RFC 6781 : DNSSEC Operational Practices, Version 2

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 22 décembre 2012

Date de publication du RFC : Décembre 2012

https://www.bortzmeyer.org/6781.html

Comme avec toute technique fondée sur la cryptographie, le protocole DNSSEC impose, non seulement des bons algorithmes et une mise en œuvre correcte, mais surtout des procédures rigoureuses et soigneusement exécutées. C'est le but de ce RFC, qui remplace le RFC 4641 [1] et qui explique tout ce à quoi doivent s'attendre les registres et autres administrateurs de zones DNS, grandes ou petites, qui déploieraient DNSSEC. (Le cas des serveurs DNS récursifs, par exemple les résolveurs d'un FAI, n'est pas étudié.)

Notre RFC rappelle donc des concepts de base du DNS (notamment le fait que la réjuvénation <https://www.bortzmeyer.org/dns-propagation.html> des modifications n'est pas instantanée) puis rappelle les différentes clés (cf. section 1.1 pour une définition précise de clé) utilisées par DNSSEC et leurs caractéristiques souhaitables (longueur, période maximale pendant laquelle on les utilise, lieu de stockages, etc). Tout n'est pas encore bien connu dans ce domaine. Certes, la racine est signée depuis juillet 2010, ainsi que plus de 80 TLD mais, au niveau en dessous du TLD, peu de zones sont signées (dans les 1 % de .fr, par exemple) et l'expérience opérationnelle peut donc encore être améliorée.

Il explique ensuite les considérations temporelles (DNSSEC utilise le temps et nécessite des horloges bien synchronisées, par exemple par NTP). Le DNS étant hiérarchique, il faut veiller, lors de toutes les manipulations, à bien rester synchronisé avec le gérant de la zone parente, dont les enregistrements de type DS ("*delegation signer*") pointeront vers notre clé. Le but est, qu'à tout moment, une **chaîne de confiance** intacte aille de la clé de confiance jusqu'aux enregistrements du domaine. Si un lien de cette chaîne casse, le domaine sera marqué comme "*bogus*" (RFC 4033, section 5) et rejeté par les résolveurs validants.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc4641.txt

Enfin, le RFC étudie le "*rollover*", le remplacement d'une clé. Les clés ne pouvant pas raisonnablement être utilisées éternellement, il faut prévoir à l'avance les remplacements périodiques et aussi, hélas les remplacements en urgence en cas de compromission. Il faut apporter beaucoup de soin à ce remplacement, si on veut éviter que, pendant une certaine période, les données publiées dans le DNS soient invalides et donc rejetées par un résolveur DNS paranoïaque (il faut publier la nouvelle clé suffisamment à l'avance pour qu'elle soit présente partout ou bien signer tous les enregistrements avec les deux clés, l'ancienne et la nouvelle). Mon article à la conférence SATIN en 2011 <https://www.bortzmeyer.org/satin.html> avait montré que c'était loin d'être le cas : les erreurs sont fréquentes, même pour les grandes zones sérieuses.

Bref, pour un gérant de zone DNS, déployer DNSSEC, ce n'est pas uniquement signer la zone : c'est aussi mettre en place des procédures de sécurité, analogues à celle d'une autorité de certification.

Maintenant, avec la section 3, voyons les détails pratiques. D'abord, la génération et le stockage des clés cryptographiques. D'abord, un message d'espoir : en lisant ce RFC, on peut avoir l'impression d'un ensemble de tâches très compliquées, impossible à réaliser correctement. Heureusement, il existe déjà des logiciels qui, en automatisant la plupart de ces tâches, rendent les choses bien plus faciles. J'utilise pour cela OpenDNSSEC <https://www.bortzmeyer.org/opendnssec-debut.html>.

D'abord, avant de se lancer, il faut faire certains choix technico-politiques :

— Est-ce qu'on n'utilise qu'un type de clé ou bien est-ce qu'on sépare KSK ("*Key Signing Key*") et ZSK ("*Zone Signing Key*") ? Contrairement à ce qu'on lit souvent, la dichotomie entre une KSK et une ZSK n'a rien d'obligatoire, et elle complique les choses.
— Est-ce que les KSK vont être configurés par certains comme clés de confiance, à partir desquels on valide ? Cela permet d'être indépendant des autres acteurs comme la racine mais cela augmente les responsabilités.
— Quels sont les délais pertinents, par exemple le champ `Expire` du SOA, le temps de réaction souhaité en cas de problème ? Il faut aussi se demander si `NOTIFY` (RFC 1996) marche bien (ou si on ne peut compter que sur le paramètre `Refresh` de l'enregistrement SOA) et si on utilise `IXFR` (RFC 1995) ou `AXFR`,
— Quels sont les choix cryptographiques (par exemple, concernant la longueur des clés RSA) ?

Une fois ces choix faits, on pourra configurer le résultat (dans OpenDNSSEC, fichier `/etc/opendnssec/kasp.xml`). Mais que choisir ?

Pour le choix entre une séparation KSK/ZSK et une clé unique, le choix est purement un problème opérationnel du côté du gestionnaire de zone. Les résolveurs ne feront pas de différence.

Un problème à garder en tête pour le stockage des clés est le risque de compromission de la clé (vol du disque sur lequel se trouve la partie privée de la clé, par exemple). Si la clé privée est sur le disque d'une machine connectée à l'Internet, le risque est plus élevé que s'il est sur une machine déconnectée et enfermée dans un coffre-fort. Encore mieux, pour les plus riches, l'usage des HSM permet d'être raisonnablement sûr qu'il n'y aura pas de vol de la clé. D'un autre côté, pour signer des enregistrements, une clé sur une machine non-connectée est certainement moins pratique. Une des façons de résoudre le dilemne est de séparer KSK et ZSK : on met la KSK en sécurité dans un endroit bien protégé (elle ne sert qu'à signer la ZSK, qui ne change pas tous les jours) et on fait les signatures des enregistrements de la zone avec la ZSK. En cas de compromission de la ZSK, son remplacement est relativement simple, puisqu'il est entièrement interne à la zone, il ne nécessite pas d'interaction avec la zone parente. (C'est ainsi, par exemple, que fonctionne la racine.)

Comme toujours en sécurité,il y a un fort risque de blinder la porte en oubliant la fenêtre : il ne sert pas à grand'chose de protéger la KSK avec des précautions de niveau militaire si les données de la zone, celles qu'on va signer, peuvent être facilement modifiées par un attaquant (encore que le RFC oublie de

dire que la possession de la clé privée permette de générer des enregistrements mensongers sans que le registre s'en aperçoive). De même, le HSM empêche certes le méchant de copier la clé privée, mais il ne l'empêche pas de signer : si le pirate prend le contrôle de la machine connectée au HSM, il peut faire signer à ce dernier ce qu'il veut. Bref, il n'y a pas de solution miracle en matière de sécurité.

Séparer KSK et ZSK permet donc plus de souplesse dans la gestion de la sécurité. Mais cela rend les choses plus complexes. Si les clés sont gérés automatiquement par un outil comme OpenDNSSEC, ce n'est pas trop un problème. Sinon, il vaut peut-être mieux n'avoir qu'une clé.

Une question qui agite les milieux DNSSEC depuis de nombreuses années est celle du remplacement ("*rollover*") des clés. Il y a deux écoles <https://www.bortzmeyer.org/remplacement-cles.html> :

— Celle qui dit que les remplacements doivent être systématiques et fréquents, pour qu'ils deviennent de la routine ; elle se sépare en deux sous-écoles, une qui favorise les remplacements périodiques et une qui préfère les faire au hasard,
— Et l'école qui dit que les remplacements ne doivent se faire que s'il y a une bonne raison, par exemple une forte suspicion que la clé privée a été copiée.

Notre RFC note qu'il n'y a pas consensus sur cette question. Il note qu'un facteur important est « est-ce que la clé est utilisée comme clé de confiance quelque part ? », c'est-à-dire est-ce que des gens l'ont configuré dans leur résolveur validant (directives `trusted-keys` ou `managed-keys` de BIND et `trust-anchor` ou `trust-anchor-file` de Unbound) ? Cela peut se faire, par exemple pour éviter de dépendre de la hiérarchie du DNS (on peut imaginer une clé DNSSEC de `gouv.fr` installée dans les résolveurs des ministères pour être sûr de ne pas être affectés par des problèmes à l'AFNIC ou à la racine). Notez que cela peut arriver sans le consentement du gérant de la zone. Si ce dernier publie un DPS ("*DNSSEC Practice Statement*") disant en gros caractères « n'utilisez pas notre KSK comme clé de confiance, ne vous fiez qu'au DS de la zone parente », alors, on peut tenir pour acquis qu'il y aura peu de cas où la KSK sera installée en dur dans les résolveurs. Autrement, le remplacement devient bien plus compliqué car il faut prévenir tous les gens qui ont configuré cette clé de confiance. Certains ne seront pas joignables ou ne réagiront pas et il faudra donc prendre le risque de casser leur accès DNSSEC. Ainsi, pour la racine (qui ne peut pas compter sur l'enregistrement DS de la zone parente) qui est forcément clé de confiance, on peut penser qu'il n'y aura jamais de remplacement de la KSK originelle, la 19036, sauf en cas de compromission : cela nécessiterait de changer trop de résolveurs.

Notez qu'il existe un mécanisme pour mettre à jour les clés de confiance, décrit dans le RFC 5011. Comme il n'est jamais sûr que tous les « clients » le mettent en œuvre, il est difficile de savoir si on peut compter dessus (dans Unbound, `trust-anchor-file: "/etc/unbound/root.key"` installe une clé de confiance statique et `auto-trust-anchor-file: "/var/lib/unbound/root.key"` - notez le `auto-` devant - installe une clé qui pourra être mise à jour par les procédures du RFC 5011). Ces procédures n'ont jamais été testées pour la racine.

Si on sépare KSK et ZSK, c'est normalement le bit SEP ("*Secure Entry Point*", cf. RFC 3757) qui les distingue (dans le cas le plus courant, la ZSK aura un champ `Flags` de 256 et la KSK de 257 à cause du bit SEP). Ce bit n'est pas utilisé par les validateurs mais certains outils s'en servent pour trouver la KSK (par exemple le `dnssec-signzone` de BIND). Si on n'a qu'une seule clé, elle sert de KSK et doit donc avoir le bit SEP (autrement, le `dnssec-signzone` de BIND vous dira `dnssec-signzone: fatal: No self signed KSK's found`). Notez que certains outils ne vous laisseront pas facilement n'utiliser qu'une clé. Par exemple, avec `dnssec-signzone`, vous aurez le message `dnssec-signzone: fatal: No non-KSK DNSKEY found; supply a ZSK or use '-z'.` et vous devrez utiliser l'option indiquée pour signer quand même.

Un peu de cryptographie, maintenant. La durée de vie raisonnable d'une clé dépend de sa longueur. Le RFC suggère de décider d'une durée de vie, en tenant compte des contraintes opérationnelles, et d'en

déduire la longueur. Une KSK qui a des enregistrements DS dans sa zone parente peut durer vingt ans sans problème, sauf si on désire tester régulièrement les procédures de remplacement ou sauf si la clé privée est compromise. Si on veut la remplacer régulièrement, une durée raisonnable proposée par le RFC est d'un an.

Quel algorithme cryptographique choisir ? Les quatre standardisés à l'heure actuelle (le RFC en cite trois mais ECDSA est apparu depuis), RSA, DSA et GOST sont bien connus et spécifiés et considérés comme fiables. En partie pour des raisons de performance, le RFC suggère la combinaison RSA/SHA-256 comme l'algorithme préféré (RFC 5702) et RSA/SHA-1 sinon.

Les algorithmes fondés sur les courbes elliptiques (GOST - RFC 5933 - et ECDSA - RFC 6605) ont des avantages importants sur RSA, notamment les clés et signatures plus petites. Mais très peu de résolveurs validant les gèrent et signer une zone uniquement avec ces algorithmes serait donc peu utile. D'autre part, même si le RFC ne le dit qu'en passant, les problèmes de brevet sont bien plus nombreux que pour RSA.

Si on a choisi RSA, il faut se décider pour une longueur de clé, suffisante pour résister à la cryptanalyse pendant la durée de vie de la clé (le RFC 3766 contient des calculs intéressants sur la solidité des clés). Comme le note le RFC, en dépit d'un FUD important, personne n'a encore annoncé la cryptanalyse d'une clé RSA de 1024 bits (le record publié est aux alentours de 700 bits). Et, le jour où cela arrivera, ce ne sera pas la fin de RSA mais la fin d'une seule clé. Bien sûr, la cryptanalyse progresse (et, le RFC oublie ce point, les meilleurs résultats ne sont pas forcément publiés) mais le RFC estime que les clés de 1024 bit sont encore sûres pour dix ans.

Néanmoins, utiliser des clés plus grandes est raisonnable si on veut une très longue durée de vie ou bien si on estime que le remplacement de la clé sera difficile (ce qui est certainement le cas de la clé de la racine). Pour ces cas, une clé de 2048 bits est donc utile. Attention, ce n'est pas gratuit, notamment la vérification est quatre fois plus lente et la signature huit fois moins rapide.

Certains férus de cryptographie peuvent tiquer à la pensée de n'utiliser que 1024 bits. Mais l'argument massue du RFC est celui de l'équilibre : on ne fait pas du DNSSEC pour le plaisir de la cryptographie. On le fait pour protéger quelque chose, par exemple un site Web de banque en ligne. Et ce genre de sites est en général protégé par TLS avec des clés qui ne sont souvent pas plus longues. Si un attaquant peut casser une clé de 1024 bits, il peut s'attaquer à TLS aussi bien qu'à DNSSEC. Avec des clés de 2048 bits, on blinde fortement la porte... alors que la fenêtre du rez-de-chaussée ne l'est pas toujours. Bref, si un brusque progrès de la cryptanalyse permet de casser ces clés de 1024 bits plus tôt que les dix ans indiqués, cela fera du bruit et cassera bien d'autres choses que DNSSEC.

Notons que, contrairement à son prédécesseur RFC 4641, notre RFC ne fait plus dépendre la solidité d'une clé du fait qu'elle soit largement utilisée ou pas. La cryptographie a évolué depuis le RFC 4641.

Où stocker ensuite ces clés ? Idéalement, sur une machine sécurisée et non connectée au réseau. On apporte les données sur cette machine, on signe et on repart avec les données signées pour les apporter au serveur de noms. C'est très sûr mais cela ne peut pas marcher si on utilise les mises à jour dynamiques du DNS (RFC 3007). Dans ce cas, il faut bien avoir la clé privée sur le serveur de noms (on parle de "*online key*"). C'est donc moins sûr et le RFC suggère, dans ce cas, que le serveur maître ne soit pas directement accessible de l'Internet (technique du "*hidden master*", où le vrai maître n'est pas dans les enregistrements NS de la zone). Notez qu'une des motivations pour la séparation KSK/ZSK vient de là : avoir une KSK peu utilisée et dont la clé privée est stockéee de manière très sûre et une ZSK en ligne, plus vulnérable mais plus facile à changer.

Comme toujours en matière de sécurité, ces bons avis doivent être mis en rapport avec les contraintes opérationnelles et économiques. Ainsi, on peut sans doute améliorer la sécurité des clés en les gardant dans un HSM. Mais ces engins sont chers et pas pratiques pour les opérations quotidiennes.

Et la génération des clés ? C'est un aspect souvent oublié de la sécurité cryptographique mais il est pourtant essentiel. La clé doit être vraiment imprévisible par un attaquant et pour cela elle doit être générée en suivant les recommandations du RFC 4086 et de NIST-SP-800-90A <http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf>, « "*Recommendation for Random Number Generation Using Deterministic Random Bit Generators*" ». Par exemple, avec les outils de génération de clés de BIND (`dnssec-keygen`), il ne faut **pas** utiliser l'option `-r /dev/urandom` en production, car elle n'utilise qu'un générateur pseudo-aléatoire. (Les HSM disposent tous d'un générateur correct.)

À noter que la version précédente de ce document, le RFC 4641 faisait une différence entre les zones DNS selon leur place dans la hiérarchie (plus ou moins haute). Cette distinction a disparu ici et les mêmes règles de sécurité s'appliquent à toutes. L'une des raisons de ce choix est qu'on ne peut pas facilement déterminer quelles zones sont importantes pour un attaquant (`grossebanque.com` est peut-être plus importante que le TLD `.td`, même si ce dernier est placé plus haut dans la hiérarchie). Cela n'empêche évidemment pas chaque gérant de zone de devoir déterminer si sa zone est critique ou pas, avant d'adapter les mesures de sécurité en conséquence.

La section 4 de notre RFC est ensuite consacrée au gros morceau de la gestion opérationnelle de DNSSEC : les remplacements de clés ("*key rollovers*"). Qu'on choisisse de faire des remplacements réguliers, pour qu'ils deviennent une simple routine, ou qu'on choisisse de n'en faire que lors d'évenements exceptionnels (comme une copie illégale de la clé), les remplacements sont une réalité et il faut s'y préparer. Par exemple, même si on décide de ne pas en faire systématiquement, il faut s'entrainer (et tester ses procédures) avec une zone de test. L'opération est délicate en raison de l'existence des caches DNS, qui vont garder la vieille information un certain temps. Personnellement, je déconseille fortement de faire les remplacements à la main : le risque d'erreur est bien trop grand. Il faut plutôt automatiser le processus, et avec un logiciel soigneusement débogué. L'expérience (mon article à la conférence SATIN en 2011 <https://www.bortzmeyer.org/satin.html>) montre que les erreurs sont, elles aussi, une réalité.

Si on a décidé d'avoir deux clés, ZSK et KSK, leurs remplacements sont indépendants. Les ZSK sont les plus simples car elles peuvent être remplacées sans interaction avec la zone parente (pas d'échange d'enregistrements DS). Le RFC explique ensuite (en grand détail : je n'en donne qu'un court résumé ici) les deux méthodes connues pour remplacer une ZSK :

— la pré-publication de la future clé,
— et la double signature.

Dans le premier cas, on génére la nouvelle ZSK, on la publie (sans qu'elle signe quoi que ce soit) puis, lorsque le nouvel ensemble `DNSKEY` est dans tous les caches, on utilise la nouvelle ZSK pour signer. Une fois que les signatures faites avec l'ancienne clé ont disparu des caches, on arrête de publier l'ancienne clé. Le remplacement est terminé. Cette description très sommaire de l'algorithme suffit déjà, je crois, à montrer qu'il vaut mieux ne pas le faire à la main.

Quant à la double signature (plus rare, en pratique), elle consiste à signer tout de suite avec la nouvelle clé, tout en continuant à signer avec l'ancienne, pour les caches ayant seulement l'ancienne clé.

La section 4.1.1.3 résume les avantages et inconvénients de chaque méthode. La pré-publication nécessite davantage d'étapes (ce qui est grave si on la fait à la main, et moins si on a tout automatisé). La double signature est plus coûteuse puisque la zone sera, pendant la période transitoire, bien plus grosse (les signatures forment l'essentiel de la taille d'une zone signée). Dans les deux cas, comme

il n'y a pas besoin d'interagir avec la zone parente, le processus peut être entièrement automatisé et c'est bien ce que fait, par exemple, OpenDNSSEC.

Et pour remplacer une KSK ? La double signature devient alors plus intéressante, puisque la KSK ne signe pas toute la zone, seulement l'ensemble d'enregistrements `DNSKEY`. Les problèmes de taille ne se posent donc pas. Par rapport au remplacement d'une ZSK, la grosse différence est l'interaction avec la zone parente. Il n'existe pas d'API standard pour le faire. Les bureaux d'enregistrement (BE) utilisent en général EPP - RFC 5910 - avec le registre mais le gestionnaire de la zone, avec son BE, a typiquement une interface Web, ou bien une API non-standard, par exemple à base de REST (cf. aussi la section 4.3.2). L'opération nécessite donc souvent une intervention manuelle comme décrit dans mon article sur un exemple avec OpenDNSSEC <https://www.bortzmeyer.org/key-rollover.html>.

Et si on a une clé unique ? Le cas est proche de celui d'un remplacement de KSK et le RFC recommande la double-signature, car c'est la solution la plus simple (et, si on n'utilise qu'une clé, c'est qu'on aime la simplicité).

La section 4.2 couvre le cas des préparatifs pour un remplacement d'urgence. On arrive le lundi matin au bureau, le coffre-fort est grand ouvert et la clé USB qui stockait la clé privée a disparu (une variante moins gênante est celle où la clé privée est détruite, mais pas copiée, par exemple en raison d'un incendie). Il faut alors remplacer cette clé et c'est bien plus facile si on a planifié avant. Notez qu'il y a toujours un compromis ici : si on supprime immédiatement l'ancienne clé, désormais compromise, une partie des résolveurs (ceux qui ont les anciennes signatures, mais pas l'ancienne clé) dans leurs caches considéreront la zone comme invalide. Mais, si on continue à diffuser l'ancienne clé, on augmente la durée pendant laquelle le voleur pourra injecter des informations mensongères. Il existe une méthode intermédiaire, faire retirer l'enregistrement DS par la zone parente. Selon les TTL en jeu, cela peut permettre de diminuer la durée du problème. Comme on le voit, il vaut mieux faire les calculs, l'analyse et le choix de la meilleure méthode avant le fatal lundi matin...

Une solution possible est d'utiliser des "*Standby keys*", qui seront publiées (et donc dans tous les caches) mais pas utilisées pour signer, et dont la partie privée sera gardée à un endroit différent.

Le problème est évidemment plus grave si la clé en question avait été configurée par certains comme clé de confiance dans leurs résolveurs. Il faut alors prévenir tout le monde de la compromission, par un message (authentifié) envoyé à plusieurs endroits, pour maximiser les chances de toucher tout le monde. Si la clé de la racine (ou d'un grand TLD comme `.net`) était compromise, on verrait sans doute l'annonce sur dns-operations <https://lists.dns-oarc.net/mailman/listinfo/dns-operations>, NANOG, Twitter, etc. Outre l'authentification de l'annonce, la nouvelle clé devra elle aussi être authentifiée, par un moyen non compromis : ne mettez pas la partie privée de la KSK dans le même coffre-fort que la partie privée de votre clé PGP !

Autre cas amusant pour les gérants d'une zone DNS, le changement d'opérateurs. Par exemple, le titulaire d'une zone décide de passer d'un hébergeur DNS à un autre (notez que l'hébergeur est souvent le BE mais ce n'est pas obligatoire). Si les deux hébergeurs (en pratique, c'est évidemment le perdant qui fait des histoires) coopèrent, le problème est relativement simple. Le RFC suppose que l'ancien hébergeur ne va pas transmettre la clé privée (si elle est dans un HSM, il ne peut tout simplement pas). Dans ce cas, l'ancien hébergeur doit mettre dans la zone la clé publique du nouveau, et les signatures calculées par le nouveau avec ses clés. Le nouvel opérateur publie la même zone et, une fois le changement terminé (une fois que les clés du nouvel hébergeur sont dans tous les caches), le nouvel hébergeur peut retirer les clés et les signatures de l'ancien. Ce n'est pas très pratique (il faut échanger clés publiques et signatures). Personnellement, je trouve cet algorithme tout à fait irréaliste. Dans le DNS d'aujourd'hui, des obligations bien plus modérées (en cas de changement d'hébergeur, continuer à servir la zone jusqu'à ce que les TTL sur les enregistrements NS expirent) ne sont jamais respectées par les hébergeurs

perdants, même lorsqu'elles figurent dans les obligations contractuelles. Le RFC note à juste titre que c'est un problème économique et qu'il n'a pas de solution technique.

Il reste donc le cas qui sera sans doute le plus probable, celui où l'ancien hébergeur de la zone DNS ne coopérera pas. (Techniquement, il peut même saboter activement le transfert, en publiant un enregistrement DNSKEY avec un très long TTL, pour « empoisonner » les caches.) La seule solution est alors de passer par une phase où la zone ne sera pas sécurisée par DNSSEC : le parent retire l'enregistrement DS, on attend que l'ancien DS ait disparu des caches, puis on en remet un, pointant vers une clé du nouvel hébergeur.

Quelques mesures techniques dans les résolveurs validants peuvent aider :
— Limiter les TTL qu'on accepte (RFC 2308) pour diminuer le risque d'obstruction,
— Vérifier la délégation lorsque les enregistrements NS vont expirer, pour être sûr de ne pas rester bloqué sur de vieux serveurs de noms (c'est également une protection contre l'attaque des domaines fantômes),
— Lorsque la validation échoue, réessayer après avoir viré les DNSKEY du cache.

Le gérant de la zone peut aussi aider en n'ayant pas des TTL trop longs.

Gérer proprement du DNSSEC impose une grande rigueur sur les questions temporelles. Avant DNSSEC, le temps dans le DNS était toujours **relatif**. Par exemple, le champ "*Expire*" de l'enregistrement SOA était relatif au moment de la dernière synchronisation avec le serveur maître. Même chose pour les TTL. Mais, avec DNSSEC, le temps devient **absolu**. Les dates de début et de fin de validité dans un enregistrement RRSIG, par exemple, sont absolues. Elles dépendent donc d'une horloge à l'heure.

Quelques conseils, donc, sur le temps :
— La période de validité des signatures devrait être au moins égale au TTL (autrement, des signatures encore dans le cache pourraient être expirées),
— Ne pas attendre le dernier moment pour re-signer. Cela laisse un délai pour corriger d'éventuels problèmes. Par exemple, si les signatures ont en permanence deux jours de validité devant elles, cela permet de survivre pendant un week-end, en cas d'incapacité à re-signer.
— Ne pas mettre de TTL trop bas (du genre 5 minutes) à la fois à cause de la charge sur les serveurs et aussi parce que la validation DNSSEC nécessite de récupérer beaucoup d'enregistrements et qu'il ne faut pas que leur TTL expire avant que le processus ne soit terminé.
— Pour une zone signée, avoir un "*Expire*" (dans le SOA) plus long que la durée de validité des signatures ne sert pas à grand'chose : un serveur esclave qui servirait des signatures qui ne sont plus valides serait pire qu'un serveur esclave qui ne servirait plus rien.

Bon, mais alors, quelle durée choisir pour la validité des signatures (les outils de signature de BIND mettent un mois, par défaut). C'est un compromis : une durée trop longue et on est vulnérable aux attaques par rejeu. Une durée trop courte et on risque de n'avoir pas assez de temps pour réagir en cas de problème. Une erreur dans la configuration du pare-feu, les serveurs esclaves qui ne peuvent plus se synchroniser, les signatures sur ces serveurs esclaves qui expirent et crac, la zone est invalide. Cela arrive vite. Mon opinion personnelle est qu'aujourd'hui, les procédures ne sont pas assez testées et l'expérience DNSSEC est insuffisante. Il faut donc jouer la sécurité et avoir des durées de validité longues (au moins deux semaines). Dans le futur, au fur et à mesure de la montée en compétence et de l'amélioration des logiciels, on pourra mettre des durées de validité des signatures plus courtes, fermant ainsi la porte aux attaques par rejeu.

Voilà, nous avons couvert le gros de ce RFC. La section 5, qui suit, se consacre à un problème rigolo, celui des enregistrements indiquant l'enregistrement suivant, les enregistrements "*next*". L'un des choix importants de conception de DNSSEC était de permettre que les signatures se fassent entièrement hors-ligne. On ne pouvait donc pas prévoir toutes les questions posées au serveur et avoir une signature pour toutes les réponses « ce domaine n'existe pas ». Il y a donc des enregistrements qui disent « il n'y a pas de domaine ici », enregistrements qui sont signés pour qu'un résolveur validant puisse authentifier

une réponse négative. Ces enregistrements "*next*" sont le NSEC et le NSEC3. Le premier est en clair (il indique le nom de l'enregistrement suivant celui qui n'existe pas), le second est brouillé (il indique le condensat suivant le condensat du nom qui n'existe pas). Pour l'administrateur, les NSEC sont plus pratiques, notamment au déboguage. Mais ils permettent la récupération complète de la zone, en marchant de NSEC en NSEC.

Le choix entre NSEC et NSEC3 dépend du type de la zone. Si on a une petite zone au contenu prévisible (juste l'apex, un `mail` et un `www`, ce qui est le cas de très nombreuses zones), aucune raison d'utiliser NSEC3. Même chose si la zone est très structurée et donc très prévisible (`ip6.arpa`, ENUM...). Même chose si la zone est publique, ce qui est le cas par exemple de la racine (il existe une autre raison d'utiliser NSEC3, l'"*opt-out*", qui intéresse surtout les grandes zones de délégation comme certains TLD). Autrement, si on n'est dans aucun de ces cas, si on est une assez grande zone au contenu imprévisible et qui n'est pas publique, il est cohérent d'utiliser NSEC3 (RFC 5155).

NSEC3 a certains paramètres à configurer (et qui sont publiés dans l'enregistrement NSEC3PARAM). Notamment, on peut définir le nombre d'itérations de hachage. Cette possibilité protège des attaques par dictionnaires pré-calculés mais elle est coûteuse et elle mène à l'exécution de code à chaque requête pour un nom inexistant. Le RFC suggère un nombre de 100 itérations, ce qui est colossal par rapport à ce que font la plupart des zones NSEC3 aujourd'hui (5 par défaut avec OpenDNSSEC, mais beaucoup l'abaissent, on trouve 1 pour `.fr` et `.com`, et même 0 pour `.org`...) NSEC3 ajoute un sel au début de l'itération et notre RFC suggère de changer ce sel en même temps que la ZSK.

Les changements depuis le RFC 4641 sont résumés dans l'annexe E. D'abord, les erreurs connues `<http://www.rfc-editor.org/errata_search.php?rfc=4641>` ont été corrigées. Ensuite, la fonction de hachage SHA-256 a été ajoutée aux recommandations. La partie sur le changement d'hébergeur DNS a été ajoutée. Le modèle avec séparation KSK/ZSK ne bénéficie plus d'un privilège particulier puisque le modèle avec une seule clé est également décrit.

(Les juristes noteront que VeriSign a un brevet `<https://datatracker.ietf.org/ipr/1924/>` sur certaines des techniques présentées dans ce RFC.)