RFC 5863 : DomainKeys Identified Mail (DKIM) Development, Deployment and Operations

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 27 mai 2010

Date de publication du RFC : Mai 2010

https://www.bortzmeyer.org/5863.html

Le mécanisme de signature (et donc d'authentification) du courrier DKIM a été normalisé dans le RFC 4871 [1] puis dans le RFC 6376. Ce nouveau RFC fait le point sur les aspects **pratiques** du déploiement de DKIM. Il ne s'agit pas encore vraiment d'un retour d'expérience (DKIM n'est pas encore assez utilisé pour cela) mais plutôt d'une collection de points auxquels il faut prêter attention quand on développe du logiciel DKIM ou quand on déploie DKIM.

La section 2 considère le problème générale d'authentification : que garantit DKIM exactement ? La lutte anti-spam aujourd'hui est **unilatérale** : le destinataire accepte ou refuse selon **ses** critères. Ces critères sont arbitraires, souvent absurdes (comme la vérification de l'enregistrement PTR) et produisent aussi bien des faux positifs que des faux négatifs. DKIM tente d'introduire une notion de **confiance** entre deux parties qui essaient de communiquer. L'expéditeur signe ses messages et, après vérification de sa signature, il peut espérer une délivrance déterministe de ses messages. Bon, cet objectif est loin d'être réalisé, mais c'est l'idée (exprimée en des termes très abstraits, qui sont la marque de l'auteur Philip Hallam-Baker, qui n'est pas toujours facile à suivre).

Donc, DKIM permet d'identifier et d'authentifier une organisation responsable ("*Responsible Identifier*", dit le RFC). Celle-ci peut être précise et affecter une étiquette particulière à différents types de messages (par exemple, pour un vendeur en ligne, le courrier de confirmation des commandes peut etre étiqueté différemment du courrier publicitaire, ce dernier ayant probablement une moins bonne réputation). Bien sûr, cette authentification ne dit pas que le message est véridique, correct ou intéressant. Elle dit simplement que telle organisation en prend la responsabilité (et elle permet de

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc4871.txt

vérifier qu'il n'y a pas d'usurpation). Le dessin de la section 2.1 illustre la complexité des relations entre les différents acteurs.

À noter que, pour DKIM, « le message » inclus les en-têtes comme `From:` donc DKIM ne garantit pas du tout que le courrier prétendant venir de `faispaslemalin@elysee.fr` vient vraiment de l'Élysée. (Ce point va probablement dérouter beaucoup d'utilisateurs, cf. section 2.4, qui insiste bien sur le fait que l'identité DKIM peut n'avoir aucun rapport avec les identités existantes dans le message.)

Alors, justement, qui est responsable du message ? Ce point est couvert dans la section 2.2, choix des paramètres. Il en existe plusieurs dans DKIM (`s=`, `d=` et `i=`) et la confusion avait même nécessité la sortie d'un RFC de correction, le RFC 5672. Donc, pour résumer, c'est le paramètre `d=` qui est le seul obligatoire et qui doit donc être rempli. Voici un exemple d'un message envoyé sur une liste de diffusion, par un utilisateur du domaine `assysm.com` :

```
DKIM-Signature: v=1; a=rsa-sha1; c=relaxed; d=cru.fr;
        h=from:sender:reply-to:subject:date:message-id:to:cc:list-id:list-help:list-unsubscribe:list-subscri
        s=lists; i=smtp-fr-request@cru.fr;
        bh=uoq1oCgLlTqpdDX/iUbLy7J1Wic=;
        ...
```

On n'y voit pas du tout le domaine expéditeur (qui ne signe probablement pas avec DKIM) mais par contre le service de listes de diffusion du CRU <http://www.cru.fr/>, Universalistes, a signé et engagé sa réputation. Le `d=cru.fr` indique que le CRU est responsable, le `i=smtp-fr-request@cru.fr` (facultatif) indique une responsabilité plus précise, celle de la liste smtp-fr. Attention, le fait qu'elle ressemble à une adresse de courrier ne doit pas faire illusion, ce n'est pas forcément une adresse utilisable. Il faut résister à la tentation d'analyser ce paramètre, il peut être différent selon les services. Il faut le traiter comme opaque.

Justement, quel nom de domaine choisir pour mettre dans les paramètres ? Comme dans l'exemple précédent, le signeur n'est pas forcément l'auteur, il peut être un fournisseur ou un intermédiaire. La section 2.3 explore en détail la question du choix du nom de domaine. Ainsi, pour une compagnie qui envoie des messages de différents types, la méthode recommandée est d'avoir plusieurs noms (par exemple `transaction.example.com` pour les messages de confirmation de commande mais `newsletter.example.com` pour la publicité) mais pas trop. Un seul nom (`example.com`) ne permettrait pas de distinguer entre la propagande du service Communication et les messages sérieux. Trop de noms et la situation devient confuse et un service ne peut plus bénéficier de la réputation des autres.

Si un fournisseur (ici `example.net`) gère le courrier et signe pour plusieurs de ses clients, il est logique d'utiliser un nom par client (`bigbank.example.net`, `pornvendor.example.net`, `viagravendor.exampl` etc). Si l'un d'eux est un spammeur, cela n'affectera pas les autres. Une autre possibilité serait de les classer selon ce qu'ils ont payé (`free.example.net`, `business.example.net` et `firstclass.example.net`).

Après ces problèmes de haut niveau, la section 3 se consacre à une question récurrente dès que la cryptographie est en jeu : la gestion des clés. DKIM n'est pas une PKI classique car la distribution et la « certification » des clés sont toutes les deux faites par le DNS. L'existence d'une signature DKIM valide indique donc finalement que le gérant de la zone DNS en question a authentifié ce message. Mais cela suppose que les bonnes pratiques de sécurité et de cryptographie aient été utilisées partout. Par exemple, si le gérant de la zone DNS gère sa zone via une interface Web en PHP avec injection SQL <https://www.bortzmeyer.org/sql-injection.html> incluse, son authentification ne vaut plus grand'chose. Même chose s'il laisse trainer la clé privée DKIM n'importe où. La section 3.1 couvre ce dernier cas : protégez vos clés, utilisez un HSM si possible, n'utilisez **pas** de système de

séquestre, etc. À noter, bien que le RFC n'en parle pas, que le destinataire d'un message n'a aucun moyen de savoir si ces bonnes pratiques ont été suivies ou pas par l'expéditeur...

Ensuite, la distribution de la clé publique dans le DNS (section 3.2). Ici, le gérant de la zone doit s'assurer que la zone est raisonnablement sécurisée, et contient bien les bons enregistrements DKIM, ce qui est d'autant plus difficile que le serveur de courrier (qui signe avec DKIM) et le serveur DNS sont souvent gérés par des équipes distinctes. Les deux groupes doivent donc se coordonner soigneusement.

Le DNS n'étant lui-même pas sans failles, l'utilisation de DNSSEC (RFC 4034) est recommandée. Si on utilise les NSEC3 du RFC 5155, il ne faut pas choisir l'option "*opt-out*", qui permettrait l'insertion de clés pirates, par exemple pour des sélecteurs DKIM choisis par l'attaquant.

La complexité est souvent le pire ennemi de la sécurité, or DKIM dispose d'un mécanisme complexe pour permettre des signatures par utilisateur et non plus juste par domaine. Les précautions à prendre lors de l'utilisation de ce mécanisme figurent en section 3.3. Là encore, un des pièges est que le destinataire ne connait pas la politique de gestion des utilisateurs du domaine expéditeur et ne peut donc pas savoir si elle est laxiste ou au contraire très rigoureuse.

Le processus de signature lui-même est ensuite couvert dans la section 4. Le module de signature peut être placé à de nombreux endroits, le plus évident étant le MTA de sortie de l'organisation, ce qui simplifie le déploiement. Par contre, cela offre moins de souplesse, moins de possibilités de réglages que si la signature se fait dans les MUA des utilisateurs. Dans les deux cas, comme la politique de l'organisation va changer au cours du temps, le RFC recommande que les choix (par exemple, quel sélecteur utiliser) soient disponibles dans la configuration plutôt que placés en dur dans le code.

Et la vérification, à l'autre bout, lorsque le message est reçu ? Quels sont les pièges à éviter ? Par exemple, la section 5.1 insiste sur un point : un message dont la signature est invalide **doit** être traité comme s'il n'avait pas de signature du tout. Il ne doit pas être moins bien traité car la signature invalide peut être due à bien des problèmes, pas toujours sous le contrôle de l'émetteur. (En outre, bien que le RFC n'en parle pas, les risques d'erreur lors de la signature étant nombreux, traiter moins bien les messages avec une signature invalide découragerait les premiers adopteurs de DKIM, en leur faisant payer cher une erreur, qui les ferai mettre après ceux qui n'essaient même pas de signer.) Et le message à signature invalide ne doit pas être mieux traité que celui sans signature car, dans ce cas, les méchants mettraient simplement des signatures invalides dans leurs messages. Donc, le programme de vérification peut traiter plus favorablement les messages à signature valide mais il doit être neutre pour ceux à signature invalide.

Un autre piège, traité très vite et de manière très sommaire par le RFC, est que l'option `l=` permet de ne signer qu'une **partie** d'un message et qu'une attaque possible est donc de rajouter du texte à un tel message. Le RFC ne propose pas de solution. On pourrait imaginer un MUA qui n'affiche pas du tout la partie non couverte par la signature, ou bien qui l'affiche d'une manière particulière (en violet sur fond rouge ?).

Un autre piège, conséquences des risques cités plus haut sur la sécurité du DNS et de la clé privée, est qu'un destinataire oublie que la sécurité de DKIM dépend d'un certain nombre de facteurs... qui ne sont pas forcément présents (section 5.3). Par exemple, une signature valide sur un message qui ne l'est pas est possible si l'expéditeur s'est fait copier sa clé privée, ou si son hébergement DNS s'est fait pirater, ou tout simplement si une des procédures de justice privée qui abondent dans le monde des noms de domaine, comme l'UDRP, lui a fait perdre son nom de domaine.

Enfin (mais il existe d'autres pièges, mentionnés dans le RFC), l'existence d'intermédiaires pas toujours respectueux du message (par exemple les MLM) complique encore la vérification, et les conclusions qu'on peut en tirer.

Pour aider les vérificateurs, la section 6 se lance dans une taxonomie des signatures, selon les cas les plus courants. Ainsi, on peut trouver le cas le plus simple et le plus évident, la signature mono-domaine (section 6.1), où l'organisation signe ses propres messages avec son propre domaine. Ainsi, la société Toto, qui détient le domaine `toto.example`, aura à la fois des `From: USER@toto.example` et une signature en `d=toto.example`. Mais il existe aussi des cas plus compliqués comme le cas où la signature est faite par une tierce partie (section 6.3, voir aussi la 7.5), par exemple un « fournisseur de réputation » qui, en signant les messages, « garantirait » la valeur de ces messages. À noter que, pour DKIM, la signature du courrier `From: smith@company.example` par un `d=company.example` (signature mono-domaine) ou bien par `d=rep-provider.example` (signature par un tiers) sont exactement équivalentes : DKIM lui-même ne privilégie pas un cas plutôt qu'un autre.

Puisque plusieurs cas sont possibles, peut-on les combiner et avoir plusieurs signatures DKIM pour un message ? Oui, dit la section 6.5 qui rappelle que, non seulement c'est légal, mais que cela peut être parfaitement raisonnable dans certains cas (par exemple, signature par l'organisation émettrice **et** par un fournisseur de réputation).

Cela vaut la peine de le répéter : DKIM fournit uniquement un moyen de s'assurer qu'une organisation (identifiée par un nom de domaine) valide un message. DKIM est un mécanisme, pas une politique, et ne dit pas **qui** doit signer (comme l'indique la section 6, il y a plusieurs possibilités raisonnables) ni ce que le récepteur doit faire lorsqu'un message est signé et valide. La section 7 illustre cette « neutralité » de DKIM par plusieurs exemples d'usages, incluant la publication des politiques de signature par ADSP (RFC 5617).

La très ancienne expérience de la cryptographie dans l'Internet indique qu'il y aura certainement des problèmes avec DKIM. La section 8 cite quelques cas qui seront probablement délicats comme le cas d'une personne en voyage professionnel qui essaie d'envoyer du courrier en passant par le MTA du FAI de son hôtel, qui n'a pas la clé privée de la société de cette personne... Un problème similaire survient avec les services du type « Envoyez cette information à un ami » où le service qui propose cette fonction n'a pas de clé privée correspondant au domaine que l'utilisateur indiquera dans son adresse. DKIM ne fournit pas de solution à ce problème.

Ainsi, même les cas qui semblent simples et banals peuvent provoquer des surprises. La section 8.2 mentionne le problème de l'authentification du courrier d'une organisation, par cette même organisation. Surement, une organisation qui signe tout avec DKIM peut rejeter le courrier entrant qui prétend venir d'elle, mais qui n'a pas de signature ? Mais c'est plus compliqué que cela, en raison des programmes qui modifient les messages (et peuvent invalider une signature), ainsi qu'en raison des messages envoyés par le VRP itinérant cité plus haut.

Certains choix peuvent rendre l'utilisation de DKIM encore plus difficile. Par exemple, l'authentification par utilisateur, quoique parfois citée par certains enthousiastes partisans de DKIM, risque fort d'être peu praticable, pour les raisons expliquées en section 8.3 : il y a peu d'intérêt à maintenir une base de réputation par utilisateur (on peut imaginer un récepteur qui fasse confiance au courrier d'`amazon.com`, on a plus de difficulté à concevoir un récepteur qui fasse confiance à `smith@amazon.com` mais pas à `jones@amazon.com`), et le coût d'une telle signature par utilisateur (distribuer les clés, garantir leur sécurité, les mettre à jour) serait énorme.

Et les logiciels ? Après tout, ils sont en première ligne dans le déploiement de DKIM et plus d'une mesure de sécurité a été annulée par un mauvais logiciel. Les sections 8.4 et 8.5 rassemblent les analyses

sur le rôle desdits logiciels. Par exemple, l'usage des MUA pour signer est découragé, car les MUA ne sont pas en général sous le contrôle direct de l'organisation, contrairement aux MTA, et sont souvent situés sur des machines compromises.

Enfin, pour continuer dans les conseils pratiques, notons l'annexe A 3, qui détaille comment réaliser une migration DKIM propre, depuis un algorithme de signature DKIM (RSA est le seul normalisé dans le RFC 6376) vers un autre et l'annexe B qui rappelle les principes généraux de programmation d'une application cryptographique sûre. Par exemple, sur une machine à mémoire virtuelle, le programmeur devrait s'assurer que la clé privée, stockée en mémoire, ne se retrouve pas dans le "*swap*" à un endroit où d'autres processus pourraient la lire (pour cela, le programmeur peut, par exemple, utiliser `mlock()` sur Unix.) Notre RFC recommande fortement d'utiliser des bibliothèques cryptographiques existantes et éprouvées, plutôt que de se croire capable d'en inventer une nouvelle.

Le ricaneur notera que l'IETF ne s'est mise à utiliser son œuvre qu'en juillet 2011 <http://www.ietf.org/mail-archive/web/ietf/current/msg67811.html>. Désormais, les messages émis par les serveurs de courrier de l'IETF arborent fièrement la signature DKIM :

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/simple; d=ietf.org; s=ietf1;
        t=1311614276; bh=qLpIZcZ8XeP5xTrgVPRjnXlZjXWiz9DqXpACtarsL0Q=;
        h=Date:From:To:Subject:Message-ID:MIME-Version:List-Id:
        List-Unsubscribe:List-Archive:List-Post:List-Help:List-Subscribe:
        Content-Type:Content-Transfer-Encoding:Sender;
        b=ppseQobrat1rQ+Brsy2LSpMAA79YgaFJ7PK2EG1N4w0zS2IZBqDQiXYHJxG/wv4wl
        GOd42GtThBVxB5BmBhkTn8M1Rqz+ZhW2pLPlcI1zHcmLmJHLMt1wC6R3wiCi4bipVd
        CszNeb58HSYGNDQmvnW9dAxi38pL/kjunJTpmVT4=
```