RFC 7583 : DNSSEC Key Rollover Timing Considerations

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 22 octobre 2015

Date de publication du RFC : Octobre 2015

https://www.bortzmeyer.org/7583.html

Contrairement au DNS classique, le système de sécurité DNSSEC est très dynamique : il faut changer les signatures régulièrement et il est souvent recommandé de remplacer (changer) les clés. Compte-tenu du fait que le DNS n'est pas synchrone (les changements n'apparaissent pas instantanément partout dans l'Internet), ce changement, ce remplacement ("*rollover*", dans la langue d'Alan Turing) est une opération délicate et qui nécessite le strict respect d'un certain nombre de durées. Ce RFC donne des conseils sur la temporalité des opérations DNSSEC.

Pourquoi faut-il changer les clés cryptographiques utilisées ? Il y a plusieurs écoles <https://www.bortzmeyer.org/remplacement-cles.html> à ce sujet. Les minimalistes considèrent qu'on ne change les clés que quand c'est nécessaire, par exemple pour passer à des clés plus longues et donc moins vulnérables à la cryptanalyse, ou évidemment quand les clés privées ont été copiées par un méchant. Les maximalistes estiment qu'il faut changer de clé régulièrement, pour être mieux préparé en cas de changement d'urgence, ou pour priver un méchant discret qui aurait réussi à copier une clé privée sans qu'on s'en aperçoive du fruit de son forfait, ou tout simplement parce que l'utilisation d'une clé fournit du matériau, qui peut être utile à un éventuel cryptanalyste. Changer de clé est une opération délicate et, à mon avis, devrait être automatisé : autrement, le risque d'erreur est trop important, comme l'avait montré mon article à la conférence SATIN <https://www.bortzmeyer.org/satin.html>.

En effet, les facteurs qui rendent l'opération compliquée sont :

— Les enregistrements DNSSEC, comme les clés (DNSKEY), les signatures (RRSIG) et les pointeurs vers les clés (DS) ne sont pas seulement dans les serveurs faisant autorité mais sont également mémorisés (« cachés ») dans les résolveurs. Si le TTL d'un DNSKEY est de 24 h, il faudra continuer, pendant une journée entière, à permettre la validation avec l'ancienne clé.
— Il n'y a pas que le remplacement des clés dans les zones déjà signées, il y a aussi la première introduction de la signature dans les zones existantes.
— Tous les remplacements de clés ne vont pas se faire de manière planifiée : il risque d'y avoir des remplacements d'urgence et ils peuvent nécessiter une préparation spécifique, avec des clés de secours pré-publiées, par exemple.

— Tout serait plus simple si on pouvait publier toutes les clés possibles, y compris celles non utilisées depuis longtemps. Mais les requêtes DNS pour des enregistrements DNSKEY renvoient toutes les clés et cela peut dépasser la taille qu'on accepte pour des réponses DNS (typiquement 1480 ou bien 4096 octets). Il faut donc supprimer les anciennes clés, et cela ne doit pas se faire trop tôt : il faut être sûr que plus aucun résolveur n'ait ces clés dans son cache.

Une description de ce processus de remplacement de clés ("*key rollover*") figure dans le RFC 6781 [1]. Un exemple avec OpenDNSSEC est raconté dans un de mes articles <https://www.bortzmeyer.org/key-rollover.html>. Notre nouveau RFC détaille la description du RFC 6781, notamment sur les aspects temporels. Deux points à garder en tête avant de le lire :

— Il est très courant d'avoir deux types de clés, les KSK ("*Key-Signing Key*") qui servent à signer les clés, et qui sont pointées depuis la zone parente, et les ZSK ("*Zone-Signing Key*") qui signent tout le reste. Comme la KSK, elle, est pointée depuis une zone gérée par une organisation différente (la zone parente), elle pose des problèmes particuliers et les deux types de clés sont donc traités séparement dans ce RFC. Le cas moins courant où on n'utilise qu'un seul type de clé n'est pas couvert.
— Le RFC ne traite pas le cas (très délicat et surtout très peu testé dans le monde réel) des remplacements d'algorithmes (comme le passage de clés RSA à des clés ECDSA, cf. RFC 6605). On trouvera sans doute bien des bogues quand on essaiera cela en vrai ! (Cf. section 5)

Rappelez-vous en outre qu'une requête DNS d'un type donné (par exemple, A, NS, DNSKEY) récupère toujours tous les enregistrements de ce type. Si l'ensemble des enregistrements (RRset pour "*Resource Record Set*") a changé entre temps, mais qu'un ensemble plus ancien est dans le cache, il y restera jusqu'à l'expiration du TTL.

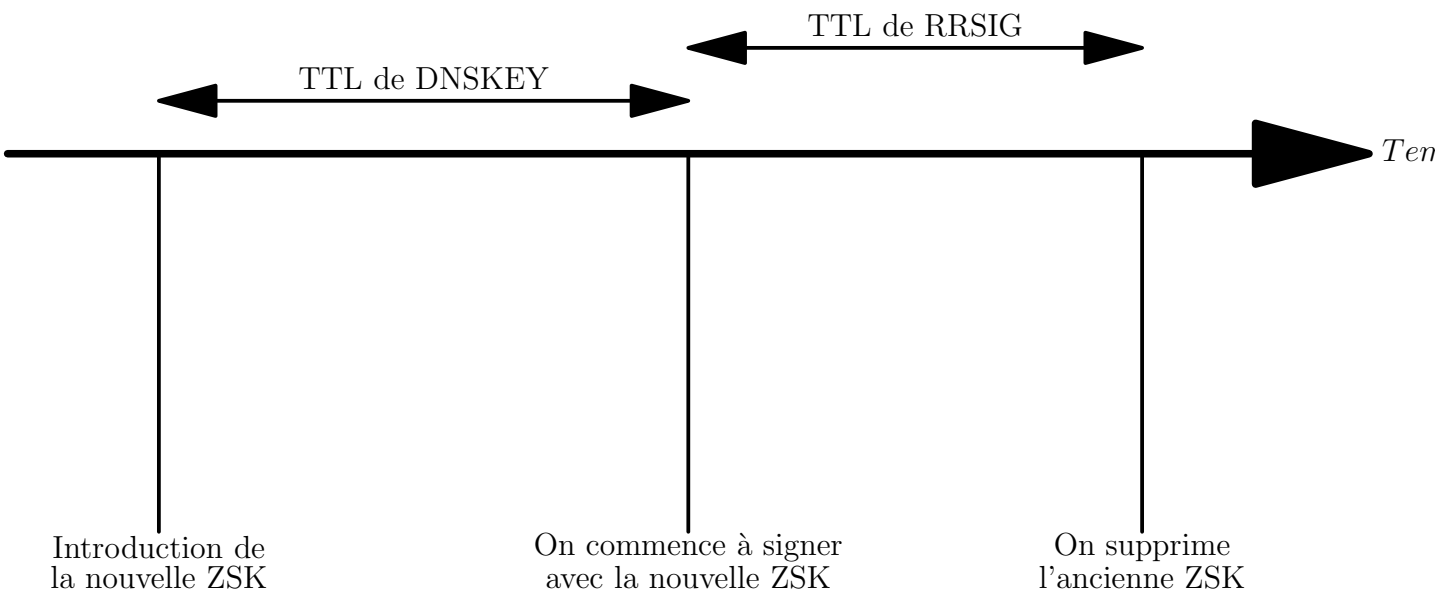
Après ces rappels, les méthodes pour faire un remplacement de clé (section 2 de notre RFC). D'abord, les ZSK ("*Zone-Signing Keys*"). Il faut que tout résolveur validant qui a accès à une signature ait également accès à la ZSK correspondante, et ce à tout moment, que l'information soit dans le cache ou pas. Pas question donc de se contenter de publier la nouvelle ZSK et de l'utiliser immédiatement pour signer (les résolveurs ayant l'ancien enregistrement DNSKEY dans leur cache ne pourraient pas valider les nouvelles signatures, et idem si c'est la signature qui est dans le cache mais pas la clé). Il y a trois techniques pour un remplacement qui marche :

— La **pré-publication**, décrite dans le RFC 6781. On met la nouvelle ZSK dans l'enregistrement DNSKEY (en laissant les anciennes), et on attend le TTL du DNSKEY, pour être sûr que tous les caches qui ont le DNSKEY ont le nouveau. On peut alors signer avec la nouvelle clé. Puis on attend le TTL des RRSIG, et on est alors certain qu'il ne reste plus de signatures faites avec l'ancienne clé dans les caches. On peut alors supprimer l'ancienne clé du DNSKEY. C'est la technique utilisée par OpenDNSSEC.
— La **double-signature** (un nom pas très rigoureux, car cette technique repose aussi sur une double clé, pas uniquement une double signature), également dans le RFC 6781. On introduit la nouvelle clé dans le DNSKEY et on l'utilise tout de suite pour faire des signatures mais en gardant les anciennes signatures (d'où le nom de double-signature). Une fois le maximum des TTL de DNSKEY et de RRSIG passé, on peut retirer l'ancienne clé et les anciennes signatures.
— La **double-RRSIG** qui est une vraie « double signature ». On signe avec la nouvelle clé (en gardant les anciennes signatures) mais sans la publier. Une fois le TTL de RRSIG expiré, on change la ZSK. Puis, une fois le TTL de DNSKEY passé, on peut retirer les anciennes signatures.

La double-signature est probablement la plus facile à comprendre : à tout moment, durant la transition, on aura à la fois l'ancienne et la nouvelle clé, les anciennes et les nouvelles signatures. Elle n'a que deux étapes (introduire la nouvelle clé et les nouvelles signatures, puis retirer l'ancienne clé et les anciennes signatures). Son principal inconvénient est d'augmenter la taille des réponses puisqu'il faut transmettre deux fois plus de signatures. La pré-publication est plus compliquée (trois étapes) mais maintient la taille des réponses au minimum nécessaire. Notez que « plus compliqué » est surtout un problème si vous voulez tout comprendre, ou bien si vous êtes le programmeur qui va devoir automatiser l'opération.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc6781.txt

3.2.1, ZSK, pré-publication



Mais, si vous utilisez un logiciel déjà fait, comme OpenDNSSEC, vous ne verrez pas cette complexité. La double-RRSIG a les inconvénients des deux autres techniques sans leurs avantages. C'est pour cela qu'elle ne figure pas dans le RFC 6781 et qu'elle n'est pas davantage mentionnée ici.

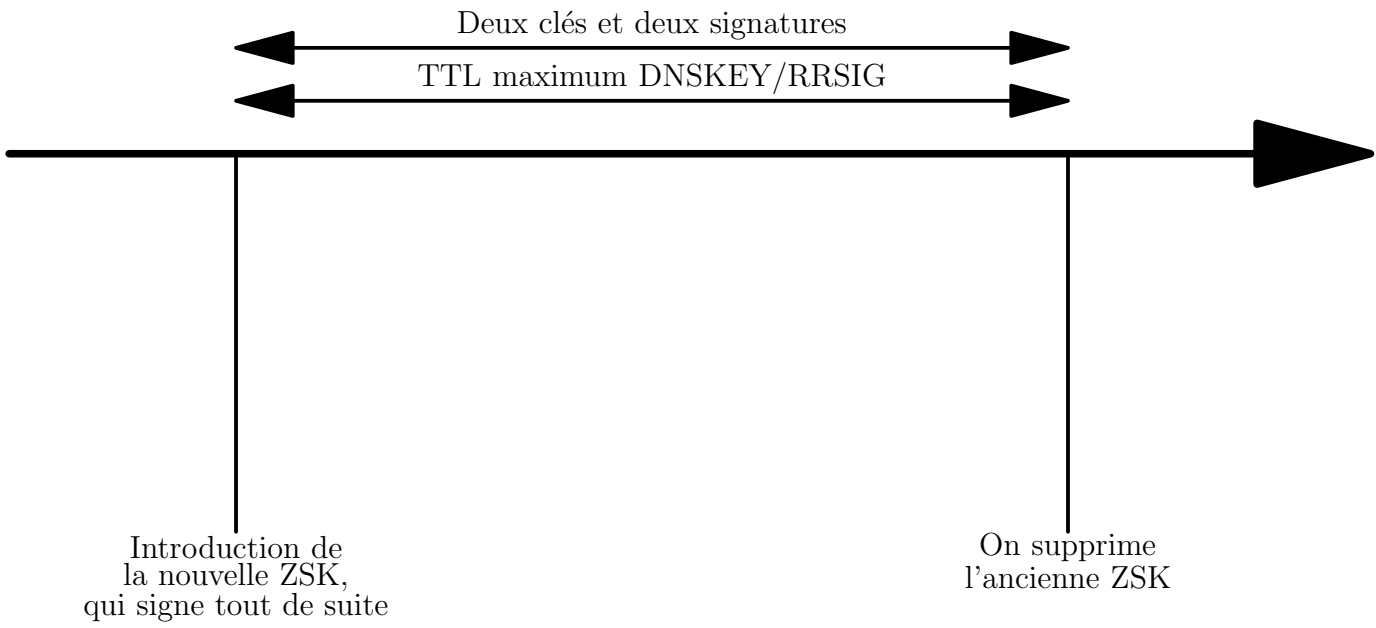
Et pour un remplacement de KSK ? Le problème est plus simple car la KSK n'est utilisée que pour signer le DNSKEY et clé et signatures voyagent donc ensemble (mais n'ont pas forcément le même TTL, ce que le RFC oublie de dire, donc l'une peut expirer du cache avant l'autre). Mais il est aussi plus compliqué car la KSK est pointée par un lien de confiance ("*trust anchor*"), typiquement un enregistrement DS dans la zone parente. Il faut donc coordonner le remplacement de KSK avec celui de ces liens. Et, comme le sait tout administrateur réseau, tout est plus compliqué quand on doit se coordonner avec le monde extérieur. Contrairement au remplacement d'une ZSK, où tout peut être automatisé car le passage d'une phase à l'autre ne dépend que de l'écoulement du temps, le remplacement d'une KSK nécessite une observation manuelle du parent, dont le délai de réaction est imprévisible.

Si le lien de confiance est un enregistrement DS (cas le plus fréquent), il faudra une interaction avec le gestionnaire de la zone parente. Si ce lien a été configuré manuellement (c'est rare, sauf pour la zone racine, qui n'a pas de parente), il faudra faire le changement dans tous les résolveurs qui ont configuré ce lien (une tâche difficile), ou bien utiliser le RFC 5011 (cf. section 3.3.4). C'est en raison des incertitudes à ce sujet qu'il n'y a jamais eu de remplacement de la KSK de la racine <https://www.bortzmeyer.org/root-key-rollover.html>.

Comme pour le changement de ZSK, il y a trois techniques :

— La **double-KSK**. On met la nouvelle KSK dans l'ensemble DNSKEY. On attend le TTL de la DNSKEY (on est alors sûr que les caches contiennent la nouvelle clé). On change le DS. Une fois celui-ci changé par le parent, on attend le TTL du DS, puis on supprime l'ancienne KSK. C'est la technique utilisée par OpenDNSSEC.
— Le **double-DS**. On publie un DS pour la nouvelle KSK. Après que le TTL du DS se soit écoulé, on change la clé. Après que le TTL de la DNSKEY se soit écoulé, on supprime l'ancien DS.
— Le **double-RRset**. On publie la nouvelle clé, on signe le DNSKEY avec l'ancienne et la nouvelle clé et on publie le nouveau DS. Une fois le maximum du TTL du DS et du TTL de la DNSKEY sont passés, on supprime l'ancien DS et l'ancienne DNSKEY.

3.2.2, ZSK, double signature



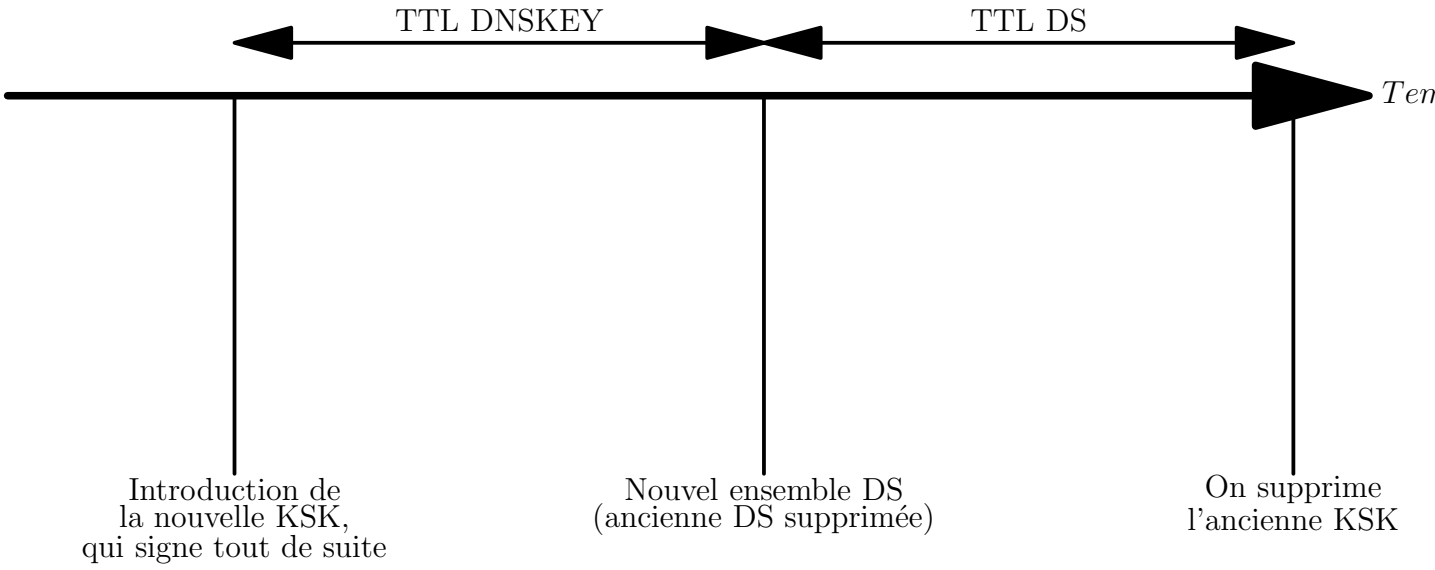
Le double-KSK est le plus simple à comprendre, mais cette technique augmente la taille de l'ensemble DNSKEY. Le double-DS n'a pas ce défaut mais nécessite deux interactions avec le parent (certaines zones parentes, comme la racine, sont très lentes à réagir). Le double-RRset a les inconvénients des deux autres techniques mais est aussi la technique qui minimise le temps total de remplacement.

La section 3 du RFC présente les frises chronologiques ("*timeline*") détaillées. Elle s'appuie sur une liste d'états des clés (états qu'affiche un logiciel comme OpenDNSSEC <https://www.bortzmeyer.org/opendnssec-states.html>) :

— "*Generated*" : clé créée mais pas encore utilisée. (Les clés peuvent être créées à l'avance.)
— "*Published*" : la clé est publiée dans le DNS.
— "*Ready*" : la clé est publiée depuis suffisamment longtemps pour qu'on soit sûr que, si le DNSKEY est dans un cache, il inclut cette clé.
— "*Active*" : la clé est utilisée pour signer (si c'est une ZSK) ou bien elle permet de valider le DNSKEY (si c'est une KSK).
— "*Retired*" : la clé n'est plus utilisée pour signer mais est encore publiée (certains caches peuvent avoir des vieilles signatures qui nécessitent cette clé).
— "*Dead*" : les caches ont tous la nouvelle clé, elle ne sert plus à rien.
— "*Removed*" : la clé n'est plus publiée.
— "*Revoked*" : cet état n'existe que si on utilise le RFC 5011. Il indique que la clé va être retirée et que les résolveurs qui l'utilisent comme lien de confiance doivent se préparer à ne plus l'utiliser.

Les sections 3.2 et 3.3 présentent les frises chronologiques ("*timelines*") des différentes techniques de remplacement d'une ZSK et d'une KSK (en art ASCII...) Les frises sont un peu complexes car, à chaque fois, on voit deux introductions d'une nouvelle clé, la N et la N+1. L'annexe A détaille les abréviations utilisées. Je n'ai pas reproduit ici les frises du RFC mais juste une version très simplifiée. Quelques points sont dignes d'être notés. D'abord, il peut être prudent, lorsqu'on met en œuvre ces frises, d'ajouter une marge à chaque opération (les éléments <PublishSafety> et <RetireSafety> dans la configuration d'OpenDNSSEC). Si une clé est publiée à l'instant T, et que le TTL est de N secondes, on peut compter que cette clé soit accessible à tous les caches à T + N mais il est plus prudent d'utiliser T + N + M où M est une marge qui permet de faire face à certains imprévus (un retard dans la mise à jour de certains serveurs faisant autorité, par exemple).

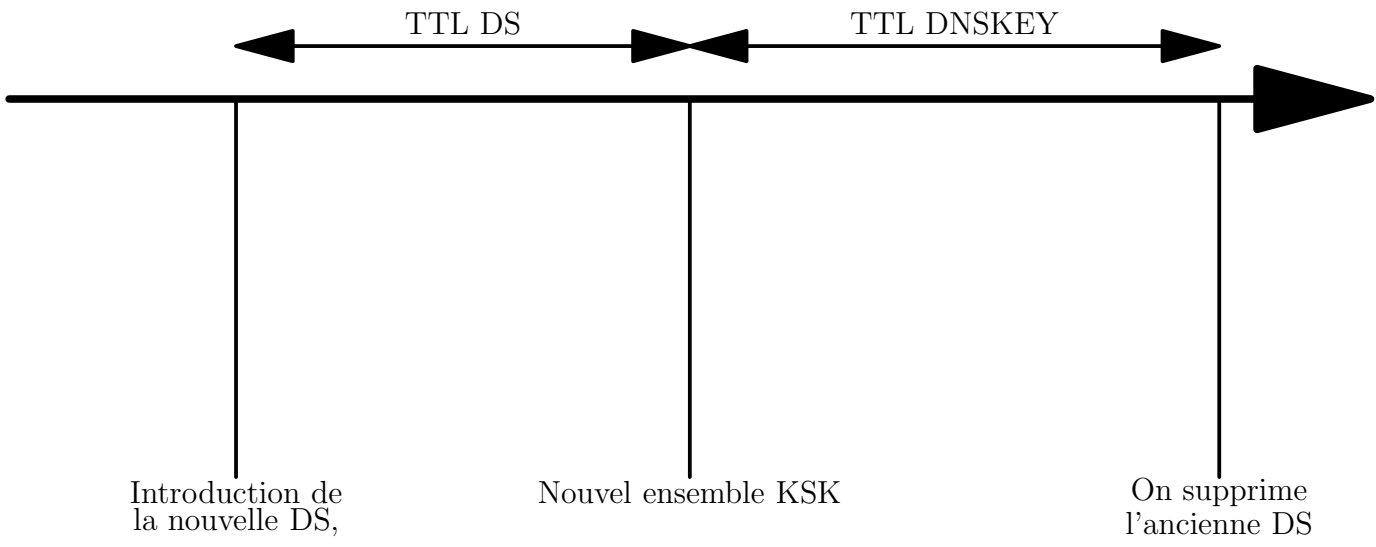
3.3.1, KSK, double KSK



On peut améliorer la technique de pré-publication en introduisant des **clés en attente** (”*standby keys*”, section 4). Ces clés sont pré-publiées mais ne sont pas utilisées même lorsque le TTL de DNSKEY est passé. Elles attendent simplement. En général, elles sont là pour des cas d'urgence. Un exemple est celui d'une clé normale, stockée sur la machine qui fait les signatures DNSSEC, et une clé en attente, stockée sur une clé USB dans un coffre-fort. On pré-publie la clé en attente. Si, un matin, on découvre que la machine de signature a été piratée, on peut basculer tout de suite vers la clé en attente, puisqu'elle est dans les caches (la clé en attente est en permanence dans l'état ”*Ready*”). À noter que, pour une KSK et pour la technique du Double-DS, ce n'est pas la clé qu'on publie mais le DS correspondant. On a donc un ”*standby DS*” (c'est le cas aujourd'hui sur `.fr`, si vous regardez). Notez aussi un point de sécurité : une clé en attente n'a de sens que si elle se trouve stockée dans un endroit différent des autres clés. Autrement, elle sera piratée avec les autres.

Donc, en résumé (section 6) :
— Pour les ZSK, la pré-publication est sans doute la technique la plus raisonnable.
— Pour les KSK, le double-RRset est sans doute la meilleure technique, en raison du temps total qu'elle prend, plus court que pour les deux autres techniques.

3.3.2, KSK, double DS



3.3.3, KSK, double RRset