RFC 8976 : Message Digest for DNS Zones

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 10 février 2021

Date de publication du RFC : Février 2021

https://www.bortzmeyer.org/8976.html

Ce nouveau RFC normalise un mécanisme pour ajouter aux zones DNS un condensat qui permet de vérifier leur intégrité. Pourquoi, alors qu'on a TSIG, TLS et SSH pour les transporter, et PGP et DNSSEC pour les signer ? Lisez jusqu'au bout, vous allez voir.

Le nouveau type d'enregistrement DNS créé par ce RFC se nomme ZONEMD et sa valeur est un condensat de la zone. L'idée est de permettre au destinataire d'une zone DNS de s'assurer qu'elle n'a pas été modifiée en route. Elle est complémentaire des différentes techniques de sécurité citées plus haut. Contrairement à DNSSEC, elle protège une zone, pas un enregistrement.

Revenons un peu sur la terminologie. Une **zone** (RFC 8499 [1], section 7) est un ensemble d'**enregistrements** servi par les mêmes serveurs. Ces serveurs reçoivent typiquement la zone depuis un maître, et utilisent souvent la technique AXFR (RFC 5936) pour cela. Ce transfert est souvent (mais pas toujours) protégé par le mécanisme TSIG (RFC 8945), qui permet de s'assurer que la zone est bien celle servie par le maître légitime. D'autres techniques de transfert peuvent être utilisées, avec leur propre mécanisme de sécurité. On peut par exemple utiliser rsync sur SSH. Les zones sont souvent stockées dans des fichiers de zone, du genre :

```
@ IN  SOA ns4.bortzmeyer.org. hostmaster.bortzmeyer.org. (
        2018110902
        7200
        3600
        604800
        43200 )

  IN  NS  ns4.bortzmeyer.org.
```

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc8499.txt

```
  IN  NS  ns2.bortzmeyer.org.
  IN  NS  ns1.bortzmeyer.org.
  IN  NS  puck.nether.net.

  IN MX 0 mail.bortzmeyer.org.
  IN TXT  "v=spf1 mx -all"

  IN  A 92.243.4.211
  IN AAAA 2001:4b98:dc0:41:216:3eff:fe27:3d3f

www  IN CNAME ayla.bortzmeyer.org.

sub IN NS toto.example.
```

Donc, les motivations de notre RFC sont d'abord le désir de pouvoir vérifier une zone après son transfert, indépendamment de la façon dont elle a été transférée. Une zone candidate évidente est la racine du DNS (cf. section 1.4.1), publiquement disponible, et que plusieurs résolveurs copient chez eux (RFC 8806). Ensuite, l'idée est de disposer d'un contrôle minimum (une somme de contrôle) pour détecter les modifications accidentelles (le rayon cosmique qui passe). Un exemple de modification accidentelle serait une troncation de la fin de la zone, d'habitude difficile à détecter puisque les fichiers de zone n'ont pas de marques de fin (contrairement à, par exemple, un fichier JSON). Notez que la « vraie » vérification nécessite DNSSEC. Si la zone n'est pas signée, le mécanisme décrit dans ce RFC ne fournit que la somme de contrôle : c'est suffisant contre les accidents, pas contre les attaques délibérées. Mais, de toute façon, tous les gens sérieux utilisent DNSSEC, non ?

Il existe de nombreuses techniques qui fournissent un service qui ressemble plus ou moins à un des deux services mentionnés ci-dessus. L'un des plus utilisés est certainement TSIG (RFC 8945). Beaucoup de transferts de zone (cf. RFC 5936) sont ainsi protégés par TSIG. Il a l'inconvénient de nécessiter un secret partagé entre serveur maître et serveur esclave. SIG(0) (RFC 2931) n'a pas cet inconvénient mais n'est quasiment jamais mis en œuvre. Toujours dans la catégories des protections du canal, on a TLS et l'IETF travaille sur un mécanisme de protection des transferts par TLS, mais on peut aussi imaginer d'utiliser simplement DoT (RFC 7858).

Ceci dit, toutes ces techniques de protection du canal ont le même défaut : une fois le transfert fait, elles ne servent plus à rien. Pas moyen de vérifier si le fichier est bien le fichier authentique. Ainsi, un serveur faisant autorité <https://www.bortzmeyer.org/serveur-dns-faisant-autorite.html> qui charge une zone à partir d'un fichier sur le disque ne peut pas vérifier que ce fichier n'a pas été modifié. Bien sûr, les protections fournies par le système de fichiers offrent certaines garanties, mais pas parfaites, par exemple si un programme a planté et laissé le fichier dans un état invalide (oui, ça s'est déjà produit <https://www.bortzmeyer.org/point-de-en-panne.html>). Bref, on préférerait une protection des données et pas seulement du canal.

Alors, pourquoi ne pas simplement utiliser DNSSEC, qui fournit effectivement cette sécurité des données ? Le problème est que dans une zone, seules les informations faisant autorité sont signées. Les délégations (vers un sous-domaine, comme `sub.bortzmeyer.org` dans l'exemple plus haut) et les colles <https://www.afnic.fr/observatoire-ressources/papier-expert/le-dns-ca-colle-ou-ca> (adresses IP des serveurs de noms qui sont dans la zone qu'ils servent) ne sont pas signés. Pour une zone comme la racine, où il n'y a quasiment que des délégations, ce serait un problème. Et ce serait encore pire avec les zones qui utilisent NSEC3 (RFC 5155) avec "*opt-out*" puisqu'on ne sait même plus si un nom non-signé existe ou pas. Donc, DNSSEC est très bien, mais ne suffit pas pour notre cahier des charges.

On a d'autres outils pour la sécurité des données, le plus évident étant PGP (RFC 9580). D'ailleurs, ça tombe bien, la racine des noms de domaine est déjà distribuée avec une signature PGP :

```
% wget -q https://www.internic.net/domain/root.zone
% wget -q https://www.internic.net/domain/root.zone.sig
% gpg --verify root.zone.sig root.zone
gpg: Signature made Wed Jan 13 08:22:50 2021 CET
gpg:                using DSA key 937BB869E3A238C5
gpg: Good signature from "Registry Administrator <nstld@verisign-grs.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: F0CB 1A32 6BDF 3F3E FA3A  01FA 937B B869 E3A2 38C5
```

Notez que Verisign utilise une signature détachée du fichier. Cela permet que le fichier reste un fichier de zone normal et chargeable, mais cela casse le lien entre la signature et le fichier. Bref, cela ne résout pas complètement le problème. (Les historiens et historiennes s'amuseront de noter qu'une autre solution, très proche de celle de notre RFC, figurait déjà dans le RFC 2065, mais qu'elle n'avait eu aucun succès.)

Bon, passons maintenant à la solution (section 1.3). On l'a dit, le principe est d'avoir un enregistrement de type `ZONEMD` qui contient un condensat de la zone. Il est généré par le responsable de la zone et quiconque a téléchargé la zone, par quelque moyen que ce soit, peut vérifier qu'il correspond bien à la zone. En prime, si la zone est signée avec DNSSEC, on peut vérifier que ce `ZONEMD` est authentique.

Du fait que ce condensat couvre l'intégralité de la zone, il faut le recalculer entièrement si la zone change, si peu que ce soit. Cette solution ne convient donc pas aux grosses zones très dynamiques (comme `.fr`). Dans le futur, le `ZONEMD` est suffisamment extensible pour que des solutions à ce problème lui soient ajoutées. En attendant, `ZONEMD` convient mieux pour des zones comme la racine (rappel, elle est disponible en ligne <`http://www.internic.net/domain/`>), ou comme la zone d'une organisation. La racine est un cas particulièrement intéressant car elle est servie par un grand nombre de serveurs <`https://www.bortzmeyer.org/combien-serveurs-racines.html`>, gérés par des organisations différentes. Sans compter les gens qui la récupèrement localement (RFC 8806). Le contrôle de son intégrité est donc crucial. Il y a une discussion en ce moment au sein du RZERC <`https://www.icann.org/rzerc`> pour proposer l'ajout de `ZONEMD` dans la racine.

Pour la zone d'une organisation, notre RFC rappelle qu'il est recommandé d'avoir une diversité des serveurs de nom, afin d'éviter les SPOF et que c'est donc une bonne idée d'avoir des serveurs esclaves dans d'autres organisations. Comme cette diversité peut entrainer des risques (le serveur esclave est-il vraiment honnête ? le transfert s'est-il bien passé ?), là aussi, la vérification de l'intégrité s'impose. Autre scénario d'usage, les distributions de fichiers de zone, comme le fait l'ICANN avec CZDS <`https://czds.icann.org/`> ou comme le fait le registre du `.ch` (téléchargez ici <`https://www.switch.ch/fr/open-data/#tab-c5442a19-67cf-11e8-9cf6-5254009dc73c-3`>). D'une manière générale, ce contrôle supplémentaire ne peut pas faire de mal.

Plongeons maintenant dans les détails techniques avec la section 2 du RFC, qui explique le type d'enregistrement `ZONEMD` (code 63 <`https://www.iana.org/assignments/dns-parameters/dns-parameters.xml#dns-parameters-4`>). Il se trouve forcément à l'apex de la zone. Il comprend quatre champs :

— Numéro de série, il permet de s'assurer qu'il couvre bien la zone qu'on veut vérifier (dont le numéro de série est dans le SOA).
— Plan, la méthode utilisée pour génerer le condensat. Pour l'instant, un seul plan est normalisé <`https://www.iana.org/assignments/dns-parameters/dns-parameters.xml#zonemd-schemes`>, nommé Simple, il fonctionne par un calcul sur l'entièreté de la zone. On peut ajouter des plans au registre, selon la procédure « Spécification nécessaire ».

— Algorithme, est l'algorithme de condensation utilisé, SHA-384 ou SHA-512 à l'heure actuelle <https://www.iana.org/assignments/dns-parameters/dns-parameters.xml#zonemd-hash-a mais on pourra en ajouter d'autres au registre (« Spécification nécessaire », là encore).
— Et le condensat lui-même.

Question présentation dans les fichiers de zone, ça ressemble à :

```
example.com. 86400 IN ZONEMD 2018031500 1 1 (
       FEBE3D4CE2EC2FFA4BA99D46CD69D6D29711E55217057BEE
       7EB1A7B641A47BA7FED2DD5B97AE499FAFA4F22C6BD647DE )
```

Ici, le numéro de série est 2018031500, le plan 1 (Simple) et l'algorithme de condensation SHA-384.

Il peut y avoir plusieurs ZONEMD dans un fichier, ils doivent avoir des couples {Plan, Algorithme} différents. (Le but est de fournir plusieurs techniques de vérification, et de permettre de passer en souplesse de l'une à l'autre.)

Comment exactement se fait le calcul du condensat ? La section 3 le détaille. D'abord, on supprime de la zone les éventuels ZONEMD existants, et on place un ZONEMD bidon (il en faut un, en cas de signature DNSSEC, pour éviter de casser la signature). Si on a DNSSEC, on signe alors la zone (et il faudra re-faire le RRSIG du ZONEMD après), on canonicalise la zone (le condensat est calculé sur le format « sur le câble » pas sur le format texte), et on calcule ensuite le condensat sur la concaténation des enregistrements (à l'exclusion du ZONEMD bidon, qui avait été mis pour DNSSEC, et de sa signature). On génère ensuite le vrai ZONEMD, puis, si on a DNSSEC, on recalcule sa signature. Autant dire que vous ne le ferez pas à la main ! L'endroit logique pour faire ce calcul du ZONEMD est dans le logiciel qui fait les signatures DNSSEC. Notez que, contrairement à DNSSEC, tous les enregistrements sont utilisés, y compris ceux qui ne font pas autorité comme les délégations et les colles.

Un mot sur la canonicalisation. Tout calcul d'un condensat cryptographique doit se faire sur des données canonicalisées, c'est-à-dire mises sous une forme canonique, une forme unique et standardisée. Comme le changement d'un seul bit change complètement le condensat, il ne faut pas laisser des petites variations de détail (par exemple, pour le DNS, entre noms comprimés et non-comprimés) à l'imagination de chaque programmeur. Voilà pourquoi notre RFC spécifie rigoureusement une forme canonique (par exemple, les noms ne doivent pas être comprimés).

La vérification se fait de la même façon (section 4 du RFC). On vérifie que le numéro de série est le même, on canonicalise, on concatène, on condense et on vérifie qu'on trouve bien ce qui était dans le ZONEMD. (S'il y a plusieurs ZONEMD, le premier qui marche suffit.) Le mieux est évidemment de tout valider avec DNSSEC, en plus.

La section 6 de notre RFC revient sur la sécurité du ZONEMD, ce qu'il garantit et ce qu'il ne garantit pas. D'abord, le point le plus important : sans DNSSEC, ZONEMD n'est qu'une somme de contrôle, il garantit l'intégrité mais pas l'authenticité, et il ne protège donc que contre les modifications accidentelles (ce qui est déjà très bien !) Le "*bit flipping*" <https://www.bortzmeyer.org/bitsquatting.html> sera détecté mais pas l'attaque délibérée et soignée. Ainsi, sans DNSSEC, un attaquant n'aurait, par exemple, qu'à retirer l'enregistrement ZONEMD de la zone avant de faire ses modifications (une attaque par repli).

D'autre part, les algorithmes de cryptographie vieillissent (RFC 7696) et ZONEMD ne peut donc pas garantir de sécurité sur le long terme. D'ailleurs, sur le long terme, on n'aurait probablement plus les clés DNSSEC disponibles.

Plus drôle, ZONEMD permet, comme toute technique de sécurité, de nouveaux modes de déni de service, comme l'a déjà expérimenté tout utilisateur de la cryptographie. Comme le condensat est strictement binaire (il colle aux données, ou pas du tout), la zone entière peut être considérée comme invalide si un problème modifie ne serait-ce qu'un seul bit. Le souci de l'intégrité de la zone et celui de sa disponibilité sont ici en conflit, et le responsable de la zone doit choisir.

Si vous aimez les comparatifs de performance, la section 7 du RFC discute de chiffres. Par exemple, une zone de 300 000 enregistrements a été ZONEMDifiée en deux secondes et demi sur un serveur ordinaire. C'est encore trop lent pour être utilisable sur des zones comme .com mais cela montre que des grandes zones qui ne changent pas trop souvent peuvent être protégées.

Et question mises en œuvre, ça se passe comment ? J'ai testé l'excellente bibliothèque ldns <https://nlnetlabs.nl/projects/ldns/>. Le code avec calcul de ZONEMD devrait être publié dans une version officielle en 2021 mais, en attendant, c'est déjà dans git, en . Une fois compilée, on peut fabriquer le ZONEMD (ici, avec la zone d'exemple montrée au début de cet article). On génère une clé, puis on signe, avec calcul de ZONEMD :

```
%  ./ldns-keygen -a ED25519 bortzmeyer.org
Kbortzmeyer.org.+015+00990

% ./ldns-signzone -o bortzmeyer.org -z 1:1  bortzmeyer-org.zone  Kbortzmeyer.org.+015+00990

% cat bortzmeyer-org.zone.signed
...
bortzmeyer.org. 3600 IN ZONEMD 2018110902 1 1 b1af60c3c3e88502746cf831d2c64a5399c1e3c951136d79e63db2ea99898ba9f2
bortzmeyer.org. 3600 IN RRSIG ZONEMD 15 2 3600 20210215174229 20210118174229 990 bortzmeyer.org. FJ98//KDBF/iMl1
```

(L'argument 1:1 signifie plan Simple et algorithme SHA-384.) On peut ensuite vérifier la zone. Si elle a un enregistrement ZONEMD, il est inclus dans la vérification :

```
% ./ldns-verify-zone  bortzmeyer-org.zone.signed
Zone is verified and complete
```

Si je modifie une délégation (qui n'est pas protégée par DNSSEC mais qui l'est par ZONEMD) :

```
% ./ldns-verify-zone  bortzmeyer-org.zone.signed
There were errors in the zone
```

Il y a aussi les dns-tools <https://github.com/niclabs/dns-tools> des gens du .cl. Pour BIND, c'est en cours de réflexion <https://gitlab.isc.org/isc-projects/bind9/-/issues/2099>. Apparemment, Unbound y travaille également. Voir aussi l'exposé de Verisign <https://indico.dns-oarc.net/event/43/contributions/920/attachments/882/1626/zonemd.pdf> à la réunion OARC de juillet 2022. Des tests sont en cours <http://zonemd-testing.verisignlabs.com/> (septembre 2022).

Depuis, un bon article de Jan-Piet Mens <https://jpmens.net/2023/04/16/the-zonemd-message-digest-r> a expliqué le fonctionnement pratique de ZONEMD, et les outils utilisables.