" [ [FWS] "=" ] ]
hdr-name = field-name
qp-hdr-value = dkim-quoted-printable                    ; avec "|" codé
```

2.11 DKIM-Quoted-Printable

La syntaxe du codage de DKIM-Quoted-Printable ressemble à celle décrite dans Quoted-Printable [RFC2045], paragraphe 6.7 : tout caractère PEUT être codé comme un "=" suivi par deux chiffres hexadécimaux provenant de l'alphabet "0123456789ABCDEF" (pas de caractère minuscule permis) représentant la valeur d'entier codée en hexadécimal de ce caractère. Tous les caractères de contrôle (ceux avec des valeurs supérieures à %x20) les caractères de 8 bits (valeurs supérieures à %x7F), et les caractères DEL (%x7F), ESPACE (%x20), et point-virgule (";", %x3B) DOIVENT être codés. Noter que toutes les espaces, incluant les caractères SPACE, CR, et LF, DOIVENT être codées. Après le codage, FWS PEUT être ajouté à des endroits arbitraires afin d'éviter des lignes excessivement longues ; cette espace NE fait PAS partie de la valeur, et DOIT être supprimée avant le décodage. L'utilisation de caractères ne figurant pas comme "sûrs pour la messagerie" dans la [RFC2049] N'EST PAS RECOMMANDÉE.

ABNF :

dkim-quoted-printable = *(FWS / hex-octet / dkim-safe-char) ; hex-octet vient de la RFC2045
dkim-safe-char = %x21-3A / %x3C / %x3E-7E ; '!' - ':', '<', '>' - '~'

Note pour information : DKIM-Quoted-Printable diffère de Quoted-Printable comme défini dans la [RFC2045] de plusieurs façons importantes :

1. Les espaces dans le texte d'entrée, y compris CR et LF, doivent être codées. La [RFC2045] n'exige pas un tel codage, et ne permet pas de coder les caractères CR ou LF qui font partie d'un CRLF de saut à la ligne.
2. Les espaces dans le texte codé sont ignorées. C'est pour permettre que des étiquettes codées en utilisant DKIM-Quoted-Printable soient enveloppées comme nécessaire. En particulier, la [RFC2045] exige que les coupures de ligne dans l'entrée soient représentées comme des coupures de ligne physiques ; ce n'est pas le cas ici.
3. La syntaxe de "saut à la ligne doux" ("=" comme dernier caractère non espace sur la ligne) ne s'applique pas.
4. DKIM-Quoted-Printable n'exige pas que les lignes codées ne fassent pas plus de 76 caractères (bien qu'il puisse y avoir d'autres exigences selon le contexte dans lequel le texte codé est utilisé).

3. Éléments du protocole

Les éléments du protocole sont des parties conceptuelles du protocole qui ne sont pas spécifiques des signataires ou des vérificateurs. Les descriptions du protocole pour les signataires et les vérificateurs sont dans les sections suivantes ("Actions du signataire" (Section 5) et "Actions du vérificateur" (Section 6)). Note : cette section doit être lue dans le contexte de ces sections.

3.1 Sélecteurs

Pour prendre en charge plusieurs clés publiques concurrentes par domaine signant, l'espace de noms de clés est subdivisé en utilisant des "sélecteurs". Par exemple, les sélecteurs pourraient indiquer les noms des localisations de bureaux (par exemple, "sanfrancisco", "coolumbeach", et "reykjavik") la date de signature (par exemple, "janvier2005", "février2005", etc.) ou même un utilisateur individuel.

Les sélecteurs sont nécessaires pour prendre en charge des cas d'utilisation importants. Par exemple :

- Les domaines qui veulent déléguer à un partenaire la capacité de signer pour une adresse spécifique pour une certaine durée, comme un fournisseur d'annonces ou autre fonction externalisée.

- Les domaines qui veulent permettre à des voyageurs fréquents d'envoyer des messages localement sans avoir besoin de se connecter à un MSA particulier.

- Les domaines "d'affinité" (par exemple, des associations d'anciens élèves de collège) qui fournissent la transmission de messages entrants, mais ne font pas l'agent de soumission de messagerie pour les messages sortants.

Des points sont permis dans les sélecteurs et sont des séparateurs de composants. Quand des clés sont restituées du DNS, les points dans les sélecteurs définissent des limites d'étiquette DNS d'une manière similaire à l'utilisation conventionnelle dans les noms de domaines. Les composants de sélecteur peuvent être utilisés pour combiner des dates avec des localisations, par exemple, "mars2005.reykjavik". Dans une mise en œuvre du DNS, cela peut être utilisé pour permettre la délégation d'une portion de l'espace de noms du sélecteur.

ABNF :

selector = sous domaine *("." sous domaine)

Le nombre de clés publiques et de sélecteurs correspondants pour chaque domaine est déterminé par le propriétaire du domaine. De nombreux propriétaires de domaine vont être satisfaits avec juste un sélecteur, tandis que les organisations à répartition administrative peuvent choisir de gérer des sélecteurs disparates et des paires de clés dans différentes régions ou sur différents serveurs de messagerie.

Au delà de l'aspect administratif, les sélecteurs rendent possible un remplacement en douceur des clés publiques de façon routinière. Si un domaine souhaite changer de l'utilisation d'une clé publique associée au sélecteur "janvier2005" à une clé publique associée au sélecteur "février2005", il s'assure simplement que les deux clés publiques sont annoncées

concurremment dans le répertoire de clés publiques pendant la période de transition durant laquelle un message peut être en transit avant la vérification. Au début de la période de transition, les serveurs de messages sortants sont configurés à signer avec la clé privée "février2005". À la fin de la période de transition, la clé publique "janvier2005" est supprimée du répertoire de clés publiques.

Note pour information : une clé peut aussi être révoquée comme décrit ci-dessous. La distinction entre révoquer et supprimer un enregistrement de sélecteur de clé est subtile. Quand on périme les clés comme décrit ci-dessus, un domaine signant va probablement simplement supprimer l'enregistrement de clé après la période de transition. Cependant, un domaine signant pourrait choisir de révoquer la clé (mais de maintenir l'enregistrement de clé) encore un peu. Il n'y a pas de différence sémantique définie entre une clé révoquée et une clé supprimée.

Alors que certains domaines peuvent souhaiter faire que les valeurs de sélecteur soient bien connues, d'autres vont vouloir allouer les noms de sélecteur d'une façon qui ne permette pas la collecte des données par des parties tierces. Par exemple, si des clés sont produites par utilisateur, le propriétaire du domaine va devoir décider si il associe ce sélecteur directement au nom d'un utilisateur final enregistré ou si il prend une valeur aléatoire non associée, comme une empreinte digitale de la clé publique.

Note de fonctionnement pour information : réutiliser un sélecteur avec une nouvelle clé (par exemple, en changeant la clé associée au nom d'un utilisateur) rend impossible de faire la différence entre un message qui ne se vérifie pas parce que la clé n'est plus valide et un message qui est en fait falsifié. Pour cette raison, les signataires sont mal avisés de réutiliser les sélecteurs pour de nouvelles clés. Une meilleure stratégie est d'allouer de nouvelles clés à de nouveaux sélecteurs.

3.2 Listes de Tag=Value

DKIM utilise une simple syntaxe de "tag=value" dans plusieurs contextes, incluant dans les enregistrements de signature de messages et de domaines.

Les valeurs sont une série de chaînes contenant du texte en clair, du texte"base64" (comme défini dans la [RFC2045] au paragraphe 6.8) "qp-section" (idem, paragraphe 6.7), ou "dkim-quoted-printable" (comme défini au paragraphe 2.11). Le nom de l'étiquette va déterminer le codage de chaque valeur. Des caractères point-virgule non codés (";") NE DOIVENT PAS apparaître dans la valeur d'étiquette, car cela sépare les spécifications d'étiquettes.

Note de mise en œuvre pour information : bien que le "plain text" défini ci-dessous (comme "tag-value") inclue seulement des caractères de 7 bits, il est conseillé à une mise en œuvre qui souhaiterait anticiper de futures normes de ne pas empêcher l'utilisation de texte codé en UTF-8 [RFC3629] dans les listes de tag=value.

Formellement, les règles de la syntaxe ABNF sont les suivantes :

```
tag-list =  tag-spec *( ";" tag-spec ) [ ";" ]
tag-spec = [FWS] tag-name [FWS] "=" [FWS] tag-value [FWS]
tag-name = ALPHA *ALNUMPUNC
tag-value = [ tval *( 1*(WSP / FWS) tval ) ]          ; interdit WSP et FWS au début et à la fin
tval = 1*VALCHAR
VALCHAR = %x21-3A / %x3C-7E                           ; de EXCLAMATION à TILDE sauf SEMICOLON
ALNUMPUNC = ALPHA / DIGIT / "_"
```

Noter que WSP est permis partout autour des étiquettes. En particulier, toute WSP après le "=" et avant le ";" de fin ne fait pas partie de la valeur ; cependant, une WSP à l'intérieur de la valeur est significative.

Les étiquettes DOIVENT être interprétée de manière insensible à la casse. Les valeurs DOIVENT être traitées comme sensibles à la casse sauf si la description de la sémantique spécifique de l'étiquette spécifie l'insensibilité à la casse.

Les étiquettes avec des noms dupliqués NE DOIVENT PAS se produire dans une seule liste d'étiquettes ; si un nom d'étiquette se produit plus d'une fois, la liste d'étiquettes entière est invalide.

Les espaces dans une valeur DOIVENT être conservées sauf si elles sont explicitement exclues par la description spécifique de l'étiquette.

Les paires tag=value qui représentent la valeur par défaut PEUVENT être incluses pour faciliter la lisibilité.

Les étiquettes non reconnues DOIVENT être ignorées.

Les étiquettes qui ont une valeur vide ne sont pas la même chose que les étiquettes omises. Une étiquette omise est traitée comme ayant la valeur par défaut ; une étiquette avec une valeur vide désigne explicitement la chaîne vide comme valeur.

3.3 Algorithmes de signature et de vérification

DKIM prend en charge plusieurs algorithmes de signature numérique. Deux algorithmes sont définis pour l'instant par la présente spécification : rsa-sha1 et rsa-sha256. Les signataires DOIVENT mettre en œuvre et DEVRAIENT signer en utilisant rsa-sha256. Les vérificateurs DOIVENT mettre en œuvre rsa-sha1 et rsa-sha256.

Note pour information : bien que rsa-sha256 soit fortement conseillé, certains envoyeurs pourraient préférer utiliser rsa-sha1 quand ils mettent en balance la force de la sécurité avec les performances, la complexité, ou d'autres besoins. En général cependant, rsa-sha256 devrait toujours être utilisé chaque fois que possible.

3.3.1 Algorithme de signature rsa-sha1

L'algorithme de signature rsa-sha1 calcule un hachage de message comme décrit au paragraphe 3.7 en utilisant SHA-1 [FIPS-180-3] comme algorithme de hachage. Ce hachage est alors signé par le signataire en utilisant l'algorithme RSA (défini dans les normes de chiffrement à clé publique (PKCS, *Public-Key Cryptography Standards*) n° 1, version 1.5 [RFC3447]) comme algorithme de chiffrement et la clé privée du signataire. Le hachage NE DOIT PAS être tronqué ou converti en une forme autre que la forme binaire native avant d'être signé. L'algorithme de signature DEVRAIT utiliser un exposant public de 65537.

3.3.2 Algorithme de signature rsa-sha256

L'algorithme de signature rsa-sha256 calcule un hachage de message comme décrit au paragraphe 3.7 en utilisant SHA-256 [FIPS-180-3] comme algorithme de hachage. Ce hachage est alors signé par le signataire en utilisant l'algorithme RSA (défini dans les normes de chiffrement à clé publique (PKCS, *Public-Key Cryptography Standards*) n° 1, version 1.5 [RFC3447]) comme algorithme de chiffrement et la clé privée du signataire. Le hachage NE DOIT PAS être tronqué ou converti en une forme autre que la forme binaire native avant d'être signé. L'algorithme de signature DEVRAIT utiliser un exposant public de 65537.

3.3.3 Tailles de clé

Le choix des tailles de clé appropriées est un compromis entre coût, performances, et risque. Comme les courtes clés RSA succombent plus facilement aux attaques hors ligne, les signataires DOIVENT utiliser des clés RSA d'au moins 1024 bits pour les clés à longue durée. Les vérificateurs DOIVENT être capables de valider des signatures avec des clés allant de 512 bits à 2048 bits, et ils PEUVENT être capables de valider des signatures avec de plus grandes clés. Les politiques de vérificateur peuvent utiliser la longueur de la clé de signature comme métrique pour déterminer si une signature est acceptable.

Les facteurs qui devraient influencer le choix de la taille de clé incluent :
- La contrainte pratique que de grandes clés (par exemple, 4096 bits) pourraient ne pas tenir dans un paquet de réponse du DNS de 512 octets.
- La contrainte de sécurité que des clés plus petites que 1024 bits sont sujettes à des attaques hors ligne.
- Les plus grandes clés imposent des coûts de CPU plus élevés pour vérifier et signer les messages.
- Les clés peuvent être remplacées de façon régulière ; donc, leur durée de vie peut être relativement courte.
- Les objectifs de sécurité de cette spécification sont modestes comparés aux objectifs normaux des autres systèmes qui emploient des signatures numériques.

Voir dans la [RFC3766] la discussion sur le choix des tailles de clés.

3.3.4 Autres algorithmes

D'autres algorithmes PEUVENT être définis à l'avenir. Les vérificateurs DOIVENT ignorer toute signature qui utilise un

algorithme qu'ils ne mettent pas en œuvre.

3.4 Canonisation

Certains systèmes de messagerie modifient les messages en transit, invalidant potentiellement une signature. Pour la plupart des signataires, une légère modification de message est imperceptible pour la validation de l'utilisation du nom de domaine DKIM. Pour de tels signataires, un algorithme de canonisation qui survit à une modeste modification en transit est préférée.

D'autres signataires demandent que toute modification de message, même mineure, résulte en un échec de vérification de signature. Ces signataires préfèrent un algorithme de canonisation qui ne tolère pas de modification en transit du message signé.

Certains signataires peuvent vouloir accepter des modifications aux champs d'en-tête qui sont dans les limites des normes de messagerie comme celles de la [RFC5322], mais ne veulent pas accepter de modification au corps des messages.

Pour satisfaire toutes les exigences, deux algorithmes de canonisation sont définis pour chaque en-tête et le corps : un algorithme "simple" qui ne tolère presque aucune modification et un algorithme "lâche" qui tolère les modifications courantes comme le remplacement des espaces et la remise à la ligne de champ d'en-tête. Un signataire PEUT spécifier l'un ou l'autre algorithme pour l'en-tête ou le corps quand il signe un message. Si aucun algorithme de canonisation n'est spécifié par le signataire, l'algorithme "simple" est par défaut pour l'en-tête et le corps. Les vérificateurs DOIVENT mettre en œuvre les deux algorithmes de canonisation. Noter que l'en-tête et le corps peuvent utiliser des algorithmes de canonisation différents. D'autres algorithmes de canonisation PEUVENT être définis à l'avenir ; les vérificateurs DOIVENT ignorer toute signature qui utilise des algorithmes de canonisation non reconnus.

La canonisation prépare simplement le message pour la présentation à l'algorithme de signature ou de vérification. Elle NE DOIT en aucune façon changer les données transmises. La canonisation des champs d'en-tête et du corps est décrite ci-dessous.

Note : ce paragraphe suppose que le message est déjà en format "normal du réseau" (le texte est codé en ASCII, les lignes sont séparées par des caractères CRLF, etc.). Voir aussi le paragraphe 5.3 pour des informations sur la normalisation du message.

3.4.1 Algorithme "simple" de canonisation d'en-tête

L'algorithme de canonisation d'en-tête "simple" ne change d'aucune manière les champs d'en-tête. Les champs d'en-tête DOIVENT être présentés à l'algorithme de signature ou de vérification exactement comme ils sont dans le message à signer ou vérifier. En particulier, les noms de champ d'en-tête NE DOIVENT PAS être changés de casse et les espaces NE DOIVENT PAS être changées.

3.4.2 Algorithme "lâche" de canonisation d'en-tête

L'algorithme de canonisation d'en-tête "lâche" DOIT appliquer dans l'ordre les étapes suivantes :

- Convertir tous les noms de champ d'en-tête (pas les valeurs de champ d'en-tête) en minuscules. Par exemple, convertir "SUBJect: AbC" en "subject: AbC".

- Déplier toutes les lignes de continuation de champ d'en-tête comme décrit dans la [RFC5322] ; en particulier, les lignes avec des terminaisons incorporées dans des valeurs de champ d'en-tête continués (c'est-à-dire, des séquences de CRLF suivies par un WSP) DOIVENT être interprétées sans le CRLF. Les mises en œuvre NE DOIVENT PAS supprimer le CRLF à la fin de la valeur du champ d'en-tête.

- Convertir toutes les séquences d'un ou plusieurs caractères WSP en un seul caractère SP. Les caractères WSP incluent ici ceux avant et après une limite de saut à la ligne.

- Supprimer tous les caractères WSP à la fin de chaque valeur de champ d'en-tête non déplié.

- Supprimer tous les caractères WSP restants avant et après les deux-points qui séparent le nom de champ d'en-tête de la valeur de champ d'en-tête. Les deux-points de séparateur DOIVENT être conservés.

3.4.3 Algorithme "simple" de canonisation de corps

L'algorithme de canonisation de corps "simple" ignore toutes les lignes vides à la fin du corps de message. Un ligne vide est une ligne de longueur zéro après suppression de la terminaison de ligne. Si il n'y a pas de corps ou de CRLF en queue sur le corps de message, un CRLF est ajouté. Il ne fait pas d'autre changement au corps de message. En des termes plus formels, l'algorithme de canonisation de corps "simple" convertit "*CRLF" à la fin du corps en un seul "CRLF".

Noter qu'un corps complètement vide ou un corps manquant est canonisé comme un seul "CRLF" ; c'est-à-dire, la longueur canonisée va être de 2 octets.

La valeur SHA-1 (en base64) pour un corps vide (canonisé en un "CRLF") est :

uoq1oCgLlTqpdDX/iUbLy7J1Wic=

La valeur SHA-256 est :

frcCV1k9oG9oKj3dpUqdJg1PxRT2RSN/XKdLCPjaYaY=

3.4.4 Algorithme "lâche" de canonisation de corps

L'algorithme "lâche" de canonisation de corps DOIT appliquer les étapes suivantes (a) et (b) dans l'ordre :

a. Réduction des espaces :
   * Ignorer toutes les espaces en fin de ligne. Les mises en œuvre NE DOIVENT PAS supprimer le CRLF en fin de ligne.
   * Réduire toutes les séquences de WSP dans une ligne à un seul caractère SP.

b. Ignorer toutes les lignes vides à la fin du corps de message. La "ligne vide" est définie au paragraphe 3.4.3. Si le corps est non vide mais ne se termine pas par un CRLF, un CRLF est ajouté. (Pour la messagerie, cela n'est possible qu'en utilisant des extensions à SMTP ou des mécanismes de transport non SMTP.)

La valeur SHA-1 (en base64) pour un corps vide (canonisé en une entrée nulle) est :

2jmj7l5rSw0yVb/vlWAYkK/YBwk=

La valeur SHA-256 est :

47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=

3.4.5 Exemples de canonisation (information)

Dans les exemples suivants, les espaces réelles ne sont utilisées que pour la lisibilité. Le texte réel d'entrée et de sortie est conçu en utilisant des descripteurs entre crochets : "<SP>" pour un caractère espace, "<HTAB>" pour un caractère de tabulation, et "<CRLF>" pour une séquence retour-chariot/saut-à-la-ligne. Par exemple, "X <SP> Y" et "X<SP>Y" représentent les mêmes trois caractères.

Exemple 1 : un message qui se lit comme :

```
A: <SP> X <CRLF>
B <SP> : <SP> Y <HTAB><CRLF>
         <HTAB> Z <SP><SP><CRLF>
<CRLF>
<SP> C <SP><CRLF>
D <SP><HTAB><SP> E <CRLF>
<CRLF>
<CRLF>
```

quand il est canonisé en utilisant la canonisation lâche pour l'en-tête et le corps résulte en un en-tête qui se lit :

a:X <CRLF>
b:Y <SP> Z <CRLF>

et un corps qui se lit :

<SP> C <CRLF>
D <SP> E <CRLF>

Exemple 2 : le même message canonisé en utilisant la canonisation simple pour l'en-tête et le corps résulte en un en-tête qui se lit :

A: <SP> X <CRLF>
B <SP> : <SP> Y <HTAB><CRLF>
<HTAB> Z <SP><SP><CRLF>

et un corps qui se lit :

<SP> C <SP><CRLF>
D <SP><HTAB><SP> E <CRLF>

Exemple 3 : quand elle est traitée en utilisant la canonisation lâche d'en-tête et la canonisation simple de corps, la version canonisée a un en-tête de :

a:X <CRLF>
b:Y <SP> Z <CRLF>

et un corps qui se lit :

<SP> C <SP><CRLF>
D <SP><HTAB><SP> E <CRLF>

3.5 Champ d'en-tête de signature DKIM

La signature du message est mémorisée dans le champ d'en-tête DKIM-Signature. Ce champ d'en-tête contient toutes les données de signature et de collecte de clé. La valeur de DKIM-Signature est une liste d'étiquettes comme décrit au paragraphe 3.2.

Le champ d'en-tête DKIM-Signature DEVRAIT être traité comme si il était un champ d'en-tête de trace comme défini au paragraphe 3.6 de la [RFC5322] et donc NE DEVRAIT PAS être réordonné et DEVRAIT être ajouté devant le message.

Le champ d'en-tête DKIM-Signature créé ou vérifié est toujours inclus dans le calcul de signature, après le reste des champs d'en-tête signés ; cependant, quand on calcule ou vérifie la signature, la valeur de l'étiquette "b=" (valeur de signature) de ce champ d'en-tête DKIM-Signature DOIT être traitée comme si elle était une chaîne vide. Les étiquettes inconnues dans le champ d'en-tête DKIM-Signature DOIVENT être incluses dans le calcul de signature mais DOIVENT être autrement ignorées par les vérificateurs. Les autres champs d'en-tête DKIM-Signature qui sont inclus dans la signature devraient être traités comme des champs d'en-tête normaux ; en particulier, l'étiquette "b=" n'est pas traitée de façon particulière.

Le codage de chaque type de champ est indiqué ci-dessous. Les étiquettes décrites comme qp-section sont codées comme décrit au paragraphe 6.7 de MIME Partie un [RFC2045], avec la conversion supplémentaire des caractères point-virgule en "=3B" ; intuitivement, c'est une ligne de texte codé en quoted-printable. La syntaxe de dkim-quoted-printable est définie au paragraphe 2.11.

Les étiquettes sur le champ d'en-tête DKIM-Signature avec leur type et l'état d'exigence sont montrées ci-dessous. Les étiquettes non reconnues DOIVENT être ignorées.

v= Version (plain-text ; EXIGÉ). Cette étiquette définit la version de cette spécification qui s'applique à l'enregistrement de signature. Elle DOIT avoir la valeur "1" pour les mises en œuvre conformes à cette version de DKIM.

ABNF :

```
sig-v-tag = %x76 [FWS] "=" [FWS] 1*DIGIT
```

Note pour information : les numéros de version de DKIM-Signature peuvent augmenter arithmétiquement lorsque de nouvelles versions de cette spécification sont publiées.

a= : algorithme utilisé pour générer la signature (plain-text ; EXIGÉ). Les vérificateurs DOIVENT prendre en charge "rsa-sha1" et "rsa-sha256"; Les signataires DEVRAIENT signer en utilisant "rsa-sha256". Voir au paragraphe 3.3 la description des algorithmes.

ABNF :

```
sig-a-tag = %x61 [FWS] "=" [FWS] sig-a-tag-alg
sig-a-tag-alg = sig-a-tag-k "-" sig-a-tag-h
sig-a-tag-k = "rsa" / x-sig-a-tag-k
sig-a-tag-h = "sha1" / "sha256" / x-sig-a-tag-h
x-sig-a-tag-k = ALPHA *(ALPHA / DIGIT)          ; pour extension future
x-sig-a-tag-h = ALPHA *(ALPHA / DIGIT)          ; pour extension future
```

b= : données de signature (base64 ; EXIGÉ). L'espace est ignorée dans cette valeur et DOIT être ignorée quand on réassemble la signature originale. En particulier, le processus de signature peut insérer en toute sécurité des FWS dans cette valeur dans des lieux arbitraires pour se conformer aux limites de longueur de ligne. Voir "Actions de signataire" (Section 5) pour la façon dont la signature est calculée.

ABNF :

```
sig-b-tag = %x62 [FWS] "=" [FWS] sig-b-tag-data
sig-b-tag-data = base64string
```

bh= : le hachage de la partie de corps canonisée du message telle que limitée par l'étiquette "l=" (base64 ; EXIGÉ). L'espace est ignorée dans cette valeur et DOIT être ignorée quand on réassemble la signature originale. En particulier, le processus de signature peut insérer en toute sécurité des FWS dans cette valeur dans des lieux arbitraires pour se conformer aux limites de longueur de ligne. Voir au paragraphe 3.7 la façon dont le hachage du corps est calculé.

ABNF :

```
sig-bh-tag = %x62 %x68 [FWS] "=" [FWS] sig-bh-tag-data
sig-bh-tag-data = base64string
```

c= : canonisation du message (plain-text ; FACULTATIF, par défaut est "simple/simple"). Cette étiquette informe le vérificateur du type de canonisation utilisé pour préparer le message à la signature. Elle consiste en deux noms séparés par un caractère barre oblique (slash, %d47) qui correspond aux algorithmes de canonisation de l'en-tête et du corps, respectivement. Ces algorithmes sont décrits au paragraphe 3.4. Si seulement un algorithme est nommé, cet algorithme est utilisé pour l'en-tête et "simple" est utilisé pour le corps. Par exemple, "c=relaxed" est traité de la même façon que "c=relaxed/simple".

ABNF :

```
sig-c-tag = %x63 [FWS] "=" [FWS] sig-c-tag-alg ["/" sig-c-tag-alg]
sig-c-tag-alg = "simple" / "relaxed" / x-sig-c-tag-alg
x-sig-c-tag-alg = hyphenated-word                    ; pour extension future
```

d= : le SDID qui revendique la responsabilité de l'introduction d'un message dans le flux de messagerie (plain-text ; EXIGÉ). Donc, la valeur du SDID est utilisée pour former l'interrogation sur la clé publique. Le SDID DOIT correspondre à un nom DNS valide sous lequel l'enregistrement de clé DKIM est publié. Les conventions et la sémantique utilisées par un signataire pour créer et utiliser un SDID spécifique sortent du domaine d'application de cette spécification, comme le sont les utilisations de ces conventions et cette sémantique. Quand il est présenté avec une signature qui ne satisfait pas ces exigences, les vérificateurs DOIVENT considérer que la signature est invalide.

Les noms de domaine internationalisés DOIVENT être codés comme des étiquettes A-, comme décrit au paragraphe 2.3 de la [RFC5890].

ABNF :
sig-d-tag = %x64 [FWS] "=" [FWS] domain-name
domain-name = sub-domain 1*("." sub-domain) ; de domain [RFC5321], excluant address-literal

h= : champs d'en-tête signés (plain-text, mais voir la description ; EXIGÉ). Une liste séparée par des caractères deux-points des noms de champ d'en-tête qui identifient les champs d'en-tête présentés à l'algorithme de signature. Le champ DOIT contenir la liste complète des champs d'en-tête dans l'ordre présenté à l'algorithme de signature. Le champ PEUT contenir des noms de champs d'en-tête qui n'existent pas quand il est signé ; les champs d'en-tête non existants ne contribuent pas au calcul de la signature (c'est-à-dire, ils sont traités comme l'entrée nulle, incluant le nom de champ d'en-tête, les deux-points de séparation, la valeur de champ d'en-tête, et tout CRLF de terminaison). Le champ PEUT contenir plusieurs instances d'un nom de champ d'en-tête, ce qui signifie que plusieurs occurrences du champ d'en-tête correspondant sont incluses dans le hachage de l'en-tête. Le champ NE DOIT PAS inclure le champ d'en-tête DKIM-Signature qui est en cours de création ou de vérification mais peut en inclure d'autres. Les espaces de saut à la ligne (FWS) PEUVENT être incluses d'un côté ou de l'autre des deux-points de séparation. Les noms de champ d'en-tête DOIVENT être comparés aux noms réels de champ d'en-tête de façon insensible à la casse. Cette liste NE DOIT PAS être vide. Voir au paragraphe 5.4 la discussion du choix des champs d'en-tête à signer et au paragraphe 5.4.2 les exigences pour signer plusieurs instances d'un seul champ.

ABNF :
sig-h-tag = %x68 [FWS] "=" [FWS] hdr-name *([FWS] ":" [FWS] hdr-name)

Explication pour information : en "signant" des champs d'en-tête qui n'existent pas réellement, un signataire peut permettre à un vérificateur de détecter l'insertion de ces champs d'en-tête après la signature. Cependant, comme un signataire ne peut pas savoir quels champs d'en-tête pourraient être définis à l'avenir, ce mécanisme ne peut pas être utilisé pour empêcher l'ajout de tous les champs d'en-tête inconnus possibles.

Note pour information : "signer" des champs qui ne sont pas présents au moment de la signature non seulement empêche que les champs et valeurs soient ajoutés, mais empêche aussi d'ajouter des champs sans valeur.

i= : identifiant d'agent ou d'utilisateur (AUID, *Agent or User Identifier*) au nom duquel le SDID prend la responsabilité (dkim-quoted-printable ; FACULTATIF, par défaut est une partir locale vide suivie par un "@" suivi par le domaine provenant de l'étiquette "d=").

La syntaxe est une adresse de messagerie électronique standard où la partie locale PEUT être omise. La partie domaine de l'adresse DOIT être la même, ou un sous domaine, que la valeur de l'étiquette "d=".

Les noms de domaine internationalisés DOIVENT être codés comme des étiquettes A-, comme décrit au paragraphe 2.3 de la [RFC5890].

ABNF :
sig-i-tag = %x69 [FWS] "=" [FWS] [Local-part] "@" domain-name

L'AUID est spécifié comme ayant la même syntaxe qu'une adresse de messagerie électronique mais il n'a pas besoin d'avoir la même sémantique. Notamment, le nom de domaine n'a pas besoin d'être enregistré dans le DNS – de sorte qu'il ne pourrait pas résoudre une interrogation -- et la partie locale PEUT être tirée d'un espace de noms sans relation avec une boîte aux lettres. Les détails de la structure et la sémantique de l'espace de noms sont déterminés par le signataire. Toute connaissance ou utilisation de ces détails par les vérificateurs ou témoins sort du domaine d'application de la présente spécification. Le signataire PEUT choisir d'utiliser le même espace de noms pour ses AUID comme ses adresses de messagerie électronique d'utilisateur ou PEUT choisir d'autres moyens de représenter ses utilisateurs. Cependant, le signataire DEVRAIT utiliser le même AUID pour chaque message destiné à être évalué comme étant dans la même sphère de responsabilité, si il souhaite offrir aux receveurs l'option d'utiliser l'AUID comme identifiant stable d'une granularité plus fine que le SDID.

Note pour information : la partie locale de l'étiquette "i=" est facultative parce que dans certains cas, un signataire peut n'être pas capable d'établir une identité individuelle vérifiée. Dans ce cas, le signataire pourrait souhaiter affirmer que bien qu'il veuille aller aussi loin que de signer pour le domaine, il est incapable ou ne veut pas s'engager à un nom d'utilisateur individuel dans le domaine. Il peut le faire en incluant la partie domaine mais pas la partie locale de l'identité.

Discussion pour information : la présente spécification n'exige pas que la valeur de l'étiquette "i=" corresponde à l'identite

dans un champ d'en-tête de message. Ceci est considéré comme étant un problème de politique de vérificateur. Les contraintes entre la valeur de l'étiquette "i=" et les autres identités dans les autres champs d'en-tête cherchent à appliquer l'authentification de base dans la sémantique de confiance associée à un rôle comme celui d'auteur du contenu. La confiance est un sujet large et complexe, et les mécanismes de la confiance sont sujets à des attaques très créatives. L'efficacité réelle de tous les liens sauf les plus basiques entre la valeur "i=" et les autres identités n'est pas bien établie, ni sa vulnérabilité à la subversion par un attaquant. Donc, s'appuyer sur l'utilisation de ces options devrait être strictement limité. En particulier, il n'est pas du tout clair dans quelle mesure un utilisateur d'extrémité receveur typique peut s'appuyer sur des assurances qui pourraient être faites par l'utilisation réussie des options "i=".

l= : compte de longueur de corps (entier décimal non signé en plain-text ; FACULTATIF, par défaut, c'est le corps entier). Cette étiquette informe le vérificateur du nombre d'octets dans le corps du message après la canonisation incluse dans le hachage cryptographique, en commençant à 0 suivant immédiatement le CRLF qui précède le corps. Cette valeur NE DOIT PAS être supérieure au nombre réel d'octets dans le corps de message canonisé. Voir la discussion du paragraphe 8.2.

Note pour information : la valeur de l'étiquette "l=" est contrainte à 76 chiffres décimaux. Cette contrainte n'est pas destinée à prédire la taille des futurs messages ou à exiger que les mises en œuvre utilisent une représentation d'entiers assez grande pour représenter la valeur maximum possible mais est destinée à rappeler à la mise en œuvre de vérifier la longueur de cette étiquette et de toutes les autres durant la vérification et de tester un débordement d'entiers lors du décodage de la valeur. Les mises en œuvre peuvent avoir besoin de limiter la valeur réelle exprimée à une valeur inférieure à 10^76, par exemple, pour permettre qu'un message tienne dans l'espace de mémorisation disponible.

ABNF :
sig-l-tag = %x6c [FWS] "=" [FWS] 1*76DIGIT

q= : liste séparée par des virgules des méthodes d'interrogation utilisées pour restituer la clé publique (plain-text ; FACULTATIF, par défaut est "dns/txt"). Chaque méthode d'interrogation est de la forme "type[/options]", où la syntaxe et la sémantique des options dépendent du type et des options spécifiées. Si il y a plusieurs mécanismes d'interrogation mentionnés, le choix du mécanisme d'interrogation NE DOIT PAS changer l'interprétation de la signature. Les mises en œuvre DOIVENT utiliser les mécanismes d'interrogation reconnus dans l'ordre présenté. Les mécanismes d'interrogation non reconnus DOIVENT être ignorés.

Actuellement, la seule valeur valide est "dns/txt", qui définit l'algorithme de recherche d'enregistrement de ressource (RR, *Resource Record*) DNS TXT décrit ailleurs dans ce document. La seule option définie pour le type d'interrogation "dns" est "txt", qui DOIT être incluse. Les vérificateurs et les signataires DOIVENT prendre en charge "dns/txt".

ABNF :
sig-q-tag = %x71 [FWS] "=" [FWS] sig-q-tag-method *([FWS] ":" [FWS] sig-q-tag-method)
sig-q-tag-method = "dns/txt" / x-sig-q-tag-type ["/" x-sig-q-tag-args]
x-sig-q-tag-type = hyphenated-word ; pour future extension
x-sig-q-tag-args = qp-hdr-value

s= : sélecteur qui subdivise l'espace de noms pour l'étiquette "d=" (domaine) (plain-text ; EXIGÉ).

Les noms de sélecteur internationalisés DOIVENT être codés comme des étiquettes A-, comme décrit au paragraphe 2.3 de la [RFC5890].

ABNF :
sig-s-tag = %x73 [FWS] "=" [FWS] selector

t= : horodatage de signature (entier décimal non signé en plain-text ; RECOMMANDÉ, par défaut est une heure de création inconnue). Heure de création de cette signature. Le format est le nombre de secondes depuis 00:00:00 le 1er janvier 1970 dans la zone horaire UTC. La valeur est exprimée comme un entier non signé en décimal ASCII. Cette valeur n'est pas contrainte à tenir dans un entier de 31 ou 32 bits. Les mises en œuvre DEVRAIENT être prêtes à traiter des valeurs jusqu'à au moins 10^12 (jusqu'à approximativement l'an 200 000 ; cela tient dans 40 bits). Pour éviter des attaques de déni de service, les mises en œuvre PEUVENT considérer que toute valeur de plus de 12 chiffres est infinie. Les sauts de secondes ne sont pas comptés. Les mises en œuvre PEUVENT ignorer les signatures qui ont un horodatage dans le futur.

ABNF :
sig-t-tag = %x74 [FWS] "=" [FWS] 1*12DIGIT

x= : expiration de signature (entier décimal non signé en plain-text ; RECOMMANDÉ, par défaut, pas d'expiration). Le format est le même que dans l'étiquette "t=", représentée comme une date absolue, pas comme une différence de temps par rapport à l'horodatage de la signature. La valeur est exprimée par un entier non signé en ASCII décimal, avec les mêmes contraintes sur la valeur que dans l'étiquette "t=". Les signatures PEUVENT être considérées invalides si l'heure de vérification chez le vérificateur est après la date d'expiration. L'heure de vérification devrait être le moment où le message a été reçu dans le domaine administratif du vérificateur si cette heure est disponible de façon fiable ; autrement, l'heure courante devrait être utilisée. La valeur de l'étiquette "x=" DOIT être supérieure à celle de l'étiquette "t=" si toutes deux sont présentes.

Note pour information : l'étiquette "x=" n'est pas destinée à être une défense contre la répétition.

Note pour information : du fait de la dérive d'horloge, la notion du receveur de quand considérer que la signature a expiré peut ne pas exactement correspondre à ce qu'attend l'envoyeur. Les receveurs PEUVENT ajouter un "facteur d'ajustement" pour permettre la possibilité d'une telle dérive.

ABNF :
sig-x-tag = %x78 [FWS] "=" [FWS] 1*12DIGIT

z= : champs d'en-tête copiés (dkim-quoted-printable, mais voir la description ; FACULTATIF, est nul par défaut). Une liste séparée par des barres verticales des champs d'en-tête choisis présents quand le message a été signé, incluant le nom et la valeur du champ. Il n'est pas exigé d'inclure tous les champs d'en-tête présents au moment de la signature. Ce champ n'a pas besoin de contenir les mêmes champs d'en-tête que dans la liste de l'étiquette "h=". Le texte du champ d'en-tête lui-même doit coder le caractère barre verticale ("|", %x7C) (c'est-à-dire, les barres verticales dans le texte "z=" sont des méta-caractères, et tout caractère barre verticale réel dans un champ d'en-tête copié doit être codé). Noter que toutes les espaces doivent être codées, incluant l'espace entre les deux-points et la valeur de champ d'en-tête. Après le codage, une FWS PEUT être ajoutée à des positions arbitraires afin d'éviter des lignes de longueur excessive ; ces espace NE font PAS partie de la valeur du champ d'en-tête et DOIVENT être supprimées avant le décodage.

Les champs d'en-tête référencés par l'étiquette "h=" se réfèrent aux champs dans l'en-tête [RFC5322] du message, et non à des champs copiés dans l'étiquette "z=". Les valeurs de champs d'en-tête copiés sont des diagnostics.

ABNF :
sig-z-tag = %x7A [FWS] "=" [FWS] sig-z-tag-copy *("|" [FWS] sig-z-tag-copy)
sig-z-tag-copy = hdr-name [FWS] ":" qp-hdr-value

Exemple pour information d'un champ d'en-tête de signature étalé sur plusieurs lignes de continuation :

```
DKIM-Signature: v=1; a=rsa-sha256; d=exemple.net; s=brisbane;
    c=simple; q=dns/txt; i=@eng.exemple.net;
    t=1117574938; x=1118006938;
    h=from:to:subject:date;
    z=From:foo@eng.exemple.net|To:joe@exemple.com|
     Subject:demo=20run|Date:July=205,=202005=203:44:08=20PM=20-0700;
    bh=MTIzNDU2Nzg5MDEyMzQ1Njc4OTAxMjM0NTY3ODkwMTI=;
    b=dzdVyOfAKCdLXdJOc9G2q8LoXSlEniSbav+yuU4zGeeruD00lszZVoG4ZHRNiYzR
```

3.6 Gestion et représentation de clés

Les applications de signature exigent un certain niveau d'assurance que la clé publique de vérification est associée au signataire prétendu. De nombreuses applications réalisent cela en utilisant des certificats de clé publique produits par un tiers de confiance. Cependant, DKIM peut réaliser un niveau de sécurité suffisant, avec une amélioration significative de l'adaptabilité, en ayant simplement le vérificateur qui interroge l'entrée de DNS du signataire prétendu (ou un équivalent en termes de sécurité) afin de restituer la clé publique.

Les clés DKIM peuvent éventuellement être mémorisées dans plusieurs types de serveurs de clés et dans de multiples

formats. La mémorisation et le format des clés ne sont pas pertinents pour le reste de l'algorithme DKIM.

Les paramètres de l'algorithme de recherche de clé sont le type de la recherche (l'étiquette "q=") le domaine du signataire (l'étiquette "d=" du champ d'en-tête DKIM-Signature) et le sélecteur (l'étiquette "s=").

public_key = dkim_find_key(q_val, d_val, s_val)

Le présent document définit un seul lien, utilisant les RR DNS TXT pour distribuer les clés. D'autres liens pourront être définis à l'avenir.

3.6.1 Représentation textuelle

On s'attend à ce que de nombreux serveurs de clés choisissent de présenter les clés dans un format de texte non structuré par ailleurs (par exemple, une forme XML ne serait pas considérée comme étant du texte non structuré). La définition suivante DOIT être utilisée pour toute clé DKIM représentée dans une forme textuelle non structurée par ailleurs.

La syntaxe globale est une liste d'étiquettes comme décrit au paragraphe 3.2. Les étiquettes actuellement valides sont décrites ci-dessous. D'autres étiquettes PEUVENT être présentes et DOIVENT être ignorées par toute mise en œuvre qui ne les comprend pas.

v= Version de l'enregistrement de clé DKIM (plain-text; RECOMMANDÉ, "DKIM1" par défaut). Si elle est spécifiée, cette étiquette DOIT être réglée à "DKIM1" (sans les guillemets). Cette étiquette DOIT être la première dans l'enregistrement. Les enregistrements commençant par une étiquette "v=" avec une autre valeur DOIVENT être éliminés. Noter que les vérificateurs DOIVENT faire une comparaison de chaîne sur cette valeur ; par exemple, "DKIM1" n'est pas la même que "DKIM1.0".

ABNF :
key-v-tag = %x76 [FWS] "=" [FWS] %x44.4B.49.4D.31

h= : algorithmes de hachage acceptables (plain-text ; FACULTATIF, tous les algorithmes sont permis par défaut). Une liste des algorithmes de hachage qui peuvent être utilisés séparés par un caractère deux-points. Les algorithmes non reconnus DOIVENT être ignorés. Voir au paragraphe 3.3 une discussion des algorithmes de hachage mis en œuvre par les signataires et les vérificateurs. L'ensemble des algorithmes mentionnés dans cette étiquette dans chaque enregistrement est un choix opérationnel du signataire.

ABNF :
key-h-tag = %x68 [FWS] "=" [FWS] key-h-tag-alg *([FWS] ":" [FWS] key-h-tag-alg)
key-h-tag-alg = "sha1" / "sha256" / x-key-h-tag-alg
x-key-h-tag-alg = hyphenated-word ; pour une future extension

k= : type de clé (plain-text ; FACULTATIF, "rsa" par défaut). Les signataires et les vérificateurs DOIVENT prendre en charge le type de clé "rsa". Le type de clé "rsa" indique qu'une RSAPublicKey ASN.1 codée en DER [UIT-X660] (voir [RFC3447], paragraphes 3.1 et A.1.1) est utilisée dans l'étiquette "p=". (Note : l'étiquette "p=" code de plus la valeur en utilisant l'algorithme base64.) Les types de clé non reconnus DOIVENT être ignorés.

ABNF:
key-k-tag = %x76 [FWS] "=" [FWS] key-k-tag-type
key-k-tag-type = "rsa" / x-key-k-tag-type
x-key-k-tag-type = hyphenated-word ; pour future extension

n= : Notes qui pourraient intéresser un humain (qp-section ; FACULTATIF, est vide par défaut). Aucune interprétation n'est faite par aucun programme. Cette étiquette devrait être utilisée avec parcimonie dans tout mécanisme de serveur de clés qui a des limitations d'espace (notamment le DNS). Ceci est destiné aux administrateurs, pas aux utilisateurs finaux.

ABNF:
key-n-tag = %x6e [FWS] "=" [FWS] qp-section

p= : données de clé publique (base64 ; EXIGÉ). Une valeur vide signifie que cette clé publique a été révoquée. La syntaxe

et la sémantique de cette valeur d'étiquette avant d'être codée en base64 sont définies par l'étiquette "k=".

Raison pour information : si une clé privée a été compromise ou autrement désactivée (par exemple, un contrat d'externalisation s'est terminé) un signataire pourrait vouloir explicitement déclarer qu'il connaît le sélecteur, mais que tous les messages qui utilisent ce sélecteur devraient échouer à la vérification. Les vérificateurs DEVRAIENT retourner un code d'erreur pour tout champ d'en-tête DKIM-Signature avec un sélecteur qui fait référence à une clé révoquée. (Voir les détails au paragraphe 6.1.2.)

ABNF :
key-p-tag = %x70 [FWS] "=" [[FWS] base64string]

Note pour information : il est permis à une chaîne base64 d'inclure des espaces (FWS) à des positions arbitraires ; cependant, tous les CRLF doivent être suivis d'au moins un caractère WSP. Les mises en œuvre et les administrateurs devraient s'assurer que les RR TXT de sélecteur se conforment à cette spécification.

s= : type de service (plain-text ; FACULTATIF; "*" par défaut). Liste séparée par un caractère deux-points des types de service auxquels cet enregistrement s'applique. Les vérificateurs d'un type de service donné DOIVENT ignorer cet enregistrement si le type approprié ne figure pas sur la liste. Les types de service non reconnus DOIVENT être ignorés. Les types de service actuellement définis sont les suivants :

* : correspond à tous les types de service

email : messagerie électronique (pas nécessairement limité à SMTP)

Cette étiquette est destinée à contraindre les utilisations de clés pour d'autres besoins, si des utilisations de DKIM devaient être définies par d'autres services à l'avenir.

ABNF :
key-s-tag = %x73 [FWS] "=" [FWS] key-s-tag-type *([FWS] ":" [FWS] key-s-tag-type)
key-s-tag-type = "email" / "*" / x-key-s-tag-type
x-key-s-tag-type = hyphenated-word ; pour future extension

t= : fanions, représentés comme une liste de noms séparés par des caractères deux-points (plain-text ; FACULTATIF, par défaut aucun fanion n'est établi). Les fanions non reconnus DOIVENT être ignorés. Les fanions définis sont :

y : ce domaine est d'essai de DKIM. Les vérificateurs NE DOIVENT PAS traiter les messages des signataires en mode d'essai différemment de la messagerie non signée, même si la signature échoue à la vérification. Les vérificateurs PEUVENT souhaiter tracer les résultats du mode d'essai pour assister le signataire.

s : tous les champs d'en-tête DKIM-Signature qui utilisent l'étiquette "i=" DOIVENT avoir la même valeur de domaine sur le côté droit du "@" dans l'étiquette "i=" et la valeur de l'étiquette "d=". C'est-à-dire, le domaine "i=" NE DOIT PAS être un sous domaine de "d=". L'utilisation de ce fanion est RECOMMANDÉE sauf si des sous domaines sont exigés.

ABNF :
key-t-tag = %x74 [FWS] "=" [FWS] key-t-tag-flag *([FWS] ":" [FWS] key-t-tag-flag)
key-t-tag-flag = "y" / "s" / x-key-t-tag-flag
x-key-t-tag-flag = hyphenated-word ; pour future extension

3.6.2 Liens DNS

On défini ici un lien utilisant les RR DNS TXT comme service de clés. Toutes les mises en œuvre DOIVENT prendre en charge ce lien.

3.6.2.1 Espace de noms

Toute les clés DKIM sont mémorisées dans un sous domaine appelé "_domainkey". Dans un champ DKIM-Signature avec une étiquette "d=" de "exemple.com" et une étiquette "s=" de "foo.bar", l'interrogation DNS va être pour "foo.bar._domainkey.exemple.com".

3.6.2.2 Types d'enregistrement de ressource pour la mémorisation de clés

Le type d'enregistrement de ressource DNS utilisé est spécifié par une option à l'étiquette query-type ("q="). La seule option définie dans cette spécification de base est "txt", qui indique l'utilisation d'un RR TXT. Une extension ultérieure de cette norme pourrait définir un autre type de RR.

Les chaînes dans un RR TXT DOIVENT être enchaînées ensemble avant utilisation sans espace entre. Les RR TXT DOIVENT être uniques pour un nom de sélecteur particulier ; c'est-à-dire, si il y a plusieurs enregistrements dans un RRset, les résultats sont indéfinis.

Les RR TXT sont codés comme décrit au paragraphe 3.6.1.

3.7 Calcul des hachages de message

Les signatures de message de signature et de vérification commencent toutes deux par une étape de calcul de deux hachages cryptographiques sur le message. Les signataires vont choisir les paramètres de la signature comme décrit dans "Actions du signataire" (Section 5) ; les vérificateurs vont utiliser les paramètres spécifiés dans le champ d'en-tête DKIM-Signature qui est vérifié. Dans la discussion qui suit, les noms des étiquettes dans le champ d'en-tête DKIM-Signature qui existent (quand c'est une vérification) ou vont être créées (quand c'est une signature) sont utilisés. Noter que la canonisation (paragraphe 3.4) est seulement utilisée pour préparer le message pour la signature ou la vérification ; elle n'affecte en aucune façon le message transmis.

Le signataire/vérificateur DOIT calculer deux hachages : un sur le corps du message et un sur les champs d'en-tête choisis du message.

Les signataires DOIVENT les calculer dans l'ordre montré. Les vérificateurs PEUVENT les calculer dans tout ordre qui convient au vérificateur, pourvu que le résultat soit sémantiquement identique à la sémantique qui aurait résulté si ils avaient été calculés dans cet ordre.

Dans l'étape 1 du hachage, le signataire/vérificateur DOIT hacher le corps de message, canonisé en utilisant l'algorithme de canonisation de corps spécifié dans l'étiquette "c=" et ensuite tronqué à la longueur spécifiée dans l'étiquette "l=". Cette valeur de hachage est ensuite convertie en forme base64 et insérée dans (signataires) ou comparée à (vérificateurs) l'étiquette "bh=" du champ d'en-tête DKIM-Signature.

Dans l'étape 2 du hachage, le signataire/vérificateur DOIT passer ce qui suit à l'algorithme de hachage dans l'ordre indiqué.

1. Les champs d'en-tête spécifiés par l'étiquette "h=", dans l'ordre spécifié dans cette étiquette, et canonisés en utilisant l'algorithme de canonisation d'en-tête spécifié dans l'étiquette "c=". Chaque champ d'en-tête DOIT être terminé par un seul CRLF.

2. Le champ d'en-tête DKIM-Signature qui existe (vérification) ou va être inséré (signature) dans le message, avec la valeur de l'étiquette "b=" (incluant toutes les espaces environnantes) supprimée (c'est-à-dire, traitée comme la chaîne vide) canonisée en utilisant l'algorithme de canonisation d'en-tête spécifié dans l'étiquette "c=", et sans CRLF en queue.

Toutes les étiquettes et leur valeurs dans le champ d'en-tête DKIM-Signature sont incluses dans le hachage cryptographique à la seule exception de la portion valeur de l'étiquette "b=" (signature) qui DOIT être traitée comme la chaîne nulle. Toutes les étiquettes DOIVENT être incluses même si elles pourraient ne pas être comprises par le vérificateur. Le champ d'en-tête DOIT être présenté à l'algorithme de hachage après le corps du message plutôt que avec le reste des champs d'en-tête et DOIT être canonisé comme spécifié dans l'étiquette "c=" (canonisation). Le champ d'en-tête DKIM-Signature NE DOIT PAS être inclus dans sa propre étiquette "h=", bien que d'autres champs d'en-tête DKIM-Signature PUISSENT être signés (voir la Section 4).

Lors du calcul du hachage sur les messages qui vont être transmis en utilisant le codage base64 ou quoted-printable, les signataires DOIVENT calculer le hachage après le codage. De même, le vérificateur DOIT incorporer les valeurs dans le hachage avant de décoder le texte en base64 ou quoted-printable. Cependant, le hachage DOIT être calculé avant les codages de niveau transport, comme le "bourrage de points" de SMTP (la modification des lignes qui commencent par un "." pour éviter de confondre avec le marqueur de fin de message de SMTP, comme spécifié dans la [RFC5321]).

À l'exception de la procédure de canonisation décrite au paragraphe 3.4, le processus de signature DKIM traite le corps des messages comme une simple chaîne d'octets. Les messages DKIM PEUVENT être soit en format plain-text, soit en format MIME ; aucun traitement particulier n'est accordé au contenu MIME. Les pièces jointes en format MIME DOIVENT être incluses dans le contenu qui est signé.

Plus formellement, le pseudo-code pour l'algorithme de signature est :

body-hash = hash-alg (canon-body, l-param)
data-hash = hash-alg (h-headers, D-SIG, body-hash)
signature = sig-alg (d-domain, selector, data-hash)

où :

body-hash : est le résultat du hachage du corps, en utilisant hash-alg.

hash-alg : est l'algorithme de hachage spécifié dans le paramètre "a".

canon-body : est une représentation canonisée du corps, produite en utilisant l'algorithme de corps spécifié dans le paramètre "c", comme défini au paragraphe 3.4 et en excluant le champ DKIM-Signature.

l-param : est la valeur de longueur de corps du paramètre "l".

data-hash : est le résultat de l'utilisation de l'algorithme hash-alg, pour hacher l'en-tête incluant l'en-tête DKIM-Signature, et le hachage de corps.

h-headers : est la liste des en-têtes à signer, comme spécifié dans le paramètre "h".

D-SIG : est le champ DKIM-Signature canonisé lui-même sans la portion valeur de signature du paramètre, c'est-à-dire, une valeur de paramètre vide.

signature : est la valeur de signature produite par l'algorithme de signature.

sig-alg : est l'algorithme de signature spécifié par le paramètre "a".

d-domain : est le nom de domaine spécifié dans le paramètre "d".

selector : est la valeur de sélecteur spécifiée dans le paramètre "s".

Note : de nombreuses API de signature numérique fournissent le hachage et l'application de la clé privée RSA en utilisant une seule primitive "sign()". Quand on utilise une telle API, les deux dernières étapes de l'algorithme vont probablement être combinées en une seule invocation qui va effectuer les deux "a-hash-alg" et "sig-alg".

3.8 Exigences d'entrée

Un message qui n'est pas conforme aux [RFC5322], [RFC2045], et [RFC2047] peut être l'objet de tentatives par des intermédiaires de corriger ou interpréter ce contenu. Voir à la Section 8 de la [RFC4409] des exemples de changements courants. De telles "corrections" peuvent invalider les signatures DKIM ou avoir d'autres effets indésirables, y compris d'impliquer des changements à la façon dont un message est présenté à un utilisateur final.

En conséquence, la conception de DKIM est prévue sur une entrée valide. Donc, les signataires et les vérificateurs DEVRAIENT prendre des mesures raisonnables pour s'assurer que les messages qu'ils traitent sont valides selon les [RFC5322], [RFC2045], et toutes autres normes pertinentes de format de message.

Voir au paragraphe 8.15 des précisions supplémentaires.

3.9 Exigences de sortie

L'évaluation de chaque signature se termine dans un de ces trois états, que le présent document appelle comme suit :

SUCCESS : une vérification réussie.

PERMFAIL : une erreur permanente, non récupérable comme un échec de vérification de signature.

TEMPFAIL : une erreur temporaire, récupérable comme une fin de temporisation d'interrogation du DNS.

Pour chaque signature qui se vérifie avec succès ou produit un résultat TEMPFAIL, le résultat de l'algorithme DKIM DOIT inclure l'ensemble de :
- o nom de domaine, pris de l'étiquette de signature "d=" ; et
- o résultat de la tentative de vérification pour cette signature.

Le résultat PEUT inclure d'autres propriétés de signature ou méta-données de résultat, incluant des signatures en échec permanent ou autrement ignorées, pour l'usage des modules qui consomment ces résultats.

Voir au paragraphe 6.1 la discussion des codes de résultats de validation de signature.

3.10 Signatures par domaines parents

Dans certaines circonstances, il est désirable qu'un domaine applique une signature au nom de tous ses sous domaines sans avoir besoin de tenir des sélecteurs séparés (enregistrements de clé) dans chaque sous domaine. Par défaut, des clés privées correspondant aux enregistrements de clé peuvent être utilisées pour signer les messages pour tout sous domaine du domaine dans lequel ils résident ; par exemple, un enregistrement de clé pour le domaine exemple.com peut être utilisé pour vérifier des messages où le AUID (étiquette "i=" de la signature) est sub.exemple.com, ou même sub1.sub2.exemple.com. Afin de limiter la capacité de telles clés quand ceci n'est pas voulu, le fanion "s" PEUT être établi dans l'étiquette "t=" de l'enregistrement de clé, pour contraindre la validité du domaine de l'AUID. Si l'enregistrement de clé référencé contient le fanion "s" au titre de l'étiquette "t=", le domaine de l'AUID (fanion "i=") DOIT être le même que celui du domaine du SDID (d=). Si ce fanion est absent, le domaine de l'AUID DOIT être le même, ou un sous domaine, que celui du SDID.

3.11 Relations entre SDID et AUID

La principale tâche de DKIM est de communiquer du signataire à un témoin d'identité côté receveur un seul identifiant de domaine signant (SDID, *Signing Domain Identifier*) qui se réfère à une identité responsable. DKIM PEUT facultativement fournir un seul agent ou identifiant d'utilisateur (AUID, *Agent or User Identifier*) responsable.

Donc, le résultat obligatoire de DKIM à un témoin d'identité côté réception est un seul nom de domaine. Dans la portée de son utilisation comme résultat DKIM, le nom a seulement la sémantique de base de nom de domaine ; toute sémantique spécifique du propriétaire possible sort du domaine d'application de DKIM. C'est-à-dire, dans son rôle comme identifiant DKIM, une sémantique supplémentaire ne peut pas être supposée par un témoin d'identité.

La signature étant bien vérifiée, un vérificateur DKIM côté receveur DOIT communiquer l'identifiant de domaine signant (d=) à un module témoin d'identité consommateur et PEUT communiquer l'identifiant d'agent ou d'utilisateur (i=) si il est présent.

Dans la mesure où un receveur tente d'imaginer intuitivement une sémantique structurée pour un des identifiants, cela est une fonction heuristique qui sort du domaine d'application de la spécification et sémantique de DKIM. Donc, ceci est de la compétence d'un service de niveau supérieur comme un filtre de traitement de livraison qui intègre diverses entrées et en effectue l'analyse heuristique.

Discussion pour information : le présent document n'exige pas que la valeur du SDID ou AUID corresponde à un identifiant d'un autre champ d'en-tête du message. Cette exigence est plutôt un problème d'une politique de témoin. L'objet d'un tel lien serait d'authentifier la valeur dans cet autre champ d'en-tête. Ceci est à son tour la base de l'application d'une affirmation de confiance fondée sur la valeur de l'identifiant. La confiance est un sujet vaste et complexe, et les mécanismes de la confiance sont l'objet d'attaques très créatives. L'efficacité réelle de tous les liens sauf les plus basiques entre le SDID ou AUID et les autres identités n'est pas bien établie, ni sa vulnérabilité à la

subversion par un attaquant. Donc, s'appuyer sur l'utilisation de tels liens devrait être strictement limité. En particulier, la mesure dans laquelle un utilisateur final receveur peut s'appuyer sur des assurances qui pourraient être données par l'utilisation réussie du SDID ou AUID n'est pas du tout claire.

4. Sémantique des signatures multiples

4.1 Exemples de scénarios

Il y a de nombreuses raisons pour qu'un message ait plusieurs signatures. Par exemple, supposons que SHA-256 soit à l'avenir trouvé être d'une force insuffisante, et que l'usage de DKIM passe à SHA-1024. Un signataire pourrait immédiatement signer en utilisant le nouvel algorithme mais aussi continuer de signer en utilisant l'ancien algorithme pour l'interopérabilité avec les vérificateurs qui ne sont pas encore mis à niveau. Le signataire ferait cela en ajoutant deux champs d'en-tête DKIM-Signature, un en utilisant chaque algorithme. Les vérificateurs plus anciens qui ne reconnaissent pas SHA-1024 comme algorithme acceptable vont sauter cette signature et utiliser l'ancien algorithme ; les vérificateurs plus récents pourraient utiliser l'une ou l'autre signature à leur choix et, toutes choses égales par ailleurs, pourraient même ne pas tenter de vérifier l'autre signature.

De même, un signataire pourrait signer un message incluant tous les champs d'en-tête et pas d'étiquette "l=" (pour satisfaire les vérificateurs stricts) et une seconde fois avec un ensemble limité de champs d'en-tête et une étiquette "l=" (anticipant uns possible modification de message en chemin de la part des autres vérificateurs). Les vérificateurs pourraient alors choisir quelle signature ils préfèrent.

Bien sûr, un message pourrait aussi avoir plusieurs signatures parce qu'il passe à travers plusieurs signataires. Un cas courant est supposé être celui d'un message signé qui passe par une liste de diffusion qui signe aussi tous les messages. En supposant que ces deux signatures se vérifient, un receveur pourrait choisir d'accepter le message si l'une de ces signatures est connue pour venir de sources de confiance.

En particulier, les receveurs pourraient choisir de faire une liste des listes de diffusion auxquelles ils se sont abonnés et qui ont des politiques anti-abus acceptables afin d'accepter les messages envoyés à cette liste même provenant d'auteurs inconnus. Ils pourraient aussi s'abonner à des listes de diffusion qui sont moins de confiance (par exemple, sans protection anti-abus) et vouloir accepter tous les messages provenant d'auteurs spécifiques mais insister pour faire un examen supplémentaire d'abus sur les autres messages.

Un autre exemple en rapport de plusieurs signataires pourrait être des services de transmission, comme ceux couramment associés aux sites académiques d'anciens élèves de collège. Par exemple, un receveur pourrait avoir une adresse à members.exemple.org, un site qui a une protection anti-abus qui est un peu moins efficace que ce que le receveur préférerait. Un tel receveur pourrait avoir des auteurs spécifiques dont les messages seraient absolument de confiance, mais des messages provenant d'auteurs inconnus qui ont réussi l'examen de passage de transmission auraient seulement une confiance moyenne.

4.2 Interprétation

Un signataire qui ajoute une signature à un message crée simplement un nouvel en-tête DKIM-Signature, en utilisant la sémantique usuelle de l'option "h=". Un signataire PEUT signer des champs d'en-tête DKIM-Signature existant précédemment en utilisant la méthode décrite au paragraphe 5.4 pour signer des champs d'en-tête de trace.

Noter que les signataires devraient être conscients que signer des champs d'en-tête DKIM-Signature peut résulter en des échecs de signature avec des intermédiaires qui ne reconnaissent pas que les champs d'en-tête DKIM-Signature sont des champs d'en-tête de trace et les réordonnent involontairement, cassant donc de telles signatures. Pour cette raison, signer des champs d'en-tête DKIM-Signature existants est mal avisé, bien que légal.

Note pour information : si un champ d'en-tête avec plusieurs instances est signé, ces champs d'en-tête sont toujours signés de bas en haut. Donc, il n'est pas possible de signer seulement des champs d'en-tête DKIM-Signature spécifiques. Par exemple, si le message signé contient déjà trois champs d'en-tête DKIM-Signature A, B, et C, il est possible de les signer tous, seulement B et C, ou C seulement, mais pas A seulement, B seulement, A et B seulement, ou A et C seulement.

Un signataire PEUT ajouter plus d'un champ d'en-tête DKIM-Signature en utilisant des paramètres différents. Par exemple,

durant une période de transition, un signataire pourrait vouloir produire des signatures en utilisant deux algorithmes de hachage différents.

Les signataires NE DEVRAIENT PAS supprimer de champs d'en-tête DKIM-Signature des messages qu'ils signent, même si ils savent que les signatures ne peuvent pas être vérifiées.

Quand il évalue un message avec plusieurs signatures, un vérificateur DEVRAIT évaluer les signatures indépendamment et sur leurs propres mérites. Par exemple, un vérificateur qui par sa politique choisit de ne pas accepter des signatures avec des algorithmes de chiffrement déconseillés va considérer ces signatures comme invalides. Les vérificateurs PEUVENT traiter les signatures dans l'ordre de leur choix ; par exemple, certains vérificateurs pourraient choisir de traiter les signatures correspondant au champ From dans l'en-tête de message avant les autres signatures. Voir au paragraphe 6.1 plus d'informations sur les choix de signatures.

Note de mise en œuvre pour information : le vérificateur qui tente de corréler les signatures valides avec des signatures invalides en tentant de deviner pourquoi une signature a échoué sont mal avisés. En particulier, il n'y a pas de moyen général par lequel un vérificateur puisse déterminer qu'une signature invalide a jamais été valide.

Les vérificateurs DEVRAIENT continuer de vérifier les signatures jusqu'à ce qu'une signature se vérifie à la satisfaction du vérificateur. Pour limiter les potentielles attaques de déni de service, les vérificateurs PEUVENT limiter le nombre total de signatures qu'ils vont tenter de vérifier.

Si un module de vérificateur rapporte des signatures dont l'évaluation produit des résultats PERMFAIL, les témoins d'identité DEVRAIENT ignorer ces signatures (voir au paragraphe 6.1) agissant comme si elles n'étaient pas présentes dans le message.

5. Actions de signataire

Les étapes suivantes sont effectuées dans l'ordre par les signataires.

5.1 Déterminer si le message devrait être signé et par qui

Un signataire ne peut évidemment signer un message que pour les domaines pour lesquels il a une clé privée et la connaissance nécessaire de la clé publique et des informations de sélecteur correspondantes. Cependant, il y a un certain nombre d'autres raisons au delà de l'absence d'une clé privée pour lesquelles un signataire pourrait choisir de ne pas signer un massage.

Note pour information : un signataire peut être mis en œuvre au titre de toute portion du système de messagerie comme réputé approprié, incluant un MUA, un serveur de soumission, ou un MTA. Partout où il est mis en œuvre, le signataire devrait faire attention lorsque il signe (et affirmant par là sa responsabilité) des messages qui peuvent être problématiques. En particulier, dans une enclave de confiance, le domaine signant pourrait être dérivé de l'en-tête conformément à la politique locale ; les serveurs de soumission pourraient seulement signer les messages provenant d'utilisateurs qui sont authentifiés et autorisés de façon appropriée.

Avis de mise en œuvre pour information : les serveurs de soumission ne devraient pas signer les champs d'en-tête Received si la passerelle sortante de MTA cache les champs d'en-tête Received, par exemple, pour cacher les détails de la topologie interne.

Si un message ne peut pas être signé pour une raison quelconque, c'est une décision de politique locale qui dicte que faire du message.

5.2 Choix d'une clé privée et informations de sélecteur correspondantes

La présente spécification ne définit pas les bases sur lesquelles un signataire devrait choisir quelle clé privée et informations de sélecteur utiliser. Actuellement, tous les sélecteurs sont égaux pour ce qui concerne cette spécification, donc la décision devrait largement être une affaire administrative. La distribution et la gestion des clés privées sort aussi du domaine d'application du présent document.

Avis de fonctionnement pour information : un signataire ne devrait pas signer avec une clé privée quand le sélecteur qui contient la clé publique correspondante est supposé être révoqué ou supprimé avant que le vérificateur ait l'opportunité de valider la signature. Le signataire devrait anticiper que les vérificateurs peuvent choisir de différer la validation, peut-être jusqu'à ce que le message soit lu par le receveur final. En particulier, quand il y a une rotation à une nouvelle paire de clés, la signature devrait immédiatement commencer avec la nouvelle clé privée, et l'ancienne clé publique devrait être conservée pendant un intervalle de validation raisonnable avant d'être supprimée du serveur de clés.

5.3 Normaliser le message pour empêcher les conversions de transport

Certains messages, en particulier ceux qui utilisent des caractères de 8 bits, sont sujets à modification durant le transit, notamment la conversion en forme à 7 bits. De telles conversions cassent les signatures DKIM. Afin de minimiser la probabilité d'une telle cassure, les signataires DEVRAIENT convertir le message en un codage de transfert de contenu MIME convenable comme du quoted-printable ou base64 comme décrit dans la [RFC2045] avant de signer. Une telle conversion sort du domaine d'application de DKIM ; le message réel DEVRAIT être converti en MIME à 7 bits par un MUA ou MSA avant la présentation à l'algorithme DKIM.

Si le message est soumis au signataire avec un codage local qui va être modifié avant transmission, cette modification à la forme canonique de la [RFC5322] DOIT être faite avant de signer. En particulier, les caractères CR ou LF nus (utilisés par certains systèmes comme convention locale de séparateur de ligne) DOIVENT être convertis en la séquence standard SMTP de CRLF avant que le message soit signé. Toute conversion de cette sorte DEVRAIT être appliquée au message réellement envoyé aux receveurs, pas juste à la version présentée à l'algorithme de signature.

De façon plus générale, le signataire DOIT signer le message comme il est supposé être reçu par le vérificateur plutôt que dans une forme locale ou interne.

5.3.1 Limites de longueur de corps

Un compte de longueur de corps PEUT être spécifié pour limiter le calcul de signature à un préfixe initial du texte de corps, mesuré en octets. Si le compte de longueur de corps n'est pas spécifié, le corps de message entier est signé.

Raison pour information : cette capacité est fournie parce que il est très courant que les listes de diffusion ajoutent des en-queues aux messages (par exemple, des instructions sur comment quitter la liste). Jusqu'à ce que ces messages soient aussi signés, le compte de longueur de corps est un outil utile pour le vérificateur car il peut, selon sa politique, accepter des messages qui ont des signatures valides avec des données étrangères.

La longueur réellement hachée devrait être insérée dans l'étiquette "l=" du champ d'en-tête DKIM-Signature. (Voir au paragraphe 3.5.)

Le compte de longueur de corps permet au signataire d'un message d'autoriser l'ajout de données à la fin du corps d'un message signé. Le compte de longueur de corps DOIT être calculé à la suite de l'algorithme de canonisation ; par exemple, les espaces ignorées par l'algorithme de canonisation ne sont pas incluses au titre du compte de longueur de corps.

Un compte de longueur de corps de zéro signifie que le corps est complètement non signé.

Les signataires qui souhaitent s'assurer qu'aucune modification de quelque sorte ne se produit devraient spécifier l'algorithme de canonisation "simple" pour l'en-tête et pour le corps et omettre le compte de longueur de corps.

Voir au paragraphe 8.2 pour les détails.

5.4 Déterminer les champs d'en-tête à signer

Le champ d'en-tête From DOIT être signé (c'est-à-dire, inclus dans l'étiquette "h=" du champ d'en-tête DKIM-Signature résultant). Les signataires NE DEVRAIENT PAS signer un champ d'en-tête existant qui a des chances d'être légitimement modifié ou supprimé en transit. En particulier, la [RFC5321] permet explicitement la modification ou la suppression du champ d'en-tête Return-Path dans le transit. Les signataires PEUVENT inclure tout autre champ d'en-tête présent au moment de la signature à la discrétion du signataire.

Note de fonctionnement pour information : le choix des champs d'en-tête à signer n'est pas évident. Une stratégie est de signer tous les champs d'en-tête existants non répétables. Une autre stratégie est de signer seulement les champs d'en-tête qui seront probablement affichés ou vont probablement affecter le traitement du message chez le receveur. Une troisième stratégie est de signer seulement les en-têtes "bien connus". Noter que les vérificateurs peuvent traiter les champs d'en-tête non signés avec un extrême scepticisme, incluant de refuser de les afficher à l'utilisateur final ou même d'ignorer la signature si elle ne couvre pas certains champs d'en-tête. Pour cette raison, signer les champs présents dans le message comme Date, Subject, Reply-To, Sender, et tous les champs d'en-tête MIME est fortement conseillé.

Le champ d'en-tête DKIM-Signature est toujours implicitement signé et NE DOIT PAS être inclus dans l'étiquette "h=" sauf pour indiquer que d'autres signatures pré-existantes sont aussi signées.

Les signataires PEUVENT prétendre avoir des champs d'en-tête signés qui n'existent pas (c'est-à-dire, les signataires PEUVENT inclure le nom du champ d'en-tête dans l'étiquette "h=" même si ce champ d'en-tête n'existe pas dans le message). Lors du calcul de la signature, le champ d'en-tête non existant DOIT être traité comme la chaîne nulle (incluant le nom de champ d'en-tête, la valeur du champ d'en-tête, toute la ponctuation, et le CRLF en queue).

Raison pour information : cela permet aux signataires d'affirmer explicitement l'absence d'un champ d'en-tête ; si ce champ d'en-tête est ajouté plus tard, la signature va échouer.

Note pour information : un nom de champ d'en-tête a seulement besoin d'être mentionné une fois de plus que le nombre réel de champs d'en-tête dans un message au moment de la signature afin d'empêcher d'autres ajouts. Par exemple, si il y a un seul champ d'en-tête Comments au moment de la signature, mentionner Comments deux fois dans l'étiquette "h=" est suffisant pour empêcher qu'un autre champ d'en-tête Comments soit ajouté ; il n'est pas nécessaire (mais est légal) de mentionner trois fois Comments ou plus dans l'étiquette "h=".

Voir au paragraphe 5.4.2 la discussion de la procédure à suivre lors de la canonisation d'un en-tête avec plus d'une instance d'un nom de champ d'en-tête particulier.

Les signataires doivent être prudents en signant des champs d'en-tête qui pourraient avoir des instances supplémentaires ajoutées plus tard dans le processus de livraison, car de tels champs d'en-tête pourraient être insérés après l'instance signée ou être autrement réordonnés. Les champs d'en-tête de trace (comme Received) et les blocs Resent-* sont les seuls champs dont la [RFC5322] interdit de les réordonner. En particulier, comme les champs d'en-tête DKIM-Signature peuvent être réordonnés par des MTA intermédiaires, signer les champs d'en-tête DKIM-Signature existants est enclin à l'erreur.

Conseil pour information : en dépit du fait que la [RFC5322] n'interdit pas le réarrangement des champs d'en-tête, réordonner les champs d'en-tête signés avec plusieurs instances par les MTA intermédiaires va causer la cassure des signatures DKIM ; un tel comportement antisocial devrait être évité.

Note de mise en œuvre pour information : bien que ce ne soit pas exigé par la présente spécification, tous les champs d'en-tête visibles par l'utilisateur final devraient être signés pour éviter un possible "envoi de pourriels indirect". Par exemple, si le champ d'en-tête Subject n'est pas signé, un envoyeur de pourriels peut renvoyer un message précédemment signé, en remplaçant le sujet légitime par un pourriel d'une ligne.

5.4.1 Contenu de signature recommandé

L'objet de l'algorithme de chiffrement DKIM est de fixer un identifiant au message d'une façon à la fois robuste contre les changements normaux relatifs au transit et résistante aux différentes sortes d'attaques en répétition. Un aspect essentiel de la satisfaction de ces exigences est de choisir quels champs d'en-tête inclure dans le hachage et quels champs en exclure.

La règle de base pour choisir des champs à inclure est de choisir les champs qui constituent le "cœur" du contenu du message. Donc, toute attaque en répétition va devoir les inclure afin que la signature réussisse ; cependant, avec ces champs inclus, le cœur du message est valide, même si il est envoyé à de nouveaux receveurs.

Des exemples courants de champs avec des adresses et des champs de contenu textuel relatif au corps sont :
- From (EXIGÉ ; voir le paragraphe 5.4)
- Reply-To
- Subject
- Date

- To, Cc
- Resent-Date, Resent-From, Resent-To, Resent-Cc
- In-Reply-To, References
- List-Id, List-Help, List-Unsubscribe, List-Subscribe, List-Post, List-Owner, List-Archive

Si l'étiquette de signature "l=" est utilisée (voir le paragraphe 3.5) le champ Content-Type est aussi candidat l'inclusion car il pourrait être remplacé d'une façon qui cause qu'un contenu complètement différent soit rendu à l'utilisateur receveur.

Il y a des compromis dans la décision de ce qui constitue le "cœur" du message, qui pour certains champs est un concept subjectif. Inclure des champs comme "Message-ID", par exemple, est utile si on considère un mécanisme pour être capable de distinguer des instances séparées du même message comme étant du cœur du contenu. De même, il pourrait être désirable d'inclure "In-Reply-To" et "References" si on considère le suivi du message comme une partie de cœur du message.

Une autre classe de champs qui peut être intéressante est celle qui porte des informations relatives à la sécurité sur le message, comme Authentication-Results [RFC5451].

La règle de base pour choisir les champs à exclure est de choisir les champs pour lesquels il y a plusieurs champs avec le même nom et les champs qui sont modifiés dans le transit. Des exemples en sont :
- Return-Path
- Received
- Comments, Keywords

Noter que le champ DKIM-Signature est aussi exclu du hachage de l'en-tête parce que son traitement est spécifié séparément.

Normalement, il est préférable d'exclure d'autres champs facultatifs à cause de l'éventualité que des champs supplémentaires du même nom soient légitimement ajoutés ou réordonnés avant la vérification. Il va probablement y avoir des exceptions légitimes à cette règle à cause de la grande variété de champs d'en-tête spécifiques d'application qui pourraient être appliqués à un message, dont certains ont peu de chances d'être dupliqués, modifiés, ou réordonnés.

Les signataires DEVRAIENT choisir des algorithmes de canonisation sur la base des types de messages qu'ils traitent et de leur aversion au risque. Par exemple, les sites de commerce électronique qui envoient principalement des reçus d'achat, qui ne sont pas supposés être traités par des listes de diffusion ou d'autres logiciels qui modifient les messages, vont généralement préférer la canonisation"simple".

Les sites qui envoient principalement des messages de personne à personne vont probablement préférer être plus résilients à la modification durant le transport en utilisant la canonisation "lâche".

Sauf si le message est traité par des intermédiaires, comme des listes de diffusion qui pourraient ajouter des instructions "unsubscribe" au bas du corps de message, l'étiquette "l=" va probablement n'apporter pas d'avantage supplémentaire tout un donnant un moyen d'ajouts non autorisés au texte d'un message. L'utilisation de "l=0" pousse cela à l'extrême, en permettant une altération complète du texte du message sans invalider la signature. De plus, un vérificateur serait en droit de considérer un corps de message partiellement signé comme inacceptable. Une utilisation judicieuse est conseillée.

5.4.2 Signatures impliquant plusieurs instances d'un champ

Les signataires qui choisissent de signer un champ d'en-tête existant qui se produit plus d'une fois dans le message (comme un Received) DOIVENT signer la dernière instance physique de ce champ d'en-tête dans le bloc d'en-têtes. Les signataires qui souhaitent signer plusieurs instances d'un tel champ d'en-tête DOIVENT inclure le nom du champ d'en-tête plusieurs fois dans l'étiquette "h=" du champ d'en-tête DKIM-Signature et DOIVENT signer ces champs d'en-tête dans l'ordre du bas du bloc de champs d'en-tête jusqu'au sommet. Le signataire PEUT inclure plus d'instances d'un nom de champ d'en-tête dans "h=" qu'il n'y a réellement de champs d'en-tête correspondants afin que la signature ne se vérifie pas si des champs d'en-tête supplémentaires de ce nom sont ajoutés.

Exemple pour information :

Si le signataire souhaite signer deux champs d'en-tête Received existants , et si l'en-tête existant contient :
Received: <A>

Received: <B>
Received: <C>

alors le champ d'en-tête DKIM-Signature résultant devrait se lire :

DKIM-Signature: ... h=Received : Received :...

et les champs d'en-tête Received <C> et <B> vont être signés dans cet ordre.

5.5 Calcul du hachage et de la signature de message

Le signataire DOIT calculer le hachage de message comme décrit au paragraphe 3.7 et ensuite le signer en utilisant l'algorithme de clé publique choisi. Il va en résulter un champ d'en-tête DKIM-Signature qui va inclure le hachage de corps et une signature du hachage de l'en-tête, où cet en-tête inclut le champ d'en-tête DKIM-Signature lui-même.

Des entités comme les gestionnaires de liste de diffusion qui mettent en œuvre DKIM et qui modifient le message ou un champ d'en-tête (par exemple, insèrent des informations de désabonnement) avant de retransmettre le message DEVRAIENT vérifier toute signature existante en entrée et DOIVENT faire de telles modifications avant de resigner le message.

5.6 Insertion du champ d'en-tête DKIM-Signature

Finalement, le signataire DOIT insérer le champ d'en-tête DKIM-Signature créé dans l'étape précédente avant de transmettre le message. Le champ d'en-tête DKIM-Signature DOIT être le même qu'utilisé pour calculer le hachage comme décrit ci-dessus, sauf que la valeur de l'étiquette "b=" DOIT être le hachage signé approprié calculé à l'étape précédente, signé en utilisant l'algorithme spécifié dans l'étiquette "a=" du champ d'en-tête DKIM-Signature et en utilisant la clé privée correspondant au sélecteur donné dans l'étiquette "s=" du champ d'en-tête DKIM-Signature, choisie au paragraphe 5.2.

Le champ d'en-tête DKIM-Signature DOIT être inséré avant tout autre champ DKIM-Signature dans le bloc d'en-têtes.

Note de mise en œuvre pour information : la façon la plus facile de réaliser cela est d'insérer le champ d'en-tête DKIM-Signature au début du bloc d'en-tête. En particulier, il peut être placé avant tout champ d'en-tête Received existant. Ceci est cohérent avec le traitement de DKIM-Signature comme champ d'en-tête de trace.

6. Actions du vérificateur

Comme un signataire PEUT supprimer ou révoquer une clé publique à tout moment, il est conseillé que la vérification se produise en temps utile. Dans de nombreuses configurations, le moment utile est durant l'acceptation par le MTA de bordure ou peu après. En particulier, différer la vérification jusqu'à ce que le message soit accessible à l'utilisateur final est déconseillé.

Un MTA de bordure ou intermédiaire PEUT vérifier la ou les signatures de message. Un MTA qui a effectué la vérification PEUT communiquer le résultat de cette vérification en ajoutant un champ d'en-tête Vérification aux messages entrants. Cela simplifie considérablement les choses pour l'utilisateur, qui peut maintenant utiliser un agent existant d'utilisateur de messagerie. La plupart des MUA ont la capacité de filtrer les messages sur la base des champs d'en-tête ou du contenu de message ; ces filtres vont être utilisés pour mettre en œuvre toute politique que l'utilisateur souhaite par rapport aux messages non signés.

Un MTA vérificateur PEUT mettre en œuvre une politique à l'égard des messages invérifiables, sans considération de si il applique ou non la vérification de champ d'en-tête aux messages signés.

Les vérificateurs DOIVENT produire un résultat sémantiquement équivalent à l'application des étapes mentionnées aux paragraphes 6.1, 6.1.1, et 6.1.2 dans l'ordre. En pratique, plusieurs de ces étapes peuvent être effectuées en parallèle afin d'améliorer les performances.

6.1 Extraction des signatures du message

L'ordre dans lequel les vérificateurs essayent les champs d'en-tête DKIM-Signature n'est pas défini ; les vérificateurs PEUVENT essayer les signatures dans l'ordre qu'ils veulent. Par exemple, une mise en œuvre pourrait essayer les signatures dans l'ordre textuel, tandis que d'autres pourraient essayer les signatures par identités qui correspondent aux contenus du champ d'en-tête From avant d'essayer d'autres signatures. Les vérificateurs NE DOIVENT PAS attribuer une signification définitive à l'ordre de multiples champs d'en-tête DKIM-Signature. En particulier, il y a des raisons de croire que des relais vont réordonner les champs d'en-tête de façon potentiellement arbitraire.

Note de mise en œuvre pour information : les vérificateurs pourraient utiliser l'ordre comme indice de l'ordre de signature en l'absence de toute autre information. Cependant, d'autres indices comme la sémantique de plusieurs signatures (comme de corréler l'hôte signataire avec les champs d'en-tête Received) pourraient aussi être considérés.

La survivabilité des signatures après le transit n'est pas garantie, et des signatures peuvent échouer à se vérifier sans faute du signataire. Donc, un vérificateur NE DEVRAIT PAS traiter un message qui a une ou plusieurs mauvaises signatures et pas de bonne signature différemment d'un message avec pas de signature du tout.

Quand une signature se vérifie bien, un vérificateur va soit arrêter le traitement, soit tenter de vérifier d'autres signatures, à la discrétion de la mise en œuvre. Un vérificateur PEUT limiter le nombre de signatures qu'il essaye, afin d'éviter des attaques de déni de service (voir la discussion du paragraphe 8.4).

Dans la description qui suit, le texte disant "état de retour (explication)" (où "état" est un de "PERMFAIL" ou "TEMPFAIL") signifie que le vérificateur DOIT immédiatement cesser de traiter cette signature. Le vérificateur DEVRAIT passer à la prochaine signature, si une est présente, et complètement ignorer la mauvaise signature. Si l'état est "PERMFAIL", la signature a échoué et ne devrait pas être reconsidérée. Si l'état est "TEMPFAIL", la signature pourrait ne pas être vérifiée à ce moment mais peut être réessayée plus tard. Un vérificateur PEUT soit s'arranger pour différer le message pour un traitement ultérieur, soit essayer une autre signature ; si aucune bonne signature n'est trouvée et si une des signatures a résulté en un état TEMPFAIL, le vérificateur PEUT s'arranger pour différer le message pour traitement ultérieur. Le "(explication)" n'est pas un texte normatif ; il est fourni seulement pour une clarification.

Les vérificateurs qui sont prêts à valider plusieurs champs d'en-tête Signature DEVRAIENT passer au prochain champ d'en-tête Signature, si il en existe un. Cependant, les vérificateurs PEUVENT prendre note du fait qu'une signature invalide était présente pour en tenir compte à une étape ultérieure.

Note pour information : la raison de cette exigence est de permettre que les messages qui ont des signatures invalides mais aussi une signature valide fonctionnent. Par exemple, un expanseur de liste de diffusion pourrait opter pour laisser en place la signature du soumetteur original même si l'expanseur sait qu'il modifie le message d'une certaine façon qui va casser cette signature, et l'expanseur insère sa propre signature. Dans ce cas, le message devrait réussir même en présence de la signature cassée connue.

Pour que chaque signature soit validée, les étapes suivantes devraient être effectuées de façon à produire un résultat sémantiquement équivalent à les effectuer dans l'ordre indiqué.

6.1.1 Validation du champ d'en-tête Signature

Les mises en œuvre DOIVENT méticuleusement valider le format et les valeurs dans le champ d'en-tête DKIM-Signature ; toute incohérence ou valeur inattendue DOIT entraîner d'ignorer complètement le champ d'en-tête et le vérificateur doit retourner une PERMFAIL (erreur de syntaxe de signature). Être "libéral dans ce qu'on accepte" est définitivement une mauvaise stratégie dans ce contexte de sécurité. Noter, cependant, que cela n'inclut pas l'existence d'étiquettes inconnues dans un champ d'en-tête DKIM-Signature, qui sont explicitement permises. Les vérificateurs DOIVENT retourner une PERMFAIL (version incompatible) quand est présenté un champ d'en-tête DKIM-Signature avec une étiquette "v=" qui n'est pas cohérente avec la présente spécification.

Note de mise en œuvre pour information : une mise en œuvre peut, bien sûr, choisir de vérifier aussi des signatures générées par de plus anciennes versions de cette spécification.

Si une étiquette mentionnée comme "exigée" au paragraphe 3.5 est omise du champ d'en-tête DKIM-Signature, le vérificateur DOIT ignorer le champ d'en-tête DKIM-Signature et retourner une PERMFAIL (étiquette de signature manquante exigée).

Note pour information : les étiquettes mentionnées comme exigées au paragraphe 3.5 sont "v=", "a=", "b=", "bh=", "d=", "h=", et "s=". Si il devait y avoir un conflit entre cette note et le paragraphe 3.5, celui-ci est normatif.

Si le champ d'en-tête DKIM-Signature ne contient pas d'étiquette "i=", le vérificateur DOIT se comporter comme si la valeur de cette étiquette était "@d", où "d" est la valeur de l'étiquette "d=".

Les vérificateurs DOIVENT confirmer que le domaine spécifié dans l'étiquette "d=" est le même que celle du domaine qui fait partie de l'étiquette "i=", ou d'un domaine parent. Sinon, le champ d'en-tête DKIM-Signature DOIT être ignoré, et le vérificateur devrait retourner une PERMFAIL (discordance de domaine).

Si l'étiquette "h=" n'inclut pas de champ d'en-tête From, le vérificateur DOIT ignorer le champ d'en-tête DKIM-Signature et retourner une PERMFAIL (champ From non signé).

Les vérificateurs PEUVENT ignorer le champ d'en-tête DKIM-Signature et retourner une PERMFAIL (signature expirée) si il contient une étiquette "x=" et que la signature a expiré.

Les vérificateurs PEUVENT ignorer le champ d'en-tête DKIM-Signature si le domaine utilisé par le signataire dans l'étiquette "d=" n'est pas associé à une entité signante valide. Par exemple, des signatures avec des valeurs "d=" telles que "com" et "co.uk" pourraient être ignorées. La liste des domaines inacceptables DEVRAIT être configurable.

Les vérificateurs PEUVENT ignorer le champ d'en-tête DKIM-Signature et retourner une PERMFAIL (en-tête de signature inacceptable) pour toute autre raison, par exemple, si la signature ne signe pas des champs d'en-tête que le vérificateur voit comme essentiels. En fait, si des champs d'en-tête MIME ne sont pas signés, certaines attaques que le vérificateur préférerait éviter peuvent être possibles.

6.1.2 Obtention de la clé publique

La clé publique pour une signature est nécessaire pour achever le processus de vérification. Le processus de restitution de clé publique dépend du type d'interrogation, défini par l'étiquette "q=" dans le champ d'en-tête DKIM-Signature. Évidemment, une clé publique a seulement besoin d'être restituée si le processus d'extraction des informations de signature est complètement réussi.

Les détails de la gestion et représentation de la clé sont décrits au paragraphe 3.6. Le vérificateur DOIT valider l'enregistrement de clé et DOIT ignorer tout enregistrement de clé publique mal formé.

Note : l'utilisation d'un RR TXT à caractère générique qui couvre un nom de domaine DKIM interrogé va produire une réponse à une interrogation DKIM qui a peu de chances d'être un enregistrement de clé DKIM valide. Ce problème n'est pas spécifique de DKIM et s'applique à de nombreux autres types d'interrogations. Le logiciel client qui traite les réponses DNS doit tenir compte de ce problème.

Quand il valide un message, un vérificateur DOIT effectuer les étapes suivantes d'une manière sémantiquement la même que les effectuer dans l'ordre indiqué ; dans certains cas, la mise en œuvre peut paralléliser ou réordonner ces étapes, pour autant que la sémantique reste inchangée :

1. Le vérificateur restitue la clé publique comme décrit au paragraphe 3.6 en utilisant l'algorithme de l'étiquette "q=", le domaine de l'étiquette "d=", et le sélecteur de l'étiquette "s=".

2. Si l'interrogation pour la clé publique échoue, le vérificateur PEUT chercher une tentative de vérification ultérieure en retournant une TEMPFAIL (clé indisponible).

3. Si l'interrogation pour la clé publique échoue parce que l'enregistrement de clé correspondant n'existe pas, le vérificateur DOIT immédiatement retourner une PERMFAIL (pas de clé pour la signature).

4. Si l'interrogation pour la clé publique retourne plusieurs enregistrements de clé, le vérificateur peut choisir un des enregistrements de clé ou peut tourner à travers les enregistrements de clé, effectuant le reste de ces étapes sur chaque enregistrement à la discrétion de la mise en œuvre. L'ordre des enregistrements de clé est non spécifié. Si le vérificateur choisit de tourner à travers les enregistrements de clé, la formulation "retourner ..." dans le reste de ce paragraphe signifie "essayer le prochain enregistrement de clé, si il y en a ; si il n'y en a pas, retourner pour essayer une autre signature de la façon usuelle".

5. Si le résultat retourné de l'interrogation ne respecte pas le format défini dans cette spécification, le vérificateur DOIT ignorer l'enregistrement de clé et retourner une PERMFAIL (erreur de syntaxe ce clé). Les vérificateurs sont invités à valider la syntaxe des enregistrements de clé avec soin pour éviter des tentatives d'attaques. En particulier, le vérificateur DOIT ignorer les clés avec un code de version (étiquette "v=") qu'il ne met pas en œuvre.

6. Si l'étiquette "h=" existe dans l'enregistrement de clé publique et si l'algorithme de hachage impliqué par l'étiquette "a=" dans le champ d'en-tête DKIM-Signature n'est pas inclus dans le contenu de l'étiquette "h=", le vérificateur DOIT ignorer l'enregistrement de clé et retourner une PERMFAIL (algorithme de hachage inapproprié).

7. Si les données de clé publique (l'étiquette "p=") sont vides, alors cette clé a été révoquée et le vérificateur DOIT traiter cela comme un échec de vérification de signature et retourner une PERMFAIL (clé révoquée). Il n'y a pas de différence sémantique définie entre une clé révoquée et un enregistrement de clé supprimé.

8. Si les données de clé publique ne conviennent pas pour être utilisées avec les types d'algorithme et de clés définis par les étiquettes "a=" et "k=" dans le champ d'en-tête DKIM-Signature, le vérificateur DOIT immédiatement retourner une PERMFAIL (algorithme de clé inapproprié).

6.1.3 Calcul de la vérification

Étant donnés un signataire et une clé publique, vérifier une signature consiste en actions sémantiquement équivalentes aux étapes suivantes :

1. Sur la base de l'algorithme défini dans l'étiquette "c=", la longueur de corps spécifiée dans l'étiquette "l=", et le nom du champ d'en-tête dans l'étiquette "h=", on prépare une version canonisée du message comme décrit au paragraphe 3.7 (noter que cette version canonisée ne remplace pas en fait le contenu original). Quand on confronte les noms de champ d'en-tête dans l'étiquette "h=" aux champs d'en-tête du message réel, les comparaisons DOIVENT être insensibles à la casse.

2. Sur la base de l'algorithme indiqué dans l'étiquette "a=", on calcule les hachages de message à partir de la copie canonique comme décrit au paragraphe 3.7.

3. Vérifier que le hachage du corps de message canonisé calculé à l'étape précédente correspond à la valeur de hachage portée dans l'étiquette "bh=". Si le hachage ne correspond pas, le vérificateur DEVRAIT ignorer la signature et retourner une PERMFAIL (le hachage du corps n'est pas vérifié).

4. En utilisant la signature portée dans l'étiquette "b=", vérifier la signature par rapport au hachage d'en-tête en utilisant le mécanisme approprié pour l'algorithme de clé publique décrit dans l'étiquette "a=". Si la signature ne se valide pas, le vérificateur DEVRAIT ignorer la signature et retourner une PERMFAIL (la signature n'est pas vérifiée).

5. Autrement, la signature se vérifie correctement.

Note de mise en œuvre facultative : les mises en œuvre pourraient souhaiter initier l'interrogation de clé publique en parallèle avec le calcul du hachage car la clé publique n'est nécessaire que quand le déchiffrement final est calculé. Les mises en œuvre peuvent aussi vérifier la signature sur l'en-tête de message avant de valider que le hachage de message mentionné dans l'étiquette "bh=" dans le champ d'en-tête DKIM-Signature correspond à celui du corps de message réel ; cependant, si le hachage du corps ne correspond pas, la signature entière doit être considérée comme ayant échoué.

Une longueur de corps spécifiée dans l'étiquette "l=" de la signature limite le nombre d'octets du corps passés à l'algorithme de vérification. Toutes les données au delà de cette limite sont non validées par DKIM. Donc, les vérificateurs pourraient tenir pour suspect un message qui contient des octets au delà de la longueur de corps indiquée et peuvent choisir de traiter la signature comme si elle était invalide (par exemple, en retournant une PERMFAIL (contenu non signé)).

Si l'algorithme atteint ce point, la vérification a réussi, et DKIM rapporte SUCCESS pour cette signature.

6.2 Communication des résultats de la vérification

Les vérificateurs qui souhaitent communiquer les résultats de la vérification aux autres parties du système de messagerie peuvent le faire de toutes les façons qui leur paraissent bonnes. Par exemple, des mises en œuvre pourraient choisir d'ajouter un champ d'en-tête email au message avant de le passer. Tout champ d'en-tête de ce genre DEVRAIT être inséré avant tout champ d'en-tête DKIM-Signature existant ou état d'authentification préexistant dans le bloc de champs d'en-tête. Le champ d'en-tête Authentication-Results: ([RFC5451]) PEUT être utilisé à cette fin.

Avis pour information aux auteurs de filtre de MUA : les schémas destinés à chercher les résultats de champs d'en-tête pour marquer de façon visible les messages authentifiés pour les utilisateurs finaux devraient vérifier qu'un tel champ d'en-tête a été ajouté par le domaine de vérification approprié et que l'identité vérifiée correspond à celle de l'auteur qui va être affichée par le MUA. En particulier, les filtres de MUA ne devraient pas être influencés par des résultats de champs d'en-tête bogués ajoutés par des attaquants. Pour contrer cette attaque, les vérificateurs PEUVENT souhaiter demander la suppression des champs d'en-tête de résultat existants après la vérification et avant de s'arranger pour ajouter un nouveau champ d'en-tête.

6.3 Interprétation de la politique locale de résultats/application

Il sort du domaine d'application de la présente spécification de décrire quelles actions peut effectuer un témoin d'identité, mais les messages qui portent un SDID validé présentent une opportunité à un témoin d'identité que n'a pas le message non authentifié. Précisément, un message authentifié crée un identifiant prévisible par lequel d'autres décisions peuvent être gérées de façon fiable, comme la confiance et la réputation. À l'inverse, le message non authentifié manque d'un identifiant fiable qui peut être utilisé pour allouer la confiance et la réputation. Il est raisonnable de traiter le message non authentifié comme manquant de confiance et n'ayant pas de réputation positive.

En général, les modules qui consomment le résultat de la vérification DKIM NE DEVRAIENT PAS déterminer l'acceptabilité de message sur la seule base du manque de signature ou d'une signature invérifiable ; un tel rejet causerait de sévères problèmes d'interopérabilité. Si un MTA souhaite rejeter de tels messages durant une session SMTP (par exemple, quand il communique avec un homologue qui, par accord préalable, accepte seulement les messages envoyés signés) et si une signature manque ou ne se vérifie pas, le MTA traitant DEVRAIT utiliser une code de réponse 550/5.7.x.

Lorsque le vérificateur est intégré dans le MTA et qu'il n'est pas possible d'aller chercher la clé publique, peut-être parce que le serveur de clé n'est pas disponible, un message d'échec temporaire PEUT être généré en utilisant un code de réponse 451/4.7.5, tel que : 451 4.7.5 Incapable de vérifier la signature - serveur de clés indisponible.

Des défaillances temporaires comme l'incapacité d'accéder au serveur de clés ou autre service externe sont les seules conditions qui DEVRAIENT utiliser un code de réponse 4xx SMTP. En particulier, les défaillances de vérification de signature cryptographique NE DOIVENT PAS provoquer de réponse 4xx SMTP.

Une fois que la signature a été vérifiée, cette information DOIT être portée au témoin d'identité (comme un système explicite d'autorisation/liste blanche et réputation) et/ou à l'utilisateur final. Si le SDID n'est pas le même que l'adresse dans le champ d'en-tête From:, le système de messagerie DEVRAIT s'assurer que le SDID actuel est clair pour le lecteur.

Bien que les symptômes d'un échec de vérification soient évidents -- la signature ne se vérifie pas – établir la cause exacte peut être plus difficile. Si un sélecteur ne peut pas être trouvé, est-ce parce que le sélecteur a été supprimé, ou la valeur a-telle changé dans le transit ? Si la ligne de signature manque, est ce parce qu'elle n'y a jamais été, ou a t-elle été supprimée par un filtre trop zélé ? Pour les besoins de diagnostic, la raison exact de l'échec de vérification DEVRAIT être rendue disponible et éventuellement enregistrée dans les journaux du système.

Si le message ne peut pas être vérifié, il DEVRAIT alors être traité de la même façon que tout message non vérifié, sans considération de si il a l'air d'être ou non signé. Voir au paragraphe 8.15 une discussion supplémentaire.

7. Considérations relatives à l'IANA

DKIM a enregistré des espaces de noms auprès de l'IANA. Dans tous les cas, les nouvelles valeurs sont allouées seulement pour des valeurs qui ont été documentées dans une RFC publiée qui a le consensus de l'IETF [RFC5226].

Le présent mémoire met à jour ces registres comme décrit ci-dessous. On notera l'ajout d'une nouvelle colonne "Statut".

Tous les enregistrements dans ces espaces de noms DOIVENT inclure le nom à enregistrer, le document dans lequel il a été enregistré ou mis à jour, et l'indication de son statut actuel, qui DOIT être soit "actif" (en utilisation) ou "historique" (n'est plus utilisé).

Aucune nouvelle étiquette n'est définie dans la présente spécification par rapport à la [RFC4871], mais une a été désignée comme "historique".

Aussi, le registre "Méthodes d'authentification de messagerie" est révisé pour se référer à cette mise à jour.

7.1 Registre des méthodes d'authentification de message électronique

Le registre "Méthodes d'authentification de messagerie" est mis à jour pour indiquer que "dkim" est défini dans ce mémoire.

7.2 Spécifications d'étiquette DKIM-Signature

Une signature DKIM fournit une liste de spécifications d'étiquettes. L'IANA a établi le registre "Spécifications d'étiquettes de signature DKIM" pour les spécifications d'étiquettes qui peuvent être utilisées dans le champ DKIM-Signature.

Type	Référence	Statut
v	RFC6376	actif
a	RFC6376	actif
b	RFC6376	actif
bh	RFC6376	actif
c	RFC6376	actif
d	RFC6376	actif
h	RFC6376	actif
i	RFC6376	actif
l	RFC6376	actif
q	RFC6376	actif
s	RFC6376	actif
t	RFC6376	actif
x	RFC6376	actif
z	RFC6376	actif

Tableau 1 : Valeurs mises à jour du registre des spécifications d'étiquette DKIM-Signature

7.3 Registre des méthodes d'interrogation DKIM-Signature

La spécification d'étiquette "q=" (spécifiée au paragraphe 3.5) fournit la liste des méthodes d'interrogation.

L'IANA a établi le registre "Méthode d'interrogation DKIM-Signature" pour les mécanismes qui peuvent être utilisés pour restituer la clé qui va permettre le processus de validation d'un message signé en utilisant DKIM.

Type	Options	Référence	Statut
dns	txt	RFC6376	actif

Tableau 2 : Valeurs mises à jour du registre des méthodes d'interrogation DKIM-Signature

7.4 Registre de canonisation de DKIM-Signature

La spécification d'étiquette"c=" (spécifiée au paragraphe 3.5) fournit un spécificateur pour les algorithmes de canonisation de l'en-tête et du corps du message.

L'IANA a établi le registre "En-tête de canonisation de DKIM-Signature" pour les algorithmes de conversion d'un message en forme canonique avant de signer ou vérifier en utilisant DKIM.

Type	Référence	Statut
simple	RFC6376	actif
lâche	RFC6376	actif

Tableau 3 : Valeurs mises à jour du registre d'en-tête de canonisation DKIM-Signature

Type	Référence	Statut
simple	RFC6376	actif
lâche	RFC6376	actif

Tableau 4 : Valeurs mises à jour du registre de corps de canonisation DKIM-Signature

7.5 Spécifications d'étiquette d'enregistrement de ressource _domainkey TXT du DNS

Un RR DNS TXT _domainkey fournit une liste de spécifications d'étiquettes. L'IANA a établi le registre DKIM "Spécifications d'étiquette d'enregistrement TXT DNS _domainkey" pour les spécifications d'étiquettes qui peuvent être utilisées dans les enregistrements de ressource TXT du DNS.

Type	Référence	Statut
v	RFC6376	actif
g	[RFC4871]	historique
h	RFC6376	actif
k	RFC6376	actif
n	RFC6376	actif
p	RFC6376	actif
s	RFC6376	actif
t	RFC6376	actif

Tableau 5 : Valeurs mises à jour du registre de spécifications d'étiquette _domainkey TXT du DNS

7.6 Registre des types de clé DKIM

Les étiquettes <key-k-tag> "k=" (spécifiée au paragraphe 3.6.1) et <sig-a-tag-k> "a=" (spécifiée au paragraphe 3.5) fournissent une liste de mécanismes qui peuvent être utilisés pour décoder une signature DKIM.

L'IANA a établi le registre "Type de clé DKIM"pour ces mécanismes.

Type	Référence	Statut
rsa	[RFC3447]	actif

Tableau 6 : Valeurs mises à jour du registre de type de clé DKIM

7.7 Registre des algorithmes de hachage DKIM

Les étiquettes "h=" <key-h-tag> (spécifiée au paragraphe 3.6.1) et "a=" <sig-a-tag-h> (spécifiée au paragraphe 3.5) fournissent la liste des mécanismes qui peuvent être utilisés pour produire un résumé des données de message.

L'IANA a établi le registre "Algorithmes de hachage DKIM" pour ces mécanismes.

Type	Référence	Statut
sha1	[FIPS-180-3]	actif
sha256	[FIPS-180-3]	actif

Tableau 7 : Valeurs mises à jour du registre des algorithmes DKIM

7.8 Registre des types de service DKIM

L'étiquette "s=" <key-s-tag> (spécifiée au paragraphe 3.6.1) fournit une liste des types de service auxquels ce sélecteur peut s'appliquer.

L'IANA a établi le registre "Types de service DKIM" pour les types de service.

Type	Référence	Statut
email	RFC6376	actif
*	RFC6376	actif

Tableau 8 : Valeurs mises à jour du registre des types de service DKIM

7.9 Registre des fanions de sélecteur DKIM

L'étiquette "t=" <key-t-tag> (spécifiée au paragraphe 3.6.1) fournit une liste de fanions pour modifier l'interprétation du sélecteur.

L'IANA a établi le registre "Fanions de sélecteur DKIM " pour des fanions supplémentaires.

Type	Référence	Statut
y	RFC6376	actif
s	RFC6376	actif

Tableau 9 : Valeurs mises à jour du registre des fanions de sélecteur DKIM

7.10 Champ d'en-tête DKIM-Signature

L'IANA a ajouté DKIM-Signature au registre "Noms de champ d'en-tête de message permanent" (voir dans la [RFC3864]) le protocole "mail", en utilisant le présent document comme référence.

8. Considérations sur la sécurité

Il a été observé que tout mécanisme introduit qui tente d'endiguer le flux de pourriels est l'objet d'attaques intenses. DKIM doit être étudié attentivement pour identifier les vecteurs d'attaque potentiels et les faiblesses de chacun. Voir aussi la [RFC4686].

8.1 Attaques sur ASCII

L'algorithme de canonisation lâche de corps peut permettre certains types d'attaques extrêmement crues de "art de l'ASCII" où un message peut être porté en ajustant l'espacement entre les mots. Si cela pose problème, l'algorithme "simple" de canonisation de corps devrait être utilisé à la place.

8.2 Emploi abusif des limites de longueur de corps (étiquette "l=")

L'utilisation de l'étiquette "l=" pourrait permettre d'afficher un contenu frauduleux sans avertissement approprié aux utilisateurs finaux. L'étiquette "l=" est destinée à augmenter la robustesse de la signature lors d'envois à des listes de diffusion qui à la fois modifient leur contenu et ne signent pas leurs messages modifiés. Cependant, en utilisant l'étiquette "l=" on permet des attaques dans lesquelles un intermédiaire avec des intentions malveillantes peut modifier un message pour y inclure du contenu qui bénéficie seulement à l'attaquant. Il est possible que le contenu ajouté remplace complètement le contenu original aux yeux du receveur final et déjoue les algorithmes de détection de message dupliqué.

Un exemple d'une telle attaque inclut d'altérer la structure MIME, exploitant l'analyse HTML lâche au MUA, et déjouant les algorithmes de détection de message dupliqué.

Pour éviter cette attaque, les signataires devraient être extrêmement prudents en utilisant cette étiquette, et les témoins

pourraient souhaiter ignorer les signatures qui utilisent cette étiquette.

8.3 Clé privée inappropriée

Comme avec toute autre application de sécurité qui utilise des paires de clés privées ou publiques, DKIM exige de faire attention pour le traitement et la protection des clés. Une clé privée ou un accès compromis à une d'elles signifie qu'un intrus ou un malveillant peut envoyer des messages signés par le domaine qui annoncent la clé publique correspondante.

Donc, les clés privées produites aux utilisateurs, plutôt qu'une utilisée par un domaine de gestion administratif (ADMD, *ADministrative Management Domain*) lui-même, crée le problème habituel de sécuriser les données mémorisées sur les ressources personnelles qui peuvent affecter le ADMD.

Une architecture plus sécurisée impliquant d'envoyer les messages à travers un MTA sortant qui peut authentifier le soumettant en utilisant les techniques existantes (par exemple, l'authentification SMTP) éventuellement valider le message lui-même (par exemple, vérifier que l'en-tête est légitime et que le contenu réussit une vérification de contenu de pourriel) et signe le message en utilisant une clé appropriée pour l'adresse du soumettant. Un tel MTA peut aussi appliquer des contrôles sur le volume de messages sortants que chaque utilisateur a la permission de générer afin de limiter encore la capacité qu'un malveillant génère des messages en masse.

8.4 Attaques de déni de service du serveur de clé

Comme les serveurs de clés sont distribués (éventuellement séparément pour chaque domaine) le nombre de serveurs qui vont devoir être attaqués pour vaincre ce mécanisme au niveau de l'Internet est très grand. Néanmoins, les serveurs de clés pour des domaines individuels pourraient être attaqués, empêchant la vérification des messages provenant de ce domaine. Ceci n'est pas significativement différent de la capacité d'un attaquant de dénier le service aux échangeurs de messages pour un domaine donné, bien qu'il affecte les messages sortants, mais pas entrants.

Une variante de cette attaque implique une très grande quantité de messages envoyés en utilisant des signatures contrefaites provenant d'un domaine : les serveurs de clés pour ce domaine pourraient être surchargés par des demandes dans une attaque de déni de service (voir la [RFC4732]). Cependant, étant donnés les faibles frais généraux de la vérification par rapport au traitement du message lui-même, une telle attaque va être difficile à monter.

8.5 Attaques contre le DNS

Comme le DNS est un lien obligé pour les services de clés, des attaques spécifiques contre le DNS doivent être envisagées.

Alors que les DNS est actuellement non sûr [RFC3833], ces problèmes de sécurité sont la motivation derrière la sécurité du DNS (DNSSEC, *DNS Security*) [RFC4033], et tous les utilisateurs du DNS vont engranger les bénéfices de ce travail.

DKIM est seulement destiné à être une méthode "suffisante" pour prouver l'authenticité. Il n'est pas destiné à fournir une forte preuve cryptographique sur l'auteur ou le contenu. D'autres technologies comme OpenPGP [RFC4880] et S/MIME [RFC5751] traitent ces exigences.

Un second problème de sécurité relatif au DNS tourne autour de l'augmentation du trafic du DNS comme conséquence de la collecte des données fondées sur le sélecteur ainsi que d'aller chercher la politique du domaine signant. Le large déploiement de DKIM va résulter en une augmentation significative des interrogations au DNS pour le domaine signant revendiqué. Dans le cas de falsifications à grande échelle, les serveurs du DNS pourraient voir une augmentation substantielle des interrogations.

Un problème de sécurité du DNS spécifique qui devrait être considéré par les vérificateurs DKIM est l'attaque de chaînage de noms décrite au paragraphe 2.3 de la [RFC3833]. Un vérificateur DKIM, lorsque il vérifie un champ d'en-tête DKIM-Signature, pourrait être invité à restituer un enregistrement de clé du choix d'un attaquant. Cette menace peut être minimisée en s'assurant que les serveurs de noms, y compris les serveurs de noms récurrents, utilisés par le vérificateur appliquent une vérification stricte des informations de "glu" et autres informations supplémentaires dans les réponses du DNS et ne sont donc pas vulnérables à cette attaque.

8.6 Attaques en répétition/diffusion de pourriels

Dans cette attaque, un diffuseur envoie un pourriel à travers un MTA qui le signe, misant sur la réputation du domaine signant (par exemple, un grand fournisseur de messagerie populaire) plutôt que sur la sienne, et ensuite renvoie ce message à un grand nombre de receveurs. Les receveurs observent la signature valide provenant du domaine bien connu, élevant leur confiance dans le message et augmentant la probabilité de livraison et présentation à l'utilisateur.

Des solutions partielles à ce problème impliquent l'utilisation de services de réputation pour porter le fait que l'adresse de messagerie électronique spécifique est utilisée pour du pourriel et que les messages provenant de ce signataire sont probablement des pourriels. Cela exige un mécanisme de détection en temps réel afin de réagir assez vite. Cependant, de telle mesures pourraient être enclines à des abus, si, par exemple, un attaquant renvoie un grand nombre de messages reçus d'une victime afin de faire passer la victime comme un envoyeur de pourriels.

Les grands vérificateurs pourraient être capables de détecter des volumes inhabituellement grands de messages avec la même signature sur une courte période. De plus petits vérificateurs peuvent obtenir substantiellement le même volume d'informations via les systèmes collaboratifs existants.

8.7 Limites de la révocation de clés

Quand un grand domaine détecte un comportement indésirable de la part d'un de ses utilisateurs, il pourrait souhaiter révoquer la clé utilisée pour signer les messages de cet utilisateur afin de désavouer la responsabilité des messages qui n'ont pas encore été vérifiés ou qui sont l'objet d'une attaque en répétition. Cependant, la capacité du domaine à faire cela peut être limitée si la même clé, pour des raisons d'adaptabilité, est utilisée pour signer les messages pour de nombreux autres utilisateurs. Des mécanismes pour révoquer explicitement les clés adresse par adresse ont été proposés mais exigent des études complémentaires sur leur utilité et la charge qu'ils représentent pour le DNS.

8.8 Enregistrements de clé intentionnellement mal formés

Il est possible à un attaquant de publier des enregistrements de clé dans le DNS qui sont intentionnellement mal formés, avec l'intention de causer une attaque de déni de service sur une mise en œuvre de vérificateur non robuste. L'attaquant pourrait alors amener un vérificateur à lire l'enregistrement de clé mal formé en envoyant un message à un de ses utilisateurs référençant l'enregistrement mal formé dans une signature (pas nécessairement valide). Les vérificateurs DOIVENT vérifier attentivement tous les enregistrements de clé restitués du DNS et être robuste contre les enregistrements de clé intentionnellement et involontairement mal formés.

8.9 Champs d'en-tête DKIM-Signature intentionnellement mal formés

Les vérificateurs DOIVENT être prêts à recevoir des messages avec des champs d'en-tête DKIM-Signature mal formés et vérifier attentivement les champ d'en-tête avant de s'appuyer sur un de leurs contenus.

8.10 Fuite d'informations

Un attaquant pourrait déterminer quand une signature particulière a été vérifiée en utilisant un sélecteur par message et en surveillant ensuite son trafic DNS pour la recherche de clé. Cela agirait comme un équivalent d'une "web bug" pour une heure de vérification plutôt que pour l'heure à laquelle le message a été lu.

8.11 Attaques de temporisation à distance

Dans certains cas, il sera peut être possible d'extraire les clés privées en utilisant une attaque de temporisation à distance [BONEH03]. Les mises en œuvre devrait envisager de masquer les heures pour empêcher de telles attaques.

8.12 Réorganisation des champs d'en-tête

Les normes existantes permettent aux MTA intermédiaires de réordonner les champs d'en-tête. Si un signataire signe deux champs d'en-tête du même nom ou plus, cela peut causer des erreurs de vérification parasites sur des messages par ailleurs légitimes. En particulier, les signataires qui signent des champs DKIM-Signature existants courent le risque d'avoir des messages qui échouent à tort à se vérifier.

8.13 Attaques RSA

Un attaquant pourrait créer une grande clé de signature RSA avec un petit exposant, exigeant donc que la clé de vérification ait un grand exposant. Cela va forcer les vérificateurs à utiliser des ressources de calcul considérables pour vérifier la signature. Les vérificateurs pourraient éviter cette attaque en refusant de vérifier les signatures qui font référence à des sélecteurs avec des clés publiques qui ont des exposants déraisonnables.

En général, un attaquant pourrait essayer de submerger un vérificateur en l'inondant de messages qui exigent une vérification. Ceci est similaire aux autres attaques de déni de service de MTA et devrait être traité de façon similaire.

8.14 Signature inappropriée par les domaines parents

La relation de confiance décrite au paragraphe 3.10 pourrait conceptuellement être utilisée par un domaine parent pour signer les messages avec des identités dans un sous domaine sans rapport administratif avec le parent. Par exemple, le registre ".com" pourrait créer des messages avec des signatures utilisant une valeur "i=" dans le domaine exemple.com. Il n'y a pas de solution générale à ce problème, car la coupure administrative pourrait se produire n'importe où dans le nom de domaine. Par exemple, dans le domaine "exemple.podunk.ca.us", il y a trois coupures administratives (podunk.ca.us, ca.us, et us) dont chacune pourrait créer des messages avec une identité dans le domaine complet.

Note pour information : ceci est considéré être un risque acceptable pour la même raison qu'il est acceptable pour la délégation de domaine. Par exemple, dans le cas ci-dessus, un des domaines pourrait éventuellement simplement déléguer "exemple.podunk.ca.us" à un serveur de son choix et remplacer complètement toutes les informations servies par le DNS. Noter qu'un vérificateur PEUT ignorer les signatures qui viennent d'un domaine improbable tel que ".com", comme discuté au paragraphe 6.1.1.

8.15 Attaques impliquant des champs d'en-tête supplémentaires

De nombreux composants de la messagerie électronique, incluant les MTA, MSA, MUA, et les modules de filtrage, mettent en œuvre seulement de façon lâche les vérifications de format de message. Cela résulte d'années de pression de la part de l'industrie pour être libéral dans ce qui est accepté dans le flux de mesages au nom de la réduction des coûts de prise en charge ; les messages formatés de façon impropre sont souvent corrigés en silence dans le transit, livrés non réparés, ou affichés de façon inappropriée (par exemple, en montrant seulement le premier de plusieurs champs From:).

Les agents qui évaluent ou appliquent le résultat de DKIM doivent savoir qu'un signataire DKIM peut signer des messages qui sont mal formés (par exemple, violent la [RFC5322], comme en ayant plusieurs instances d'un champ qui est seulement permis une fois) qui devient mal formé dans le transit, ou comporte un en-tête ou contenu de corps qui n'est pas vrai ou valide. L'utilisation de DKIM sur de tels messages pourrait constituer une attaque contre un receveur, en particulier lorsque une créance supplémentaire est accordée à un message signé sans évaluation adéquate du signataire.

Cela peut représenter des attaques sérieuses, mais elles n'ont rien à voir avec DKIM ; ce sont des attaques contre le receveur ou sur l'auteur identifié à tort.

De plus, un agent serait incorrect s'il concluait que toutes les instances d'un champ d'en-tête sont signées juste parce que une l'est.

Une signature authentique provenant du domaine attaqué peut être obtenue par des moyens légitimes, mais des champs d'en-tête supplémentaires peuvent être ajoutés ensuite, par interception ou par répétition. Dans ce scénario, DKIM peut aider à détecter l'ajout de champs spécifiques dans le transit. C'est fait par la liste établie par le signataire des noms de champs dans l'étiquette "h=" une fois de plus (par exemple, "h=from:from:..." pour un message avec un champ From) de sorte que cet ajout d'une instance de ce champ en aval va rendre impossible que la signature se vérifie. (Voir les détails au paragraphe 3.5.) Ceci est par nature une indication explicite que le signataire répudie la responsabilité d'un tel message mal formé.

DKIM signe et valide les données qu'on lui dit et fonctionne correctement. De sorte que dans ce cas, DKIM a fait son travail de livraison d'un domaine validé (la valeur "d=") et, étant donnée la sémantique d'une signature DKIM, le signataire a essentiellement pris une certaine responsabilité pour un message problématique. Il appartient au témoin d'identité ou à quelque autre agent suivant d'agir sur de tels messages comme nécessaire, comme de dégrader la confiance accordée au

message (ou, bien sûr, au signataire) avertir le receveur, ou même refuser la livraison.

Tous les composants du système de messagerie qui effectuent l'application lâche des autres normes de messagerie vont devoir revisiter cette posture quand il vont incorporer DKIM, en particulier quand ils considèrent les questions d'attaques potentielles comme celles décrites.

9. Références

9.1 Références normatives

[FIPS-180-3] U.S. Department of Commerce, "Secure Hash Standard", FIPS PUB 180-3, octobre 2008.

[UIT-X660] Recommandation UIT-T X.660, "Technologies de l'information - Règles de codage de l'ASN.1 : Spécification des règles de codage de base (BER) des règles de codage canoniques (CER) et des règles de codage distinctives (DER)", 1997.

[RFC1034] P. Mockapetris, "Noms de domaines - Concepts et facilités", STD 13, novembre 1987. (*MàJ par* RFC1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936, 8020, 8482, 8767)

[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.)

[RFC2049] N. Freed, N. Borenstein, "Extensions multi-objets de la messagerie Internet (MIME) Partie cinq : critères de conformité et exemples", novembre 1996. *(Remplace* RFC1521, RFC1522, RFC1590) *(D.S.)*

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

[RFC3447] J. Jonsson et B. Kaliski, "Normes de cryptographie à clés publiques (PKCS) n° 1 : Spécifications de la cryptographie RSA version 2.1", février 2003. *(Obsolète, remplacée par* RFC8017) *(Information)*

[RFC5234] D. Crocker, P. Overell, "BNF augmenté pour les spécifications de syntaxe : ABNF", janvier 2008. (STD0068)

[RFC5321] J. Klensin, "Protocole simple de transfert de messagerie(SMTP)", octobre 2008. (*Remplace* RFC2821) (*MàJ* RFC1123) *(D.S.)*

[RFC5322] P. Resnick, éd., "Format du message Internet", octobre 2008. (*Remplace* RFC2822) (MàJ RFC4021) *(D.S.)*

[RFC5598] D. Crocker, "Architecture de la messagerie Internet", juillet 2009. *(Information)*

[RFC5890] J. Klensin, "Noms de domaine internationalisés pour les applications (IDNA) : Définitions et cadre documentaire", août 2010. *(P.S. ; remplace RFC*3490)

9.2 Références pour information

[BONEH03] "Remote Timing Attacks are Practical", Proceedings 12th USENIX Security Symposium, 2003.

[RFC2047] K. Moore, "MIME (Extensions de messagerie Internet multi-objets) Partie trois : extensions d'en-tête de message pour texte non ASCII", novembre 1996. (*MàJ par* RFC2184, RFC2231) *(D.S.)*

[RFC3629] F. Yergeau, "UTF-8, un format de transformation de la norme ISO 10646", STD 63, novembre 2003.

[RFC3766] H. Orman, P. Hoffman, "Détermination de la force des clés publiques utilisées pour l'échange de clés symétriques", avril 2004. (BCP0086)

[RFC3833] D. Atkins, R. Austein, "Analyse des menaces contre le système des noms de domaines (DNS)", août 2004. *(Info.)*

[RFC3864] G. Klyne, M. Nottingham, J. Mogul, "Procédures d'enregistrement pour les champs d'en-tête de message", septembre 2004. (BCP0090 ; *MàJ par* RFC9110)

[RFC4033] R. Arends, et autres, "Introduction et exigences pour la sécurité du DNS", mars 2005.

[RFC4409] R. Gellens, J. Klensin, "Soumission du message de messagerie électronique", avril 2006. (*Remplacé par la* RFC6409 STD072*)*

[RFC4686] J. Fenton, "Analyse des menaces qui motivent la messagerie identifiée par clés de domaines (DKIM)", septembre 2006. *(Info.)*

[RFC4732] M. Handley et autres, "Considérations sur le déni de service dans l'Internet", décembre 2006. *(Information)*

[RFC4870] M. Delany, "Authentification de message électronique fondée sur le domaine avec des clés publiques annoncées dans le DNS (clés de domaine)", mai 2007. (*Obsolète, voir* RFC4871) *(Historique)*

[RFC4871] E. Allman et autres, "Signatures de messagerie identifiées par clés de domaines (DKIM)", mai 2007. *(P.S, remplace* RFC4870 ; *MàJ par* RFC5672 *; Remplacée par* RFC6376*).)*

[RFC4880] J. Callas et autres, "Format de message OpenPGP", novembre 2007. *(P.S., remplace* RFC1991, RFC2440)

[RFC5226] T. Narten et H. Alvestrand, "Lignes directrices pour la rédaction d'une section Considérations relatives à l'IANA dans les RFC", BCP 26, mai 2008. (*Remplace* RFC2434 *; remplacée par* RFC8126)

[RFC5451] M. Kucherawy, "Champ d'en-tête de message pour indiquer l'état d'authentification du message", avril 2009 *(P. S. ; MàJ par la* RFC6577 ; *Remplacée par* RFC8601)

[RFC5585] T. Hansen, D. Crocker, P. Hallam-Baker, "Généralités sur le service de messagerie identifiée par clés de domaine (DKIM)", juillet 2009. *(Information)*

[RFC5672] D. Crocker, "Signatures de messagerie identifiée par clés de domaines (DKIM) de la RFC 4871 – Mise à jour", août 2009. *(P. S. ; MàJ RFC4871* ; r*emplacée par la* RFC6376)

[RFC5751] B. Ramsdell, S. Turner, "Spécification du message des extensions de messagerie Internet multi objets sécurisée (S/MIME) version 3.2", janvier 2010. *(P. S. ;* r*emplace* RFC3851, *remplacée par* RFC8551*)*

[RFC5863] T. Hansen, E. Siegel, P. Hallam-Baker, D. Crocker, " Développement, déploiement et fonctionnement de la messagerie identifiée par clés de domaine (DKIM)", mai 2010. *(Information)*

[RFC6377] M. Kucherawy, "Messagerie identifiée par clés de domaine (DKIM) et listes de diffusion", septembre 2011. (BCP0167)

Appendice A. **Exemple d'utilisation** (pour information)

Cette section montre le flux complet d'un message, de la soumission à la livraison finale, montrant comment les divers composants s'assemblent. La clé utilisée dans cet exemple est montré dans l'Appendice C.

A.1 L'utilisateur compose un message

From: Joe SixPack <joe@football.exemple.com>
To: Suzie Q <suzie@shopping.exemple.net>
Subject: le dîner est il prêt ?
Date: Fri, 11 Jul 2003 21:00:37 -0700 (PDT)
Message-ID: <20030712040037.46341.5F8J@football.exemple.com>

Hi.

On a perdu la partie. As tu déjà faim ?

Joe.

Figure 1 : L'utilisateur compose un message

A.2 Le message est signé

Ce message est signé par le serveur de messagerie sortante exemple.com et ressemble maintenant à :

```
DKIM-Signature: v=1; a=rsa-sha256; s=brisbane; d=exemple.com;
    c=simple/simple; q=dns/txt; i=joe@football.exemple.com;
    h=Received : From : To : Subject : Date : Message-ID;
    bh=2jUSOH9NhtVGCQWNr9BrIAPreKQjO6Sn7XIkfJVOzv8=;
    b=AuUoFEfDxTDkHlLXSZEpZj79LICEps6eda7W3deTVFOk4yAUoqOB
    4nujc7YopdG5dWLSdNg6xNAZpOPr+kHxt1IrE+NahM6L/LbvaHut
    KVdkLLkpVaVVQPzeRDI009SO2Il5Lu7rDNH6mZckBdrIx0orEtZV
    4bmp/YzhwvcubU4=;
Received: from client1.football.exemple.com  [192.0.2.1]
    by submitserver.exemple.com avec SUBMISSION;
    Fri, 11 Jul 2003 21:01:54 -0700 (PDT)
From: Joe SixPack <joe@football.exemple.com>
To: Suzie Q <suzie@shopping.exemple.net>
Subject: le dîner est il prêt ?
Date: Fri, 11 Jul 2003 21:00:37 -0700 (PDT)
Message-ID: <20030712040037.46341.5F8J@football.exemple.com>

Hi.

On a perdu la partie. As tu déjà faim ?

Joe.
```

Figure 2 : le message est signé

Le serveur de messagerie signant exige l'accès à la clé privée associée au sélecteur "brisbane" pour générer cette signature.

A.3 La signature du message est vérifiée

La signature est normalement vérifiée par un serveur SMTP entrant ou éventuellement l'agent de livraison final. Cependant, les MTA intervenants peuvent aussi effectuer cette vérification si c'est leur choix. Le processus de vérification utilise le domaine "exemple.com" extrait de l'étiquette "d=" et le sélecteur "brisbane" de l'étiquette "s=" dans le champ d'en-tête DKIM-Signature pour former l'interrogation DNS DKIM pour : brisbane._domainkey.exemple.com

La vérification de signature commence avec le dernier champ physique d'en-tête Received, le champ d'en-tête From, et ainsi de suite, dans l'ordre mentionné dans l'étiquette "h=". La vérification suit avec un seul CRLF suivi par le corps (qui commence avec "Hi."). Le message est préparé canoniquement pour vérification avec la méthode "simple". Le résultat de l'interrogation et de la vérification suivante de la signature est mémorisé (dans cet exemple) dans la ligne de champ d'en-tête X-Authentication-Results. Après la réussite de la vérification, le message ressemble à :

```
X-Authentication-Results: shopping.exemple.net
  header.from=joe@football.exemple.com; dkim=pass
Received: from mout23.football.exemple.com (192.168.1.1)
  by shopping.exemple.net avec SMTP;
  Fri, 11 Jul 2003 21:01:59 -0700 (PDT)
```

```
DKIM-Signature: v=1; a=rsa-sha256; s=brisbane; d=exemple.com;
  c=simple/simple; q=dns/txt; i=joe@football.exemple.com;
  h=Received : From : To : Subject : Date : Message-ID;
  bh=2jUSOH9NhtVGCQWNr9BrIAPreKQjO6Sn7XIkfJVOzv8=;
  b=AuUoFEfDxTDkHlLXSZEpZj79LICEps6eda7W3deTVFOk4yAUoqOB
    4nujc7YopdG5dWLSdNg6xNAZpOPr+kHxt1IrE+NahM6L/LbvaHut
    KVdkLLkpVaVVQPzeRDI009SO2Il5Lu7rDNH6mZckBdrIx0orEtZV
    4bmp/YzhwvcubU4=;
Received: from client1.football.exemple.com  [192.0.2.1]
  by submitserver.exemple.com avec SUBMISSION;
  Fri, 11 Jul 2003 21:01:54 -0700 (PDT)
From: Joe SixPack <joe@football.exemple.com>
To: Suzie Q <suzie@shopping.exemple.net>
Subject: le dîner est il prêt ?
 Date: Fri, 11 Jul 2003 21:00:37 -0700 (PDT)
Message-ID: <20030712040037.46341.5F8J@football.exemple.com>

Hi.

On a perdu la partie. As tu déjà faim ?

Joe.
```

Figure 3 : Vérification réussie

Appendice B. Exemples d'usage (pour information)

La signature et validation DKIM peuvent être utilisées de différentes façons, pour différents scénarios de fonctionnement. Le présent Appendice discute certains exemples courants.

Note : les descriptions de cet Appendice sont seulement pour information. Elles décrivent diverses façons dont DKIM peut être utilisé, étant données des contraintes et besoins particuliers. En aucun cas ces exemples ne sont destinés à être pris comme fournissant des explications ou directives concernant les détails de la spécification DKIM dans la création d'une mise en œuvre.

B.1 Autres scénarios de soumission

Dans le plus simple des scénarios, le MUA, MSA, et MTA Internet (frontière) d'un utilisateur sont tous dans le même environnement administratif, utilisant le même nom de domaine. Donc, tous les composants impliqués dans la soumission et le transfert initial sont en relation. Cependant, il est courant que deux composants ou plus soient sous un contrôle administratif indépendant. Cela crée des défis pour choisir et administrer le nom de domaine à utiliser pour signer et pour sa relation aux champs d'en-tête communs d'identité de messagerie électronique.

B.1.1 Fonctions commerciales déléguées

Certaines organisations allouent des fonctions commerciales spécifiques à des groupes séparés, à l'intérieur ou à l'extérieur de l'organisation. Le but est alors d'autoriser ce groupe à signer certains messages mais de mettre des contraintes sur quelles signatures il peut générer. Les sélecteurs DKIM (l'étiquette de signature "s=") facilitent cette sorte d'autorisation restreinte. Des exemples de ces fonctions commerciales externalisées sont des fournisseurs légitimes de démarchage par messagerie électronique et des fournisseurs de bénéfices pour l'entreprise.

Ici, le groupe délégué doit être capable d'envoyer des messages qui sont signés, en utilisant le domaine de messagerie électronique de l'entreprise cliente. En même temps, le client est souvent réticent à enregistrer une clé pour le fournisseur qui accorde la capacité d'envoyer des messages pour des adresses arbitraires dans le domaine.

Il y a plusieurs façons d'administrer ces scénarios d'usage. Dans un cas, l'organisation cliente fournit toute l'administration de service d'interrogation publique (par exemple, le DNS) et dans un autre, elle utilise la délégation DNS pour permettre

toute l'administration courante de l'enregistrement de clé DKIM par le groupe délégué.

Si l'organisation cliente conserve la responsabilité pour toute l'administration DNS, la compagnie qui externalise peut générer une paire de clés, fournissant la clé publique à la compagnie cliente, qui l'enregistre alors dans le service d'interrogation en utilisant un unique sélecteur. La compagnie cliente conserve le contrôle sur l'utilisation de la clé déléguée parce qu'elle conserve la capacité de révoquer la clé à tout moment.

Si le client veut que le groupe délégué fasse l'administration du DNS, il peut avoir le nom de domaine qui est spécifié avec le sélecteur qui pointe sur le serveur DNS du fournisseur. Le fournisseur crée alors et maintient toutes les informations de signature DKIM pour ce sélecteur. Donc, le client ne peut pas fournir de contraintes sur la partie locale des adresses qui sont signées, mais il peut révoquer les droits de signature du fournisseur en retirant l'enregistrement de délégation DNS.

B.1.2 PDA et appareils similaires

Les PDA démontrent le besoin d'utiliser plusieurs clés par domaine. Supposons que John Doe veuille être capable d'envoyer des messages en utilisant son adresse de messagerie électronique d'entreprise, jdoe@exemple.com, et son appareil de messagerie n'a pas la capacité de faire une connexion de réseau privé virtuel (VPN, *Virtual Private Network*) au réseau d'entreprise, soit parce que l'appareil est limité, soit parce que il y a des restrictions qui sont appliquées par son fournisseur d'accès Internet. Si l'appareil est équipé avec une clé privée enregistrée pour jdoe@exemple.com par l'administrateur du domaine exemple.com et le logiciel approprié pour signer les messages, John pourrait signer le message sur l'appareil lui-même avant la transmission à travers le réseau sortant du fournisseur de service d'accès.

B.1.3 Utilisateurs en itinérance

Les utilisateurs en itinérance se trouvent souvent dans des circonstances où il est pratique ou nécessaire d'utiliser un serveur SMTP autre que leur serveur de rattachement ; des exemples sont des conférences et de nombreux hôtels. Dans ces circonstances, une signature ajoutée par le service de soumission va utiliser une identité qui est différente de celle du système de rattachement de l'utilisateur.

Idéalement, les utilisateurs en itinérance vont se reconnecter à leur serveur de rattachement en utilisant un VPN ou un serveur de soumission fonctionnant avec l'authentification SMTP sur l'accès 587. Si la signature peut être effectuée sur l'ordinateur portable de l'utilisateur en itinérance, il peut alors signer avant la soumission, bien que le risque d'une modification ultérieure soit élevé. Si aucune n'est possible, ces utilisateurs en itinérance ne vont pas être capables d'envoyer des messages signés en utilisant leur propre clé de domaine.

B.1.4 Soumission de message indépendante (kiosque)

Des services autonomes, tels que des kiosques itinérants et des services d'information fondés sur la Toile, n'ont pas de relation de service de messagerie électronique permanente avec l'utilisateur, mais les utilisateurs demandent occasionnellement que des messages soient envoyés en leur nom. Par exemple, un site de la Toile fournissant des nouvelles permet souvent au lecteur de transmettre une copie de l'article à un ami. Cela est normalement fait en utilisant la propre adresse de messagerie électronique du lecteur, pour indiquer qui est l'auteur. Ceci est parfois appelé le problème "Evite", nommé d'après le site du même nom qui permet à un utilisateur d'envoyer des invitations à des amis.

La façon courante de faire cela est de continue de mettre l'adresse de messagerie électronique du lecteur dans le champ d'en-tête From du message mais de mettre une adresse possédée par le site d'envoi du message dans le champ d'en-tête Sender. Le site d'envoi peut alors signer le message, en utilisant le domaine qui est dans le champ Sender. Cela fournit des informations utiles au site receveur, qui est capable de corréler le domaine signant avec le rôle initial de soumission de message.

Les sites receveurs souhaitent souvent fournir à leurs utilisateurs finaux des information sur les messages qui sont traitées de cette façon. Bien que l'efficacité réelle des différentes approches soit un sujet de recherche sur l'utilisation des facteurs humains, une technique utilisée est que le système qui vérifie réécrive le champ d'en-tête From pour indiquer que l'adresse a été vérifiée, par exemple : From: John Doe via news@news-site.exemple <jdoe@exemple.com>. (Noter qu'une telle réécriture va casser une signature, sauf si c'est fait après l'achèvement de la vérification.)

B.2 Autres scénarios de livraison

Les messages sont souvent reçus à une boîte aux lettres qui a une adresse différente de celle utilisée durant la soumission initiale. Dans ce cas, un mécanisme intermédiaire opère à l'adresse originellement utilisée, et il passe ensuite le message à la destination finale. Ce processus de médiation présente certains défis pour les signatures DKIM.

B.2.1 Adresses d'affinité

Les "adresses d'affinité" permettent à un utilisateur d'avoir une adresse de messagerie électronique qui reste stable, même quand l'utilisateur bouge parmi différents fournisseurs de messagerie. Elles sont normalement associées à des associations d'anciens élèves de collège, des organisations professionnelles, et des organisations récréatives avec lesquelles elles s'attendent à avoir des relations à long terme. Ces domaines assurent généralement la transmission des messages entrants, et ont souvent une application de la Toile associée qui authentifie l'utilisateur et permet de changer l'adresse de transmission. Cependant, ces services dépendent généralement d'utilisateurs qui envoient des messages sortants à travers leurs propres MTA de fournisseur de service. Donc, le message qui est signé avec le domaine de l'adresse d'affinité n'est pas signé par une entité qui est administrée par l'organisation qui possède ce domaine.

Avec DKIM, les domaines d'affinité pourraient utiliser l'application de la Toile pour permettre aux utilisateurs d'enregistrer des clés par utilisateur à utiliser pour signer les messages au nom de leur adresse d'affinité. L'utilisateur laisserait la moitié secrète de la paire de clé pour signer, et le domaine d'affinité publierait la moitié publique dans le DNS pour l'accès par les vérificateurs.

Ceci est une autre application qui tire parti du chiffrement au niveau utilisateur, et les domaines utilisés pour les adresses d'affinité auraient normalement un très grand nombre de clés de niveau utilisateur. Autrement, le domaine d'affinité pourrait traiter la messagerie sortante, opérant un agent de soumission de message qui authentifie les utilisateurs avant d'accepter et signer les messages pour eux. Cela dépend, bien sûr, de ce que le fournisseur de service de l'utilisateur ne bloque pas les accès TCP pertinents utilisés pour la soumission de messages.

B.2.2 Alias simple d'adresse (.forward)

Dans certains cas, il est permis à un receveur de configurer une adresse de messagerie électronique à causer une redirection automatique des messages de l'adresse originale à une autre, comme par l'utilisation d'un fichier Unix .forward. Dans ce cas, les messages sont normalement redirigés par le service de traitement de messages du domaine du receveur, sans modification, sauf l'ajout d'un champ d'en-tête Received au message et un changement dans l'enveloppe d'adresse de receveur. Dans ce cas, le receveur à la boîte aux lettre de l'adresse finale va probablement être capable de vérifier la signature originale car le contenu signé n'a pas changé, et DKIM est capable de valider la signature du message.

B.2.3 Listes de diffusion et renvoyeurs

Il y a une large gamme de comportements dans les services qui prennent livraison d'un message et ensuite le re-soumettent. Un exemple principal est avec les listes de diffusion (collectivement appelées des "transmetteurs" ci-dessous) allant de celles qui ne font pas de modification au message lui-même, autre que d'ajouter un champ d'en-tête Received et changer les informations d'enveloppe, à celles qui ajoutent des champs d'en-tête, changent le champ d'en-tête Subject, ajoutent du contenu au corps (normalement à la fin) ou reformattent le corps d'une manière ou d'une autre. Les simples produisent des messages qui sont assez similaires aux service d'alias automatiques. Les systèmes plus élaborés créent essentiellement un nouveau message.

Un transmetteur qui ne modifie pas le corps ou les champs d'en-tête signés d'un message va probablement maintenir la validité de la signature existante. Il pourrait aussi choisir d'ajouter sa propre signature au message.

Les transmetteurs qui modifient un message d'une façon qui pourrait rendre une signature existante invalide sont des candidats particulièrement bons pour ajouter leur propre signature (par exemple, mailing-list-name@exemple.net). Comme (re-)signer est prendre la responsabilité du contenu du message, ces transmetteurs signants vont probablement être sélectifs et transmettre ou resigner un message seulement si il est reçu avec une signature valide ou si ils ont quelque autre base pour savoir que le message n'est pas usurpé.

Une pratique courante parmi les systèmes qui sont principalement redistributeurs de messagerie est d'ajouter un champ d'en-tête Sender au message pour identifier l'adresse utilisée pour signer le message. Cette pratique va supprimer tout champ d'en-tête Sender préexistant comme exigé par la [RFC5322]. Le transmetteur applique un nouveau champ d'en-tête

DKIM-Signature avec la signature, la clé publique, et les informations relatives au transmetteur.

Voir dans la [RFC6377] des sujets en rapport supplémentaires et une discussion.

Appendice C. Création d'une clé publique (pour information)

La signature par défaut est un résumé SHA-256 signé avec RSA du message complet. Pour faciliter l'explication, la commande openssl est utilisée pour décrire le mécanisme par lequel les clés et signatures sont gérées. Une façon de générer une clé privée de 1024 bits, non chiffrée convenable pour DKIM est d'utiliser openssl comme ceci :

```
$ openssl genrsa -out rsa.private 1024
```

Pour une sécurité accrue, le paramètre "-passin" peut aussi être ajouté pour chiffrer la clé privée. L'utilisation de ce paramètre exige d'entrer un mot de passe pour plusieurs des étapes suivantes. Les serveurs peuvent préférer utiliser la prise en charge d'un chiffrement matériel.

L'étape "genrsa" résulte en un fichier rsa.private contenant les informations de clé similaire à :

```
-----DÉBUT DE CLÉ PRIVÉE RSA-----
MIICXwIBAAKBgQDwIRP/UC3SBsEmGqZ9ZJW3/DkMoGeLnQg1fWn7/zYtIxN2SnFC
jxOCKG9v3b4jYfcTNh5ijSsq631uBItLa7od+v/RtdC2UzJ1lWT947qR+Rcac2gb
to/NMqJ0fzfVjH4OuKhitdY9tf6mcwGjaNBcWToIMmPSPDdQPNUYckcQ2QIDAQAB
AoGBALmn+XwWk7akvkUlqb+dOxyLB9i5VBVfje89Teolwc9YJT36BGN/l4e0l6QX
/1//6DWUTB3KI6wFcm7TWJcxbS0tcKZX7FsJvUz1SbQnkS54DJck1EZO/BLa5ckJ
gAYIaqlA9C0ZwM6i58lLlPadX/rtHb7pWzeNcZHjKrjM461ZAkEA+itss2nRlmyO
n1/5yDyCluST4dQfO8kAB3toSEVc7DeFeDhnC1mZdjASZNvdHS4gbLIA1hUGEF9m
3hKsGUMMPwJBAPW5v/U+AWTADFCS22t72NUurgzeAbzb1HWMqO4y4+9Hpjk5wvL/
eVYizyuce3/fGke7aRYw/ADKygMJdW8H/OcCQQDz5OQb4j2QDpPZc0Nc4QlbvMsj
7p7otWRO5xRa6SzXqqV3+F0VpqvDmshEBkoCydaYwc2o6WQ5EBmExeV8124XAkEA
qZzGsIxVP+sEVRWZmW6KNFSdVUpk3qzK0Tz/WjQMe5z0UunY9Ax9/4PVhp/j61bf
eAYXunajbBSOLlx4D+TunwJBANkPI5S9iylsbLs6NkaMHV6k5ioHBBmgCak95JGX
GMot/L2x0IYyMLAz6oLWh2hm7zwtb0CgOrPo1ke44hFYnfc=
-----FIN DE CLÉ PRIVÉE RSA-----
```

Pour extraire le composant de clé publique de la clé privée, on utilise openssl comme ceci :

```
$ openssl rsa -in rsa.private -out rsa.public -pubout -outform PEM
```

Il en résulte le fichier rsa.public contenant les informations de clé similaires à :

```
-----DÉBUT DE CLÉ PUBLIQUE-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDwIRP/UC3SBsEmGqZ9ZJW3/DkM
oGeLnQg1fWn7/zYtIxN2SnFCjxOCKG9v3b4jYfcTNh5ijSsq631uBItLa7od+v/R
tdC2UzJ1lWT947qR+Rcac2gbto/NMqJ0fzfVjH4OuKhitdY9tf6mcwGjaNBcWToI
MmPSPDdQPNUYckcQ2QIDAQAB
-----FIN DE CLÉ PUBLIQUE-----
```

Ces données de clé publique (sans les étiquettes DÉBUT et FIN) sont placées dans le DNS:

```
$ORIGIN _domainkey.exemple.org.
brisbane IN  TXT  ("v=DKIM1; p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQ"
                  "KBgQDwIRP/UC3SBsEmGqZ9ZJW3/DkMoGeLnQg1fWn7/zYt"
                  "IxN2SnFCjxOCKG9v3b4jYfcTNh5ijSsq631uBItLa7od+v"
                  "/RtdC2UzJ1lWT947qR+Rcac2gbto/NMqJ0fzfVjH4OuKhi"
                  "tdY9tf6mcwGjaNBcWToIMmPSPDdQPNUYckcQ2QIDAQAB")
```

C.1 Compatibilité avec les enregistrements de clés de domaine

Les enregistrements de clé DKIM ont été conçus pour être rétro compatibles dans de nombreux cas avec les enregistrements de clé utilisés par les clés de domaine [RFC4870] (parfois appelées des "enregistrements de sélecteur" dans le contexte des clés de domaine). Une zone d'incompatibilité mérite une attention particulière. La valeur d'étiquette "g=" peut être utilisée dans des clés de domaine et des enregistrements de clé de la [RFC4871] pour fournir une plus fine granularité de la validité de l'enregistrement de clé à une partie locale spécifique. Une valeur "g=" nulle dans les clés de domaine est valide pour toutes les adresses dans le domaine. Ceci diffère de l'usage dans la spécification DKIM originale [RFC4871], où la valeur "g=" nulle n'est valide pour aucune adresse. En particulier, voir l'exemple d'enregistrement de clé publique au paragraphe 3.2.3 de la [RFC4870].

C.2 Compatibilité avec la RFC 4871

Bien que l'étiquette "g=" ait été déconseillée dans cette version de la spécification DKIM (et donc DOIT maintenant être ignorée) les signataires sont invités à ne pas inclure l'étiquette "g=" dans les enregistrements de clé parce que certains vérificateurs conformes à la [RFC4871] vont l'utiliser encore pendant une durée considérable.

Appendice D. Considérations de MUA (pour information)

Quand une signature DKIM est vérifiée, le système traitant rend parfois le résultat disponible au MUA de l'utilisateur receveur. Comment présenter ces informations aux utilisateurs d'une façon qui les aide est un sujet de recherche continuel sur les facteurs humains d'usage. La tendance est que le MUA souligne le SDID, en tentant de montrer à l'utilisateur l'identité qui revendique la responsabilité du message. Un MUA pourrait faire cela avec des indices visuels comme des graphiques, pourrait inclure l'adresse d'une autre manière, ou pourrait même réécrire l'adresse From originale en utilisant les informations vérifiées. Certains MUA pourraient indiquer quels champs d'en-tête ont été protégés par la signature DKIM validée. Cela pourrait être fait avec une indication positive sur les champs d'en-tête signés, avec une indication négative sur les champs d'en-tête non signés, en cachant visuellement les champs d'en-tête non signés, ou une combinaison de tout cela. Si un MUA utilise des indications visuelles pour les champs d'en-tête signés, le MUA a probablement besoin de veiller à ne pas afficher les champs d'en-tête non signés d'une façon qui pourrait être comprise par l'utilisateur final comme ayant été signés. Si le message a une étiquette "l=" dont la valeur ne s'étend pas jusqu'à la fin du message, le MUA pourrait aussi cacher ou marquer la portion du corps de message qui n'a pas été signée.

Les informations susmentionnées ne sont pas destinées à être exhaustives. Le MUA peut choisir de souligner, accentuer, cacher, ou autrement afficher toutes autres informations qui peuvent, de l'avis de l'auteur du MUA, être réputées importantes pour l'utilisateur final.

Appendice E. Changements par rapport à la RFC 4871

- Le résumé et l'introduction ont été précisés sur la base de l'expérience accumulée.
- Diverses références ont été mises à jour.
- Plusieurs errata ont été résolus (voir http://www.rfc-editor.org/) : 1376 appliqué, 1377 appliqué, 1378 appliqué, 1379 appliqué, 1380 appliqué, 1381 appliqué, 1382 appliqué, 1383 éliminé (ne s'applique plus), 1384 appliqué, 1386 appliqué, 1461 appliqué, 1487 appliqué, 1532 appliqué, 1596 appliqué.
- Ajout de la section d'introduction qui énumère les documents d'architecture pertinents.
- Ajout d'une section introductive qui discute brièvement la question de l'intégrité des données.
- Permet la tolérance d'une certaine dérive d'horloge.
- Abandon de l'étiquette "g=" dans les enregistrements de clé. Le rapport de mise en œuvre indique qu'elle n'est pas utilisée.
- Suppression de la note sur les caractères génériques dans le DNS.
- Suppression de l'avis spécifique de SMTP dans la plupart des endroits.
- Réduction de la liste (non normative) des contenus de signature recommandés, et révision du texte de ce paragraphe.
- Précision de l'algorithme de génération de signature en réécrivant son pseudo-code.
- De nombreux paragraphes de terminologie ajoutés, importés de la [RFC5672]. Aussi, on utilise ces termes dans tout le document (par exemple, SDID, AUID).
- Ajout de paragraphes qui spécifient les exigences d'entrée et de résultats. Les exigences d'entrée visent un souci de sécurité soulevé par le groupe de travail (voir aussi les nouveaux paragraphes des considérations sur la sécurité). Les

exigences de résultats sont importées de la [RFC5672].
- o Ajout d'un appendice discutant de la compatibilité avec les enregistrements de clés de domaine ([RFC4870]).
- o Référence à la [RFC5451] comme exemple de méthode de communication des résultats de vérification DKIM.
- o Suppression de l'avis sur de possibles utilisations de l'étiquette de signature "l=".
- o Mises à jour des registres de l'IANA.
- o Ajout de deux paragraphes des considérations sur la sécurité parlant des attaques de message mal formé.
- o Diverses corrections rédactionnelles.

Appendix F. Remerciements

La précédente version de DKIM [RFC4871] était éditée par Eric Allman, Jon Callas, Mark Delany, Miles Libbey, Jim Fenton, et Michael Thomas.

La présente spécification est le résultat d'un effort collaboratif étendu, incluant la participation de Russ Allbery, Edwin Aoki, Claus Assmann, Steve Atkins, Rob Austein, Fred Baker, Mark Baugher, Steve Bellovin, Nathaniel Borenstein, Dave Crocker, Michael Cudahy, Dennis Dayman, Jutta Degener, Frank Ellermann, Patrik Faeltstroem, Mark Fanto, Stephen Farrell, Duncan Findlay, Elliot Gillum, Olafur Gudmundsson, Phillip Hallam-Baker, Tony Hansen, Sam Hartman, Arvel Hathcock, Amir Herzberg, Paul Hoffman, Russ Housley, Craig Hughes, Cullen Jennings, Don Johnsen, Harry Katz, Murray S. Kucherawy, Barry Leiba, John Levine, Charles Lindsey, Simon Longsdale, David Margrave, Justin Mason, David Mayne, Thierry Moreau, Steve Murphy, Russell Nelson, Dave Oran, Doug Otis, Shamim Pirzada, Juan Altmayer Pizzorno, Sanjay Pol, Blake Ramsdell, Christian Renaud, Scott Renfro, Neil Rerup, Eric Rescorla, Dave Rossetti, Hector Santos, Jim Schaad, l'équipe Spamhaus.org, Malte S. Stretz, Robert Sanders, Rand Wacker, Sam Weiler, et Dan Wing.

Les clés de domaine antérieures sont la principale source d'où DKIM est dérivé. Plus d'informations sur les clés de domaine sont dans la [RFC4870].

Cette révision a reçu des contributions de Steve Atkins, Mark Delany, J.D. Falk, Jim Fenton, Michael Hammer, Barry Leiba, John Levine, Charles Lindsey, Jeff Macdonald, Franck Martin, Brett McDowell, Doug Otis, Bill Oxley, Hector Santos, Rolf Sonneveld, Michael Thomas, et Alessandro Vesely.

Adresse des auteurs

Dave Crocker
Brandenburg InternetWorking
675 Spruce Dr.
Sunnyvale, CA 94086
USA
mél : dcrocker@bbiw.net
URI : http://bbiw.net

Tony Hansen
AT&T Laboratories
200 Laurel Ave. South
Middletown, NJ 07748
USA
mél : tony+dkimsig@maillennium.att.com

Murray S. Kucherawy
Cloudmark
128 King St., 2nd Floor
San Francisco, CA 94107
USA
mél : msk@cloudmark.com