RFC 7929 : DNS-Based Authentication of Named Entities (DANE) Bindings for OpenPGP

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 25 août 2016. Dernière mise à jour le 12 décembre 2017

Date de publication du RFC : Août 2016

https://www.bortzmeyer.org/7929.html

Un problème classique du système de cryptographie OpenPGP, normalisé dans le RFC 4880 [1] est de vérifier les clés publiques des correspondants. Les trouver, c'est relativement facile : le correspondant pense à vous les envoyer ou bien on se sert tout simplement d'un serveur de clés. Mais ceux-ci ne garantissent rien sur la clé. N'importe qui peut créer une clé marquée `flotus@whitehouse.gov` et la mettre sur les serveurs de clé, même si cette clé n'a rien à voir avec la Maison-Blanche. Avant la solution de ce nouveau RFC, il n'existait pas de mécanisme sécurisé pour récupérer une clé publique PGP. Que propose ce RFC ? De mettre les clés dans le DNS (nouveau type d'enregistrement `OPENPGPKEY`), dans le domaine de la partie droite de l'adresse de courrier, sécurisée par DNSSEC. En gros, il s'agit de faire pour le courrier et PGP ce que fait déjà DANE (RFC 6698) pour le Web/TLS.

Les serveurs de clés utilisent le protocole HKP (jamais décrit dans un RFC). Ils fournissent un service très utile en permettant de chercher une clé, par son identificateur, ou par l'adresse de courrier associé. Voici un exemple par identificateur :

```
% gpg --recv-key  0xF3396311465F8E5D
gpg: requesting key 465F8E5D from hkps server hkps.pool.sks-keyservers.net
gpg: key 465F8E5D: public key "Amaelle G <amaelle.guiton@techn0polis.net>" imported
...
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc4880.txt`

et un par adresse (notez que les adresses sont probablement toutes fausses) :

```
% gpg --search-key elysee.fr
gpg: searching for "elysee.fr" from hkps server hkps.pool.sks-keyservers.net
(1) hollande (flamby) <hollande@elysee.fr>
  2048 bit RSA key CF22758A, created: 2014-01-06
(2) jacques chirac (ancienne adresse) <jacques-chirac@elysee.fr>
  2048 bit RSA key 2A97F759, created: 2007-11-15, expires: 2007-11-16 (expired)
(3) kaziwan <kaziwan@elysee.gouv.fr>
  1024 bit DSA key AA7FD67C, created: 2005-11-28
(4) Gerard Calestroupat <gerard.calestroupat@elysee.fr>
  1024 bit DSA key 82F02C73, created: 2003-08-05, expires: 2003-08-08 (expired)
(5) Toto Berlingo <T.Berlingo@Elysee.fr>
  1024 bit DSA key E9E920B7, created: 1999-06-10
```

Ces serveurs n'offrent aucune garantie : n'importe qui peut y publier une clé, avec n'importe quelle adresse et certaines clés sont clairement mensongères. L'usage normal est de récupérer la clé et ses signatures, puis de vérifier les signatures. Si elles sont faites par des gens qu'on a validés (directement, ou bien transitivement, jusqu'à une certaine profondeur), on estime la clé correcte (c'est ce qu'on nomme le "*web of trust*"). Autrement, la clé ne vaut rien. En outre, le seul système de révocation est de signer une révocation avec sa clé privée : si on l'a perdue, on ne pourra jamais retirer la clé des serveurs de clé. Pour ces deux raisons (fausses clés, et clés devenues inutilisables), il est donc difficile d'utiliser automatiquement, depuis un MUA ou un MTA, ces serveurs.

La solution proposée dans ce RFC est, comme souvent aujourd'hui, d'utiliser le DNS, qui a montré sa fiabilité et son ubiquité. Tout le monde peut faire des requêtes DNS, même coincé derrière un pare-feu, et tout le monde peut les valider, grâce à DNSSEC (RFC 4035).

On va donc publier dans sa zone DNS des enregistrements de type `OPENPGPKEY`, indexés par la partie gauche de l'adresse de courrier (c'est un peu plus compliqué, car elle peut contenir des caractères qui sont spéciaux pour le DNS ; voir plus loin). Le correspondant qui veut envoyer du courrier à quelqu'un cherchera cet enregistrement dans le DNS, et, s'il en trouve un, le validera avec DNSSEC et récupérera ainsi une clé PGP relativement sûre. La révocation d'une clé se fait simplement en retirant l'enregistrement du DNS.

La solution de ce RFC rend envisageable de récupérer et de vérifier automatiquement une clé avant l'envoi d'un message. Mais le RFC note bien qu'elle ne remplace pas complètement le "*web of trust*", qui reste nécessaire si on veut une vérification sérieuse.

Ce RFC a le statut « Expérimental ». Il s'agit de tester l'idée et de voir si elle marche bien, et n'a pas trop d'inconvénients (par exemple de taille des zones DNS pour les domaines gérant beaucoup de comptes de courrier, surtout vu la taille des enregistrements `OPENPGPKEY`). Si le nombre de messages chiffrés avec OpenPGP augmente significativement suite à ce RFC, ce sera bon signe.

Notez qu'une expérience ressemblant à celle-ci avait déjà été faite avec le type d'enregistrement DNS `CERT` du RFC 4398. Ce fut un échec (peu de déploiement, peut-être en raison de la complexité du type `CERT`).

La section 2 de notre RFC décrit le format du nouveau type d'enregistrement DNS. Chaque enregistrement contient une et une seule clé. Si un utilisateur a plusieurs clés, il doit créer plusieurs enregistrements. Le type est 61 (et enregistré à l'IANA <https://www.iana.org/assignments/

dns-parameters/dns-parameters.xml#dns-parameters-4> depuis août 2014). La partie droite de l'enregistrement (les données) contient la clé et au moins un ID et une auto-signature. Les clés PGP complètes, avec des tas de signatures, peuvent être grosses, trop pour le DNS ; le RFC recommande de ne publier que des clés minimales (pas trop de signatures, par exemple, et évidemment pas les photos qu'on peut inclure dans un attribut de la clé, cf. RFC 4880, section 5.12.1). Avec GnuPG, regardez l'exportation de ma clé avec toutes ses méta-données, et en exportation minimale (l'annexe A du RFC décrit les commandes GnuPG à utiliser) :

```
%  gpg --export CCC66677 > key.pgp
% ls -lh key.pgp
-rw-r--r-- 1 bortzmeyer bortzmeyer 86K Aug 25 17:17 key.pgp

%  gpg --export --export-options export-minimal,no-export-attributes CCC66677 > key.pgp
% ls -lh key.pgp
-rw-r--r-- 1 bortzmeyer bortzmeyer 5.8K Aug 25 17:18 key.pgp
```

Le format utilisé est celui du RFC 4880, section 11.1. C'est donc du binaire qui circule sur le réseau (rappelez-vous bien que, dans le DNS, le format de présentation, celui des fichiers de zone, et de la sortie de dig, n'a rien à voir avec le format binaire utilisé sur le réseau.) Les formats « texte » d'OpenPGP (« "*ASCII armor*" ») ne sont pas utilisés sur le réseau. (Donc, avec GnuPG, pas d'option `--armor`.)

Le format de présentation (celui des fichiers de zone et de la sortie de dig) encode la clé en Base64.

Et la partie gauche de l'enregistrement DNS ? Quel est le nom de domaine utilisé ? La section 3 du RFC fixe les règles :

- Le domaine de l'adresse de courrier (partie droite de l'adresse de courrier) est celui où on met les enregistrements DNS `OPENPGPKEY`. La clé pour l'adresse `stephane+chose@trucmachin.example` sera donc dans la zone `trucmachin.example`.
- Le nom de domaine sera la concaténation d'un condensat de la partie gauche de l'adresse de courrier (`stephane+chose`, dans l'exemple ci-dessus), et du composant `_openpgpkey`, avec le domaine de l'adresse de courrier (`trucmachin.example` dans l'exemple ci-dessus). Le condensat est tronqué à 28 octets. (Le nom de domaine n'est pas utilisé dans le condensat, pour faciliter la vie des opérateurs qui proposent la même adresse dans différents domaines.)
- En fait, la règle est plus compliquée en raison des équivalences entre certains caractères (voir les exemples plus loin). Une correspondance est donc faite pour certains caractères. (Ce fut l'un des points les plus discutés dans le groupe de travail à l'IETF.)
- Par exemple, les guillemets (oui, `"jipoune le meilleur"@example.com` est une adresse de courrier légale) sont retirés.
- La condensation de la partie gauche de l'adresse de courrier est faite en SHA-256 (RFC 5754). Cela permet une protection limitée (cf. section 7.4) de la vie privée : même si un méchant met la main sur tout le fichier de zone, il ne trouvera pas facilement toutes les adresses (qui sont des données personnelles). Mais le but principal de cette condensation est de résoudre le problème de certains caractères qui sont permis dans la partie locale d'une adresse de courrier, mais qui posent des problèmes dans le DNS.

Ainsi, si l'adresse de l'utilisateur est `hugh@example.com`, la requête `OPENPGPKEY` devra chercher `c93f1e400f26708f98cb19d936620da35eec8f72e57f9eec01c1afd6._openpgpkey.example.com`. Voici comment calculer cela avec les outils du shell Unix (28 octets = 56 caractères dans la représentation en hexadécimal) :

```
% echo -n hugh | sha256sum | cut -c -56
c93f1e400f26708f98cb19d936620da35eec8f72e57f9eec01c1afd6
```

Une des difficultés pour trouver le bon nom de domaine est que les applications doivent traiter la partie gauche des adresses de courrier comme opaque (pas le droit d'analyser sa structure) et qu'elles ne connaissent pas les règles de canonicalisation qu'appliquera le domaine de destination, comme d'ignorer la casse de la partie locale (ce qui est souvent fait, mais pas toujours). Par exemple, Gmail ignore les points dans les adresses (donc `foobar@gmail.com` et `foo.bar@gmail.com` arrivent dans la même boîte aux lettres). L'émetteur qui ne connait pas cette règle va chercher la clé dans un domaine qui ne sera pas le bon. Idem avec les sous-adresses utilisées par certains domaines (en général avec le séparateur plus, comme `stephane+blog`, `stephane+ietf`, etc). Le RFC rappelle que l'émetteur ne peut pas se permettre de deviner ces règles locales, et qu'elles peuvent changer à tout moment. C'est au destinataire de se débrouiller, en publiant la clé à plusieurs noms, et en faisant attention aux variantes qu'il publie.

L'internationalisation des adresses de courrier complique évidemment encore un peu les choses (voir par exemple la section 10.1 du RFC 6530).

La section 6 du RFC se penche sur un problème pratique qu'on rencontre parfois avec le DNS : la difficulté à recevoir des réponses au-delà d'une certaine taille (il y a trois limites fréquemment rencontrées, la très ancienne limite de 512 octets du DNS, largement dépassée de nos jours, la limite de la MTU à 1 500 octets, au-delà de laquelle peut commencer la fragmentation, et la limite par défaut de la plupart des clients DNS à 4 096 octets). Les clés PGP peuvent être grosses, et le RFC recommende donc si possible de les récupérer sur TCP, pas UDP.

La section 7 de notre RFC analyse les questions de sécurité liées à cette technique. Elle rappelle que DNSSEC **doit** être utilisé : les enregistrements `OPENPGPKEY` récupérés ne doivent être utilisés que s'ils sont signés, et que la signature est valide. (Autrement, il serait trop facile à un attaquant de répondre avec une fausse clé.) Mais si DNSSEC est nécessaire, il n'est pas suffisant et la validation habituelle des clés PGP reste nécessaire si on veut un haut niveau de confidentialité. Ceci dit, comme souvent en sécurité, le mieux est l'ennemi du bien, et il vaut mieux une clé pas très vérifiée plutôt que d'envoyer le message en clair, comme le fait presque tout le monde aujourd'hui.

Et, évidemment, la sécurité DNSSEC doit être équivalente à la sécurité PGP puisqu'un attaquant qui aurait cassé la clé DNSSEC pourrait remplacer toutes les clés PGP du domaine. Il faut donc une cohérence dans les politiques de sécurité entre PGP et DNSSEC (section 7.6).

Autre problème de sécurité, cette fois lié à la vie privée : les requêtes DNS révèlent avec qui on veut communiquer de manière sécurisée par courrier (RFC 7626). Le fait que le nom de domaine utilisé soit un condensat de la partie locale de l'adresse de courrier limite un peu les risques, mais pas suffisamment (si on soupçonne qu'Alice écrit à `bob@example.com` mais qu'on n'en est pas sûr, il suffit de construire le nom où se trouve l'enregistrement `OPENPGPKEY` et de vérifier que ce nom est demandé, cf. section 7.4). C'est d'autant plus grave que les clients DNS actuels envoient en général le nom de domaine complet à **tous** les serveurs, même ceux qui n'en ont pas besoin. La minimisation de la requête (RFC 9156) limite ce problème. Le chiffrement des requêtes DNS (RFC 7858) peut faire le reste. Le cache du DNS limite un peu les risques et il est donc essentiel de ne pas faire une requête DNS externe à chaque fois qu'on envoie un message PGP à quelqu'un, cela ferait fuiter bien trop d'informations (section 7.5).

Pour limiter les risques qu'un attaquant récolte toutes les adresses de courrier du domaine, le RFC recommande de signer la zone en utilisant NSEC3 (RFC 5155).

À l'inverse de ce souci de protection de la vie privée, si une organisation veut lire le courrier de ses employés, la solution est qu'elle publie une clé d'organisation dans le DNS, pour pouvoir déchiffrer les messages entrants.

Un autre problème de sécurité est le risque d'utilisation dans des attaques par amplification <https://www.bortzmeyer.org/amplification-dns-combien.html>. La taille importante des enregistrements OPENPGPKEY (surtout avec les clés RSA) aggrave ce risque. Le RFC suggère de n'envoyer ces enregistrements via UDP que si l'adresse IP source de la requête a été vérifiée, par exemple avec les petits gâteaux du RFC 7873.

Où en sont les mises en œuvre de ce RFC ? GnuPG contient le code pour gérer ces clés dans le DNS depuis la version 2.1.9. Même chose pour openpgp-milter <http://github.com/letoams/openpgpkey-milter/>.

L'outil hash-slinger <http://people.redhat.com/pwouters/hash-slinger/> permet quant à lui de générer et de vérifier des enregistrements OPENPGPKEY :

```
% openpgpkey --fetch --uid paul@nohats.ca  paul@nohats.ca
-----BEGIN PGP PUBLIC KEY BLOCK-----
Comment: paul@nohats.ca key obtained from DNS
Comment: key transfer was protected by DNSSEC
Version: GnuPG v1

mQENBFaJkKsBCADDSwQawRsKYqY/DuxWZjNNn39f14tDaswbpuF+PorNnt0MrepI
0yVY28NQ+5P09j75Os1jlqksK06aAVBtkJvr+T1ip85AxPUdTjD3U3zhM5/YATMi
...
```

On peut alors enregistrer la clé dans le trousseau PGP :

```
% openpgpkey --fetch --uid paul@nohats.ca  paul@nohats.ca | gpg --import
gpg: key BBAE5D31: public key "Paul Wouters (online key) <paul@nohats.ca>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

Voici un exemple de récupération de ma clé :

```
% openpgpkey  --fetch --uid 'stephane@bortzmeyer.org' stephane@bortzmeyer.org |gpg

pub  4096R/CCC66677 2014-02-08 Stéphane Bortzmeyer (Main key) <stephane@bortzmeyer.org>
uid                            Stéphane Bortzmeyer <stephane@sources.org>
uid                            Stéphane Bortzmeyer (Work address) <bortzmeyer@nic.fr>
uid                            TextSecure fingerprint (05 d6 3b dc b7 e4 d7 69 2f f6 24 d5 51 31 88 2f a5 59 ae
sub  4096R/96A4A254 2014-02-09 [expires: 2018-01-10]
sub  4096R/57F02AA1 2014-02-09 [expires: 2017-01-10]
```

Mais comment ai-je fait pour que ça marche ? hash-slinger permet de créer la clé directement au bon format :

```
% openpgpkey --create stephane@bortzmeyer.org
; keyid: 555F5B15CCC66677
28182f0a278161989f90f090dabd6cab331663d8509ddbae617bb1e7._openpgpkey.bortzmeyer.org. IN OPENPGPKEY mQINBFL2VNABE
```

Il n'y a plus qu'à la mettre dans le fichier de zone, et à re-signer. Mais, car il y a un mais, cela ne marche que si on a des logiciels récents, qui connaissent le type 61 (`OPENPGPKEY`). Si ce n'est pas le cas, le signeur refusera de signer, ou le serveur de recharger la zone. C'était mon cas, en raison d'une trop vieille version d'OpenDNSSEC. Trois solutions, commençons par la plus simple, demander à hash-slinger de générer un enregistrement DNS à la syntaxe générique (≪ types inconnus ≫, du RFC 3597) :

```
% openpgpkey --create stephane@bortzmeyer.org --output generic
; keyid: 555F5B15CCC66677
28182f0a278161989f90f090dabd6cab331663d8509ddbae617bb1e7._openpgpkey.bortzmeyer.org. IN TYPE61 \# 5874 9902
```

Et c'est cet enregistrement à la syntaxe générique qu'on met dans le fichier de zone. Sinon, si on aime bien faire l'encodage soi-même, utilisons `xxd` :

```
% openpgpkey --create stephane@bortzmeyer.org > key.zone
[Edit to keep the zone data]

% base64 -d key.zone > key.bin
[wc -c key.bin to know what number to put in the zone file]

% xxd -p key.bin > key.hex
```

Et on met le contenu de `key.hex` dans le fichier de zone. Sinon, l'annexe A du RFC fournit une variante de cette solution, utilisant `hexdump`.

Voici la récupération de cette clé dans le DNS, avec un dig récent, qui connait ce type `OPENPGPKEY` et sait formater le résultat :

```
% dig OPENPGPKEY   28182f0a278161989f90f090dabd6cab331663d8509ddbae617bb1e7._openpgpkey.bortzmeyer.org
;; Truncated, retrying in TCP mode.
...
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36368
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 9, ADDITIONAL: 13
...
;; ANSWER SECTION:
28182f0a278161989f90f090dabd6cab331663d8509ddbae617bb1e7._openpgpkey.bortzmeyer.org. 85841 IN OPENPGPKEY ( m
ExJHaQ7LHPRVjAQtBiBN0vI3Uh0VgFzjA+0H2sqTduJY
tqd8mrTh9clDnCbrMU8svc7MeWxkW21ogjqBYL8puA3d
                                ...
```

Notez le ≪ "*Truncated, retrying in TCP mode*" ≫. L'enregistrement est trop gros pour les paquets UDP qu'accepte dig par défaut (il fait huit kilo-octets, dig accepte quatre par défaut). Notez aussi le bit AD ("*Authentic Data*") dans la réponse : celle-ci a bien été validée par DNSSEC.

Avec un dig ancien, qui ne connait pas ce nouveau type (et, cette fois, on demande directement en TCP, comme le recommande le RFC) :

```
% dig +tcp -t TYPE61 28182f0a278161989f90f090dabd6cab331663d8509ddbae617bb1e7._openpgpkey.bortzmeyer.org
...
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19989
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 9, ADDITIONAL: 29
...
;; ANSWER SECTION:
28182f0a278161989f90f090dabd6cab331663d8509ddbae617bb1e7._openpgpkey.bortzmeyer.org. 86206 IN TYPE61 \# 5874
674B1F33DE5F31D97EF8A4131247690ECB1CF4558C04
2D06204DD2F237521D15805CE303ED07DACA9376E258
B6A77C9AB4E1F5C9439C26EB314F2CBDCECC796C645B
                                ...
```

Sur ce sujet, vous pouvez aussi lire l'article de Johannes Weber <https://blog.webernetz.net/pgp-key-distribution-via-dnssec-openpgpkey/>, qui détaille une utilisation de l'outil de Shumon Huque <https://www.huque.com/bin/openpgpkey>.