Les extensions de sécurité DNS (DNSSEC)

Gilles GUETTE
IRISA
Campus de Beaulieu,
35 042 Rennes Cedex, France
gilles.guette@irisa.fr

I. INTRODUCTION

Lorsqu'une machine connectée à un réseau veut contacter une autre machine, l'un des premiers protocoles appelés est le DNS (*Domain Name System*). Le DNS [1], [2], [3] est une base de données distribuée et hiérarchique utilisée le plus souvent pour effectuer la correspondance entre un nom de machine et son adresse IP. Les informations contenues dans le DNS sont publiques et les serveurs DNS sont accessibles par tout le monde. Dans sa conception originelle, le DNS n'inclut aucun service de sécurité tels que l'intégrité ou l'authentification, ce qui laisse ce protocole vulnérable [4], [5], [6], [7]. Pour pallier ces vulnérabilités, l'*Internet Engineering Task Force* (IETF) a standardisé les extensions de sécurité DNS (DNSSEC).

DNSSEC [8], [9], [10], [11], [12] repose sur l'utilisation de la cryptographie à clé publique pour fournir l'intégrité et l'authenticité des données DNS. Chaque nœud de l'arbre DNS, appelé *zone*, possède au moins une paire de clés publique/privée utilisée pour générer les signatures numériques des informations de zone. L'unité de base de ces informations est l'enregistrement de ressource (RR). Chaque RR possède un type particulier qui indique le type des données qu'il contient. Par exemple, un enregistrement DNSKEY contient une clé publique de zone, un enregistrement RRSIG contient une signature et un enregistrement A contient une adresse IPv4.

Pour faire confiance à des données DNS, un résolveur (le client DNS) construit une chaîne de confiance [13] en partant d'un point d'entrée sécurisé dans l'arbre DNS [14] (c'est-à-dire d'une clé de confiance configurée statiquement dans le résolveur), jusqu'à l'enregistrement de ressource demandé. Un résolveur est capable de construire une chaîne de confiance s'il possède un point d'entrée sécurisé et s'il ne traverse que des zones sécurisées jusqu'à la ressource demandée.

Dans cet article, nous présentons, dans la Section II, le fonctionnement du protocole DNS. Puis, dans la Section III, nous décrivons quelques vulnérabilités de ce protocole qui ont amené la définition des extensions de sécurité DNS. Nous détaillons le protocole DNSSEC dans la Section IV.

II. DNS : PRINCIPES ET FONCTIONNEMENT

Le système de noms de domaine [1], [2], [3] permet de conserver des informations sur les machines connectées au réseau. Le protocole DNS est utilisé de manière transparente pour effectuer la correspondance entre un nom de machine, compréhensible par l'homme, et son adresse IP. Il est important de connaître l'architecture sur laquelle repose ce protocole pour comprendre ses faiblesses et le besoin de sécurisation de ce protocole.

A. Domaines et zones

Le système de noms de domaine repose sur une structure arborescente. Cet arbre possède deux types de constituantes : les domaines et les zones.

Un domaine est un sous-arbre complet de l'arbre DNS. Le nom d'un domaine est obtenu en concaténant l'étiquette de tous les nœuds, de la racine du sous-arbre jusqu'à la racine de l'arbre DNS. La construction de l'arbre, qui interdit à deux nœuds frères d'avoir le même nom, assure l'unicité de nom dans la base. Un domaine peut donc en englober un autre, comme c'est le cas par exemple sur la figure 1 : le domaine `fr.` contient le domaine `irisa.fr.`.

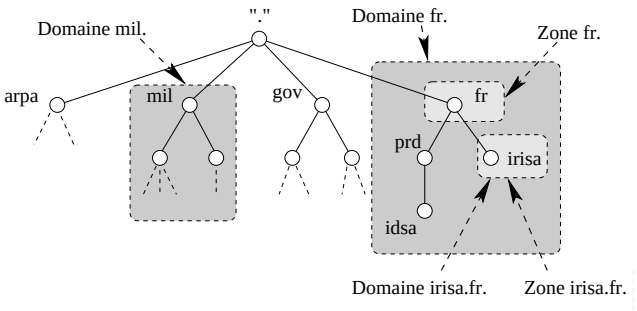

Fig. 1. Domaines et zones DNS.

Chaque domaine est constitué d'une ou plusieurs zones. La zone est l'unité administrative de l'arbre DNS. Une zone est gérée par un ou plusieurs serveurs qui disposent de toutes les informations de cette zone. Les administrateurs de ces serveurs sont responsables des informations de zone, de leur mise à jour et de leur disponibilité.

B. La délégation d'autorité

Une zone est en constante évolution, les informations qu'elle contient doivent être mises à jour et la taille de la zone peut augmenter de manière importante. Pour des raisons de facilité de gestion ou de séparation des tâches, une zone peut décider de transmettre la gestion d'une partie de ses noms : il s'agit d'une délégation.

Lorsqu'une zone délègue la responsabilité d'une partie de ses noms, une nouvelle zone est créée pour les gérer. Cela correspond à la création d'un nouveau nœud dans l'arbre. Un nouveau fichier de zone contenant ces noms est placé sur de nouveaux *serveurs autoritaires*, chargés de l'administration de la nouvelle zone. La nouvelle zone est généralement appelée, la *zone fille*. La zone de départ est appelée *zone mère* ou *zone parente*. Lors de la création d'une délégation, un lien est créé entre la zone mère et la zone fille. Ce lien est représenté par un enregistrement NS (*Name Server*) placé dans le fichier de zone de la zone mère. Cela permet d'avoir un chemin explicite entre la racine et la nouvelle zone.

C. L'architecture du DNS

Le système de noms de domaine est composé de trois entités permettant la gestion et l'utilisation des informations contenues dans le fichier de zone. Il s'agit des serveurs de noms autoritaires, des serveurs caches récursifs et des résolveurs.

1) Les serveurs de noms autoritaires: Le serveur de noms fait autorité sur une ou plusieurs zones, il est responsable des informations de la zone dont il conserve les informations dans un fichier de zone. Chaque enregistrement de ressource est associé à un nom. Les serveurs de noms reçoivent des requêtes sur des noms DNS et répondent avec les enregistrements de resource contenus dans leur fichier de zone.

2) Les serveurs caches récursifs: Les serveurs caches récursifs, ou serveurs caches, sont une des raisons de la résistance au facteur d'échelle du DNS. Les serveurs caches ne font autorité sur aucune zone, ils ne possèdent pas de fichier de zone. Un serveur cache est généralement placé sur un réseau afin de recevoir les requêtes des résolveurs locaux. Le serveur cache a pour rôle de répondre à ces requêtes en utilisant les informations précédemment reçues qu'il conserve en mémoire durant un certain temps. S'il ne possède pas la réponse, le serveur cache fait suivre ces requêtes au serveur autoritaire qu'il pense le plus à même de posséder la réponse, met en cache la réponse reçue et la fait suivre au résolveur. L'utilisation de serveurs caches permet de diminuer la charge sur les serveurs autoritaires en mutualisant les informations [15], [16], [17].

3) Les résolveurs: Le résolveur est l'entité cliente. C'est lui qui reçoit les demandes en provenance d'une application et envoie les requêtes DNS appropriées, généralement à un serveur cache récursif. Lorsque le résolveur reçoit la réponse à sa requête, il transmet l'information à l'application.

D. La résolution de noms

Chaque zone possède un fichier de zone conservé sur ses serveurs autoritaires. Ce fichier contient toutes les informations de la zone. Ces informations sont représentées par des enregistrements de ressource (RR) qui constituent les briques de base de ce fichier. Ces enregistrements ont des types différents selon les informations qu'ils contiennent.

L'action d'effectuer l'association entre un nom et une adresse s'appelle la *résolution de noms*. Il existe deux types de résolution de noms : la résolution itérative et la résolution récursive. La figure 2 montre ces deux principes.

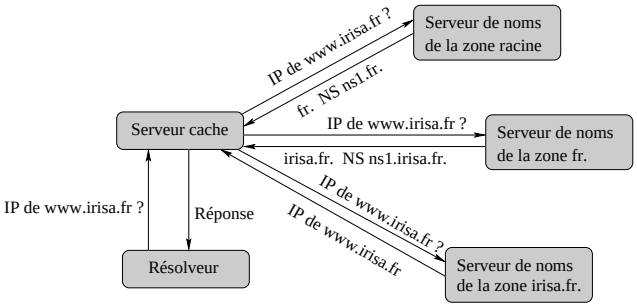

Fig. 2. Les deux types de résolution de noms.

En mode de résolution itératif, le serveur répondra uniquement avec les informations dont il dispose. S'il a autorité sur le nom demandé alors il donnera la réponse, sinon il donnera des informations sur le serveur de noms le plus à même, selon lui, de posséder la réponse. Pour trouver ce serveur, il regarde dans ses délégations celle ayant le plus long suffixe (en terme d'étiquettes) commun au nom demandé. Par exemple le serveur de noms `ns1.` de la zone racine répond avec le nom du serveur de la zone `fr.` (`ns1.fr.`), celle-ci ayant le plus long suffixe commun avec `www.irisa.fr.`. Sur la figure 2 les serveurs de noms autoritaires sont en mode de résolution itératif.

En mode récursif, le serveur effectue la résolution complète, c'est-à-dire qu'il va contacter les serveurs autoritaires en suivant les délégations jusqu'à obtenir la réponse désirée. Sur la figure 2 le serveur cache est en mode de résolution récursif.

En général, les serveurs autoritaires sont configurés en mode itératif pour éviter de consommer du temps et des ressources dans des résolutions de noms, tandis que les serveurs caches sont configurés en mode récursif afin de récupérer un maximum d'informations pour répondre aux résolveurs.

III. Les faiblesses du DNS

Le DNS est une ressource critique et nécessaire au bon fonctionnement d'Internet et des réseaux, mais il est aussi très vulnérable [7], [18], [6]. La figure 3 recense les points faibles de l'architecture DNS. Chaque entité du DNS ainsi que le trafic circulant entre ces entités, que ce soit le trafic standard (requête/réponse) ou le trafic de gestion (mise à jour des fichiers de zone), sont des cibles potentielles. De plus, le DNS étant un service public accessible à tous, il n'échappe pas aux attaques par Déni de Service (DoS) [19], dont certaines ont eu lieu récemment [20], [21]. Nous détaillons ces faiblesses dans les paragraphes suivants.

A. Attaque par oracle (man in the middle)

L'attaque classique par oracle peut s'appliquer au protocole DNS. Un attaquant se trouvant sur le chemin entre sa cible et un serveur de noms ou un serveur cache peut observer les paquets circulant sur ce lien. Il lui suffit alors de répondre plus vite que le serveur interrogé lorsqu'il voit passer une requête DNS.

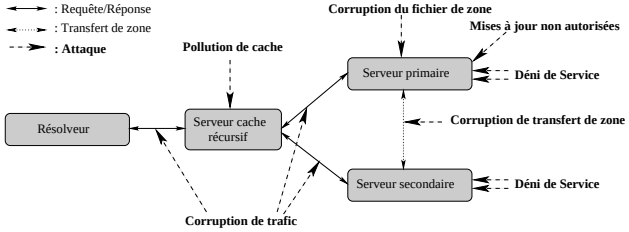

Fig. 3. Les faiblesses du DNS.

Les conséquences d'une telle attaque peuvent être multiples et dépendent uniquement du but de l'attaquant. Il peut construire une fausse réponse contenant des informations erronées pour détourner sa cible vers une machine désignée. Il peut aussi envoyer une fausse réponse signalant que la ressource ou la zone demandée n'existe pas, ce qui a pour conséquence un déni de service.

B. Attaque grâce au paradoxe de l'anniversaire

L'en-tête d'un message DNS contient un identifiant de 16 bits généré aléatoirement par la machine envoyant la requête. Cet identifiant est repris dans la réponse. Les résolveurs et les serveurs ayant souvent plusieurs requêtes en attente, cela permet de faire l'association entre une requête et sa réponse. Une réponse reçue avec un mauvais identifiant n'est pas prise en compte.

Lorsqu'un attaquant est capable d'intercepter ou d'écouter le trafic DNS il lui suffit de reprendre l'identifiant de la requête. En revanche, s'il n'a pas cette possibilité, il doit deviner l'identifiant correct à placer dans sa fausse réponse.

Dans les premières versions des logiciels DNS, l'identifiant était incrémenté de 1 à chaque nouvelle requête. Ceci rendait assez facile la découverte de la valeur courante de l'identifiant de requête pour un résolveur ou un serveur donné. Un attaquant pouvait ainsi créer une fausse réponse valide assez facilement en envoyant un très petit nombre de fausses réponses.

Depuis, le calcul de l'identifiant a évolué, l'incrément est maintenant aléatoire. Pour deviner l'identifiant, un nombre de messages plus conséquent est nécessaire, mais pas du tout impossible à générer suivant le paradoxe de l'anniversaire.

L'identifiant des messages DNS étant sur 16 bits, il y a 65536 identifiants possibles. Une attaque de force brute serait, pour une requête donnée, de générer 65536 fausses réponses. Cela a très peu de chance de fonctionner car l'attaque est limitée dans le temps. Une fois la bonne réponse reçue, les fausses réponses sont rejetées même si l'identifiant est correct.

C'est ici qu'intervient le paradoxe de l'anniversaire. Si l'attaquant envoie un ensemble de n requêtes à sa cible (un serveur cache par exemple), celle-ci va faire suivre les requêtes au serveur de noms autoritaires concernés. Pendant ce temps l'attaquant envoie n fausses réponses avec n identifiants différents à sa cible. Le paradoxe de l'anniversaire donne la formule 1 sur la probabilité de collision des identifiants :

$$\mathcal{P}(ID_x = ID_y) = 1 - (1 - \frac{1}{65536})^{\frac{n \times (n-1)}{2}} \qquad (1)$$

Nous déduisons donc de la formule 1 que lorsque l'attaque est menée avec un ensemble de 302 messages, la probabilité de trouver le bon identifiant est supérieure à $\frac{1}{2}$. Sept cent messages permettent d'approcher une probabilité de 1 (0,97).

C. Pollution d'un serveur cache

Comme nous l'avons vu dans le paragraphe II-C, l'architecture DNS repose sur l'utilisation des serveurs caches récursifs. Ces serveurs utilisent des enregistrements qu'ils possèdent en mémoire pour répondre aux résolveurs. La confiance qu'ont les résolveurs en leurs serveurs caches font de ces derniers des cibles particulièrement intéressantes. Si un attaquant réussit à placer de fausses informations dans un serveur cache, alors ces fausses informations pourront être transmises de manière tout à fait licite aux résolveurs. Ce type d'attaque porte le nom de *pollution de cache* ou *cache poisoning*.

La première phase d'une pollution de cache, est la récupération de l'identifiant de requête courant du serveur cache cible. Cela peut être réalisé grâce au paradoxe de l'anniversaire. La figure 4 présente une attaque par pollution de cache avec une variante dans la récupération de l'identifiant de requête courant. Il faut pour cela que l'attaquant possède un serveur de noms.

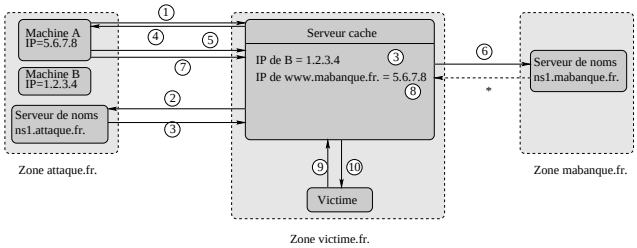

Fig. 4. Attaque par pollution de cache.

Le déroulement de l'attaque est le suivant : la machine A interroge le serveur cache de la zone `victime.fr.` ① en demandant l'adresse IP d'une machine se trouvant dans la zone `attaque.fr.`, la machine B dans notre exemple. Le serveur cache de la zone `victime.fr.` n'ayant pas la réponse, fait suivre la requête au serveur de noms de la zone `attaque.fr.` ② qui va lui donner l'adresse IP demandée ③. Le serveur cache de la zone `victime.fr.` conservera en mémoire l'adresse IP reçue et la fera suivre à la machine A ④. Cette dernière étape n'a pas d'importance dans l'attaque. À l'étape 2, l'attaquant ayant les droits nécessaires sur `ns1.attaque.fr.` est alors en possession de l'identifiant de requête courant du serveur cache de la zone `victime.fr.`.

L'attaque proprement dite commence à partir de la cinquième étape. A envoie une requête pour obtenir l'adresse de `www.mabanque.fr.` ⑤. Le serveur cache de la zone `victime.fr.` va interroger le serveur de noms autoritaire de la zone `mabanque.fr.` ⑥. Pendant ce temps la machine A inonde ⑦ le serveur cache de la zone `victime.fr.` avec des réponses à sa propre demande en indiquant que `www.mabanque.fr.` possède l'adresse IP de A. Ces requêtes successives sont identiques, à l'exception de l'identifiant qui est incrémenté à chaque réponse envoyée, l'identifiant

de départ étant celui récupéré lors de l'étape 4. Une fois le bon identifiant atteint, le serveur de noms accepte ce message comme étant la réponse correcte et la garde en mémoire ⑧. À partir de cet instant, la réponse du serveur de noms de la zone `mabanque.fr.` sera ignorée (étape *).

Cette attaque est généralement couplée à un déni de service sur le serveur de la zone `mabanque.fr.` pour le ralentir.

Dès lors que le cache est pollué, lorsqu'une machine de la zone `victime.fr.` demande l'adresse de `www.mabanque.fr.` à son serveur cache ⑨, celui-ci examine sa mémoire et renvoie l'adresse IP de A ⑩. La machine victime, alors convaincue d'avoir obtenu le site web de sa banque, dialogue avec l'attaquant.

D. Corruption de mise à jour dynamique et de transfert de zone

Lorsque des modifications de topologie ont lieu dans la zone (nouvelle adresse attribuée, présence d'une nouvelle machine, etc.), le serveur primaire en est informé par un message *Dynamic Update* [22]. Tout comme les requêtes et les réponses DNS, les messages de mise à jour sont vulnérables. Il y a deux manières d'exploiter ces vulnérabilités : en interceptant ou en créant un message de mise à jour. Les méthodes d'attaque par interception de messages sont similaires à celles présentées précédemment.

Si l'attaque se fait par la création d'un message, l'attaquant devra masquer son adresse IP et se faire passer pour une machine ayant le droit d'effectuer une mise à jour, comme le serveur DHCP par exemple.

Si cette usurpation d'identité réussit, le serveur primaire met à jour sa base de données en fonction des informations contenues dans le message. Les attaques visant à falsifier les messages de mise à jour DNS ont un impact important : en modifiant les données contenues dans le fichier de zone du serveur primaire, les données corrompues se propagent grâce au fonctionnement normal du DNS et atteignent ainsi les serveurs secondaires, les serveurs caches et les résolveurs.

Il est possible d'intervenir de la même manière sur les messages de transfert de zone. Les méthodes d'attaque sont les mêmes que celles présentées ci-dessus pour les messages de mise à jour : l'interception de message et la création d'un message de transfert de zone. Dans les deux cas, le but est de fournir de fausses informations à un serveur secondaire qui les conservera comme copie du fichier de zone et s'en servira pour répondre aux requêtes DNS qu'il recevra.

E. Attaques utilisant un serveur de noms

Les serveurs de noms hébergeant le fichier de zone ou une copie de celui-ci sont des cibles privilégiées. Une attaque peut être menée lorsque l'attaquant possède les droits nécessaires sur un serveur de noms autoritaire, soit parce qu'il les a usurpés ou soit tout simplement parce qu'il en est l'administrateur. Ce type d'attaque a été décrit par Steve M. Bellovin [4] et repris par Christoph L. Schuba [5]. Il est intéressant de noter que les travaux de Bellovin datent de 1990 mais n'ont été publiés qu'en 1995. Les raisons sont expliquées dans l'épilogue de [4] : l'auteur avait retardé la publication de ses travaux car ils décrivaient une sérieuse vulnérabilité qui n'avait à l'époque pas de correctif.

Dans son article, Bellovin montre la faiblesse des "r-commandes" de Berkeley du type `rlogin`, `rsh`, etc. Avant que Berkeley ne produise un correctif pour combler une déficience du démon `rlogin`, celui-ci se servait de l'adresse IP de la machine qui voulait se connecter à distance, pour retrouver le nom de cette machine (*reverse* DNS). Il vérifiait ensuite si la machine ayant ce nom avait le droit de se connecter. Il était alors facile pour quelqu'un disposant d'un serveur DNS, et donc gérant son fichier de zone et son *reverse* DNS, de modifier dans le *reverse* le nom correspondant à l'adresse IP d'une machine en sa possession. Cette attaque fournissait ainsi une fausse identité de la machine et permettait de s'introduire dans d'autres systèmes.

Grâce au correctif, le démon `rlogin` effectue maintenant une double vérification. Il obtient le nom associé à l'adresse de la machine qui veut se connecter puis il interroge le DNS en utilisant ce nom. L'adresse reçue en réponse est alors comparée à celle obtenue précédemment. Si les deux couples nom-IP obtenus sont identiques alors le démon vérifie le droit de connection, sinon il refuse l'accès.

IV. Les extensions de sécurité du DNS (DNSSEC)

Aujourd'hui, le protocole DNS étant incontournable dans l'utilisation des réseaux, l'apport de services de sécurité pour le protéger est essentiel. Dans sa conception originelle, DNS n'intégrait aucun service de sécurité. C'est pourquoi les extensions de sécurité du DNS ont été créées : il s'agit de DNSSEC [8], [9], [10], [11], [12].

Pour fournir l'intégrité et l'authentification des données DNS, DNSSEC s'appuie sur l'utilisation de signatures numériques. DNSSEC définit donc de nouveaux enregistrements de ressource afin de conserver les clés publiques dans les fichiers de zone, ainsi que les signatures numériques des enregistrements de ressource.

DNSSEC garde la compatibilité entre les anciens équipements qui ne supportent pas DNSSEC et ceux qui supportent ces extensions. En particulier, DNSSEC reprend le même format de données et de messages que DNS. Lorsqu'un serveur ou un client DNSSEC reçoit un enregistrement qu'il ne connaît pas, il l'ignore simplement.

Nous illustrons dans le paragraphe IV-A la signature d'un fichier de zone par un exemple, puis nous utilisons cet exemple pour détailler, dans les sections IV-B et IV-C, les nouveaux enregistrements de ressource propre à DNSSEC.

A. Signature du fichier de zone : principes et fonctionnement

L'ajout de signatures pour chaque enregistrement augmenterait de manière non négligeable la taille du fichier de zone. Pour limiter cette augmentation, les enregistrements de ressource ayant le même nom, le même type et la même classe sont regroupés en un *Resource Record set* ou *RRset*. Les signatures portent alors sur chaque RRset et sont placées dans des enregistrements RRSIG. Notons qu'une zone peut posséder plusieurs clés, auquel cas chaque RRset doit être

signé par chacune des clés (voir le paragraphe IV-C.2 pour plus de détails).

Ainsi, si une zone possède trois paires de clés et donc trois enregistrements DNSKEY, après signature du fichier de zone, cela représentera six enregistrements : trois DNSKEY RR (le DNSKEY RRset) et trois RRSIG RR. Sans le regroupement en RRset, il y aurait eu trois signatures par DNSKEY RR soit au total douze enregistrements.

La figure 5 montre une partie du fichier de zone de la zone `irisa.idsa.prd.fr.` avant et après signature. Nous pouvons y voir l'augmentation de taille malgré la présence de RRset (NS RRset, DNSKEY RRset).

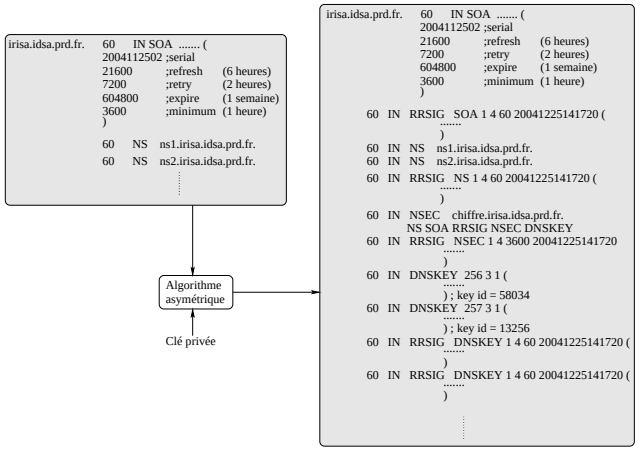

Fig. 5. Signature d'un fichier de zone.

Il existe deux exceptions à la signature : les enregistrements RRSIG ne sont pas signés ni les *Glue Records*. En effet, les *Glue Records* (enregistrements NS et A d'une délégation) sont des informations d'une zone fille dupliquée dans sa zone parente. La zone parente n'étant pas propriétaire des *Glue Records*, elle ne doit pas les signer.

B. Les nouveaux enregistrements de ressource

DNSSEC assure lui-même la diffusion des clés publiques de zone et des signatures numériques. DNSSEC définit deux nouveaux enregistrements : il s'agit de l'enregistrement DNSKEY [11] pour conserver les clés publiques (anciennement appelé KEY [8]) et de l'enregistrement RRSIG [11] pour conserver les signatures numériques (anciennement appelé SIG [8]), que nous détaillons dans les deux paragraphes suivants.

1) L'enregistrement DNSKEY : Une zone DNSSEC possède au moins une paire de clés publique/privée. La partie privée permet de générer les signatures numériques des enregistrements DNS contenus dans le fichier de zone. La partie publique est conservée dans un enregistrement DNSKEY. Cet enregistrement contient aussi toutes les informations nécessaires à l'utilisation de cette clé. Un résolveur recevant les enregistrements DNSKEY d'une zone peut alors vérifier les signatures des enregistrements de la zone qui ont été générées grâce à ces clés. La structure arborescente de DNS permet de distribuer les clés publiques de manière efficace.

Une clé n'a pas de durée de vie particulière. Elle reste présente dans le fichier de zone tant que son administrateur ne décide pas de la changer. [23] donne des recommandations sur la fréquence de renouvellement des clés de zone et sur leur longueur. La figure 6 représente le format d'un enregistrement DNSKEY. Celui-ci contient quatre champs : *Flags*, *Protocol*, *Algorithm* et *Public Key*.

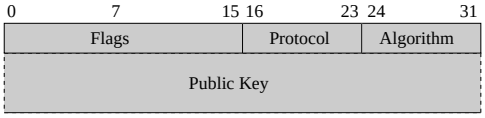

Fig. 6. L'enregistrement DNSKEY.

Seuls deux bits du champs *Flags* sont actuellement utilisés : il s'agit du bit 7 et du bit 15. Lorsque le bit 7 est positionné, cela signifie que l'enregistrement contient une clé de zone DNS. Le nom du propriétaire de l'enregistrement doit être le nom de la zone. Si ce bit a une valeur de zéro, l'enregistrement contient un autre type de clé publique DNS qui ne doit pas être utilisée pour vérifier les signatures des enregistrements.

Le bit 15 définit le *Secure Entry Point Flag* ou point d'entrée sécurisé [14]. Lorsque ce bit est positionné, il signifie que la clé contenue dans l'enregistrement DNSKEY peut être utilisée comme point d'entrée sécurisé. La notion de point d'entrée sécurisé est détaillée dans la section IV-C.

Le champ *Protocol* doit toujours avoir une valeur de 3. Tout enregistrement DNSKEY ayant une valeur différente pour ce champ sera rejeté. La valeur 3 indique que la clé sera utilisée par le protocole DNSSEC. Lors de la première standardisation de DNSSEC [8], cet enregistrement pouvait contenir des clés utilisables par n'importe quel protocole : IPsec, SSH, *etc*. Mais le RFC 3445 [24] a restreint l'utilisation de cet enregistrement pour les clés utilisées par le protocole DNSSEC.

Le champ *Algorithm* définit l'algorithme cryptographique associé à la clé publique et définit ainsi son format. Par exemple, la valeur 1 signifie que l'algorithme RSA est utilisé, la valeur 3 représente l'algorithme DSA. La clé publique est placée dans le champ *Public Key*.

La figure 7 présente le premier enregistrement DNSKEY issu du fichier de la zone `irisa.idsa.prd.fr.` utilisé lors du projet IDsA [25].

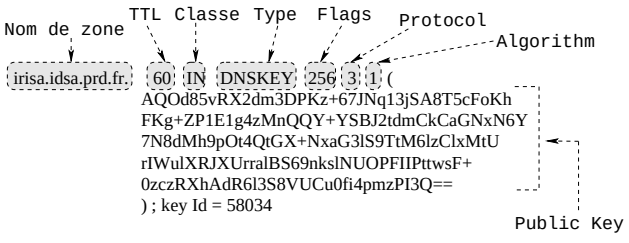

Fig. 7. Un exemple d'enregistrement DNSKEY.

Nous y trouvons le nom du propriétaire : la zone `irisa.idsa.prd.fr.`. Le TTL indique le temps pendant lequel un serveur cache qui reçoit cet enregistrement peut le conserver (60 secondes). Nous distinguons aussi les valeurs des champs présentés ci-dessus. La valeur 1 du champ *Algorithm* signifie que la clé contenue dans cet enregistrement est

une clé RSA. L'ensemble de ces informations permet aussi de générer un identifiant de clé ou *Key Id*.

2) L'enregistrement RRSIG : Les signatures numériques générées à l'aide des clés privées de la zone doivent être présentes dans le fichier de zone pour être disponibles. C'est le rôle de l'enregistrement RRSIG.

Un enregistrement RRSIG contient les informations nécessaires à l'identification de l'enregistrement auquel il est associé, ainsi que les informations permettant d'identifier la clé ayant généré la signature numérique. La figure 8 présente le format d'un enregistrement RRSIG.

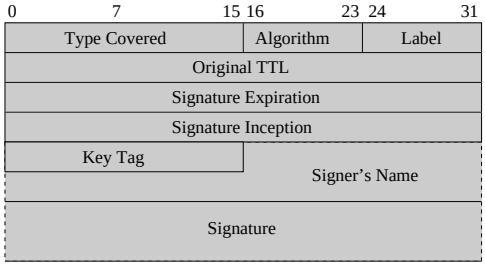

Fig. 8. L'enregistrement RRSIG.

Le champ *Type Covered* indique le type de l'enregistrement couvert par cette signature, s'il s'agit par exemple d'un enregistrement A ou NS. Le champ *Algorithm* précise l'algorithme utilisé pour générer la signature. Les valeurs sont les mêmes que pour l'enregistrement DNSKEY : 1 représente l'algorithme RSA, *etc*. Le champ *Label* indique le nombre d'étiquettes que comportait le nom associé à l'enregistrement RRSIG lors de la création de la signature. Cette indication permet de vérifier si la réponse a été générée à partir d'un *wildcard*.

Un *wildcard* permet de définir une réponse par défaut. La figure 9 présente un exemple de *wildcard* pour un enregistrement A concernant les noms terminés par `wildcard.irisa.idsa.prd.fr`. Lorsqu'une requête parvient à un serveur de noms, celui-ci regarde si le nom demandé est présent dans son fichier de zone pour la ressource demandée. Si ce n'est pas le cas le serveur regarde si un *wildcard* correspond au nom demandé ; si un *wildcard* existe, le nom du *wildcard* est modifié dans la réponse pour correspondre au nom contenu dans la requête.

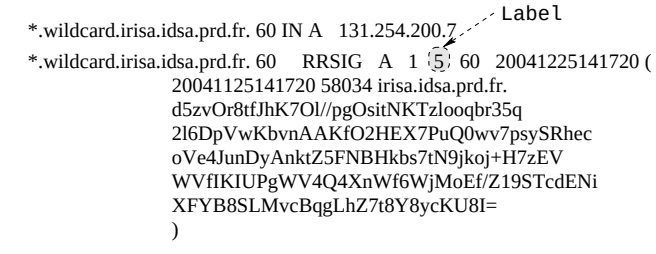

Fig. 9. Un exemple de wildcard et sa signature.

Sur notre exemple en supposant que le nom `test.wildcard.irisa.idsa.prd.fr.` n'existe pas dans la zone, une requête demandant l'adresse associée à ce nom aura pour réponse le *wildcard* dont le caractère «*» aura été remplacé par «test». Nous remarquons bien ici que le nom reçu (`test.wildcard.irisa.idsa.prd.fr.`) et le nom du *wildcard* (`*.wildcard.irisa.idsa.prd.fr.`) sont différents. C'est la valeur du champ *Label* qui va permettre la vérification de la signature. La valeur 5 de ce champ dans notre exemple indique que la signature a été générée à partir des 5 derniers labels du nom soit `wildcard.irisa.idsa.prd.fr.`. Grâce au champ *Label*, le résolveur déduit qu'il faut vérifier la signature en employant seulement les cinq dernières étiquettes du nom `test.wildcard.irisa.idsa.prd.fr.`.

Chaque enregistrement de ressource possède un *Time To Live* (TTL) associé. Ce TTL est positionné par l'administrateur de la zone. Lors de la signature d'un enregistrement, son TTL est pris en compte dans la signature. Le champ *Original TTL* contient la valeur du TTL de l'enregistrement couvert par la signature lors de la génération de celle-ci. Dès qu'un enregistrement est placé dans une réponse ou dans la mémoire d'un serveur cache, son TTL diminue. La valeur du TTL originel contenu dans l'enregistrement RRSIG permet ainsi de pouvoir vérifier les signatures en remplaçant la valeur du TTL courant par la valeur du TTL originel lors de la vérification.

Les champs *Signature Inception* et *Signature Expiration* permettent de définir la période de validité de la signature. Ces champs contiennent une valeur sur 32 bits représentant le nombre de secondes écoulées depuis le premier janvier 1970 à minuit. Lorsqu'un résolveur reçoit une signature dont la période de validité est dépassée ou au contraire n'est pas encore atteinte, il la rejette.

Le champ *Key Tag* contient l'identifiant de la clé (la valeur du key Id) qui a généré la signature et le champ *Signer's Name* identifie le propriétaire de la clé qu'un résolveur doit utiliser pour vérifier la signature. Ce champ contient le nom de la zone possédant l'enregistrement couvert par la signature.

Enfin, le champ *Signature* contient la signature de l'enregistrement et des données de signature, excepté le champ *Signature* lui-même. La figure 10 montre l'enregistrement RRSIG contenant la signature de l'enregistrement SOA issu du fichier de la zone `irisa.idsa.prd.fr.`. L'ensemble des informations permet de déduire qu'il s'agit de la signature de l'enregistrement SOA de la zone `irisa.idsa.prd.fr`, 58034 indique l'identifiant de la clé et `irisa.idsa.prd.fr.` est le nom du signataire.

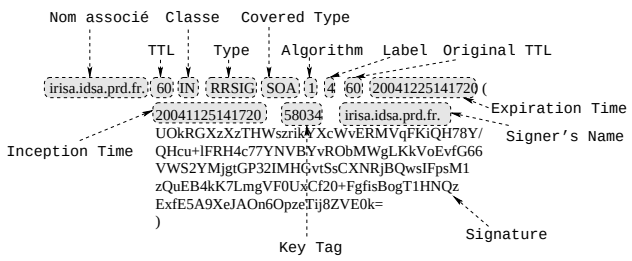

Fig. 10. Un exemple d'enregistrement RRSIG.

Pour vérifier cette signature, le résolveur doit donc disposer de la clé publique ayant l'identifiant 58034 ainsi que de l'enregistrement SOA de la zone `irisa.idsa.prd.fr.`.

Ces deux enregistrements supplémentaires, DNSKEY et RRSIG, mettent à disposition les clés publiques de la zone ainsi que les signatures des enregistrements. Les signatures numériques protègent les enregistrements qu'elles couvrent. Elles ne protègent donc que les réponses positives, c'est-à-dire les réponses contenant des enregistrements et leurs signatures. Mais lorsqu'il s'agit d'une réponse négative, c'est-à-dire que l'enregistrement demandé n'existe pas, la réponse envoyée par le serveur de noms est vide : elle contient juste un code d'erreur tel que NXRR (enregistrement inexistant) ou NXDOMAIN (nom inexistant). La réponse étant vide il n'est pas possible de la sécuriser grâce à des signatures numériques existant dans le fichier de zone. C'est pourquoi l'enregistrement NSEC (pour Next SECure) [26] a été créé (anciennement appelé NXT [8]).

3) L'enregistrement NSEC : Dans le protocole DNS, lorsqu'une requête porte sur une ressource ou un nom de domaine qui n'existe pas, la réponse contient uniquement un code d'erreur dans son en-tête et cet en-tête n'est pas protégé.

L'enregistrement NSEC a été créé pour pouvoir vérifier les réponses négatives. Un enregistrement NSEC contient les informations nécessaires à l'identification des enregistrements existants pour un nom donné, ainsi que le prochain nom existant dans la zone (dans l'ordre lexicographique). Ces deux informations suffisent pour prouver qu'un enregistrement ou qu'un domaine n'existe pas. La figure 11 montre le format d'un enregistrement NSEC.

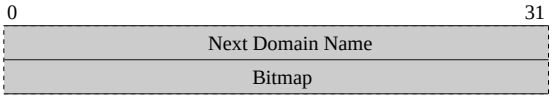

Fig. 11. L'enregistrement NSEC.

Le champ *Next Domain* contient le prochain nom dans la zone. Le champ *Bitmap* représente chacun des types d'enregistrements présents dans le fichier de zone pour le nom associé à l'enregistrement NSEC.

Lorsqu'un résolveur reçoit un enregistrement NSEC, il vérifie sa signature, puis regarde le nom associé. Si ce n'est pas le nom qu'il a demandé, mais que le nom demandé est dans l'intervalle [*Nom associé*,..,*Next Domain Name*] alors le résolveur déduit que le nom demandé n'existe pas. L'enregistrement NSEC permettant de vérifier qu'un domaine donné n'existe pas est appelé le *NSEC couvrant*. Il permet de prouver les réponses contenant le code d'erreur NXDOMAIN.

Les enregistrements NSEC permettent aussi de prouver qu'un enregistrement donné n'existe pas. Lorsqu'un résolveur reçoit un enregistrement NSEC pour le nom qu'il a demandé, il regarde dans le champ *Bitmap* si le type d'enregistrement demandé est présent. Si ce type ne s'y trouve pas alors l'enregistrement demandé n'existe pas. L'enregistrement NSEC étant protégé par une signature numérique, nous pouvons prouver que les données qu'il contient n'ont pas été modifiées.

Un exemple d'enregistrement NSEC est présenté sur la figure 12. Cet enregistrement est le premier enregistrement NSEC issu du fichier de zone présenté sur la figure 5 :

Cet enregistrement spécifie qu'il n'existe pas de nom entre `irisa.idsa.prd.fr.` et le prochain nom `chiffre.irisa.idsa.prd.fr.` et que les types d'enregistrements existants pour le nom `irisa.idsa.prd.fr.` sont NS, SOA, RRSIG, NSEC, DNSKEY.

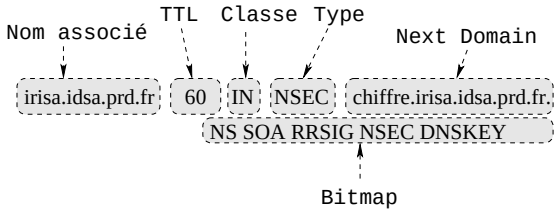

Fig. 12. Un exemple d'enregistrement NSEC.

Ainsi un résolveur demandant l'adresse de `a.irisa.idsa.prd.fr.` recevra cet enregistrement NSEC signé. Après avoir vérifié la signature, le résolveur vérifiera que le nom demandé est dans l'intervalle [`irisa.idsa.prd.fr.`,..,`chiffre.irisa.idsa.prd.fr.`]. De même s'il demande l'enregistrement A associé au nom `irisa.idsa.prd.fr.`, il vérifiera que le type A n'apparaît pas dans le *Bitmap*.

L'enregistrement NSEC permet de prouver les réponses négatives, mais il amène aussi un autre type d'attaque baptisé *NXT walking*. Comme l'enregistrement NSEC contient deux noms existants dans la zone, si une personne récupère tous les enregistrements NSEC d'une zone, cette personne possède alors tous les noms existants dans cette zone. Il n'est pas du goût de tous les administrateurs d'avoir la possibilité de cartographier complètement leur zone, par exemple la zone `mil.` (armée américaine). Une proposition pour éviter une telle attaque est actuellement à l'étude [27]. Cette solution propose de hacher tous les noms de la zone et d'ordonner les enregistrements NSEC selon l'ordre lexicographique sur les hachés. Le nom associé à un NSEC et le nom contenu dans le champ *Next Domain* sont donc des hachés. Un serveur recevant une requête portant sur un nom qui n'existe pas, produira un haché de ce nom et répondra avec l'enregistrement NSEC dont l'intervalle contient le haché obtenu. En utilisant les hachés plutôt que les noms et comme la fonction de hachage utilisée est par définition non-inversible, il est impossible d'obtenir le nom à partir du haché et donc de cartographier une zone.

Nous venons de voir trois enregistrements supplémentaires utilisés par DNSSEC pour assurer l'intégrité et l'authenticité des données DNS, contenues dans le fichier de zone. Ces enregistrements remplissent donc bien les objectifs de DNSSEC au niveau d'une zone. Néanmoins, comme pour toute application utilisant la cryptographie à clé publique, il faut avoir confiance en un point d'entrée.

C. Chaîne de confiance et authentification de clés en cascade

Dans DNSSEC, avoir confiance en une clé est ce que nous appelons posséder un point d'entrée sécurisé (SEP) [14] ou une clé de confiance. Cette clé de confiance est une clé publique d'une zone et est configurée par l'administrateur du résolveur. Le résolveur fait alors confiance à cette clé et il peut ainsi vérifier la signature du DNSKEY RRset de la zone grâce à sa clé de confiance. Une fois la signature vérifiée, le résolveur

fait confiance aux clés se trouvant dans le DNSKEY RRset et peut ainsi vérifier les signatures des autres enregistrements de la zone grâce aux clés contenues dans le DNSKEY RRset. Ainsi avec une clé de zone configurée comme clé de confiance, un résolveur peut, après avoir vérifié les signatures, étendre sa confiance à toutes les clés de la zone.

En suivant ce modèle, il faudrait configurer dans un résolveur une clé de confiance pour chaque zone, ce qui est impossible car il existe plusieurs millions de zones dans l'arbre DNS (plus de 30 millions au quatrième trimestre 2004). Pour résoudre ce problème, DNSSEC tire partie de la structure arborescente de l'architecture DNS. Un nouvel enregistrement, l'enregistrement *Delegation Signer* (DS) [9], est placé au point de délégation. Cet enregistrement, conservé dans le fichier de zone de la zone parente, permet d'identifier une clé de sa zone fille.

1) L'enregistrement DS : Un enregistrement DS référence un enregistrement DNSKEY, il est utilisé pour authentifier une clé. L'enregistrement DS contient les informations nécessaires à l'identification d'une clé, telles que le *Key Tag*, l'algorithme utilisé et un haché de la clé concernée. En obtenant l'enregistrement DNSKEY référencé par un enregistrement DS, puis en vérifiant la signature de cet enregistrement DS, un résolveur peut authentifier la clé contenue dans l'enregistrement DNSKEY.

L'enregistrement DS et l'enregistrement DNSKEY possèdent le même nom de propriétaire, mais l'enregistrement DS est toujours conservé dans la zone parente au point de délégation. Par exemple, l'enregistrement DS pour `irisa.idsa.prd.fr.` est conservé dans la zone `idsa.prd.fr.` tandis que l'enregistrement DNSKEY est conservé dans la zone fille `irisa.idsa.prd.fr.`. La figure 13 décrit le format d'un enregistrement DS.

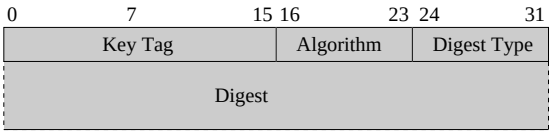

Fig. 13. L'enregistrement DS.

Le champ *Key Tag* contient l'identifiant de la clé référencée dans l'enregistrement DS. Le champ *Algorithm* est identique au champ *Algorithm* de l'enregistrement DNSKEY. Le champ *Digest Type* contient l'algorithme utilisé pour générer le haché de l'enregistrement DNSKEY. Ce haché est placé dans le champ *Digest*. Il s'agit en fait d'un haché de la concaténation du nom du propriétaire de l'enregistrement DNSKEY et des champs *Flags*, *Protocol*, *Algorithm* et *Public Key* de cet enregistrement.

La figure 14 montre un enregistrement DNSKEY et son enregistrement DS associé. L'enregistrement DNSKEY se trouve dans la zone `irisa.idsa.prd.fr.` et l'enregistrement DS se trouve dans la zone `idsa.prd.fr.`.

Lorsqu'une zone DNS déploie DNSSEC, elle doit dans un premier temps créer ses clés de zone et signer son fichier de zone. Puis, l'administrateur de cette nouvelle zone sécurisée doit contacter l'administrateur de sa zone parente afin de créer

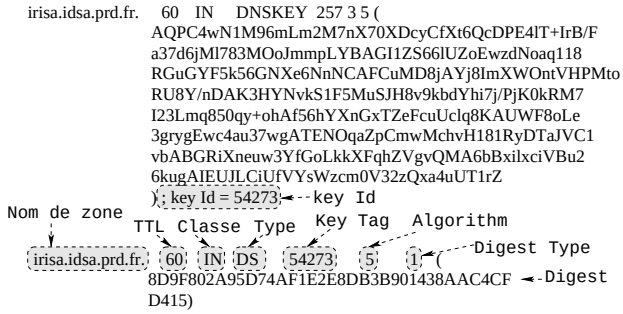

Fig. 14. Un exemple d'enregistrement DS.

les enregistrements DS associés aux clés de la nouvelle zone sécurisée.

Le problème du démarrage est qu'il n'existe pas encore de lien de confiance entre la zone parente et la zone fille. Les deux administrateurs doivent alors utiliser un autre moyen pour s'authentifier mutuellement (certificats X.509, formulaires administratifs, *etc.*).

2) Les deux types de clés, ZSK et KSK : L'enregistrement DS crée un lien entre une zone parente et une zone fille en référençant une de ses clés. Le modèle *Delegation Signer* introduit donc une distinction entre les clés : les clés ayant un enregistrement DS associé, appelées *Key Signing Key* (KSK) et les clés n'ayant pas d'enregistrement DS associé, appelées *Zone Signing Key* (ZSK).

La création d'un enregistrement DS nécessitant une interaction entre les deux zones (échange de clés de la zone fille vers la zone parente, puis création de l'enregistrement DS et de sa signature), la distinction entre ces deux types de clés a été faite pour minimiser la charge de travail induite par la création et la mise à jour des enregistrements DS.

En effet, les ZSK signent tous les enregistrements d'une zone. Comme les ZSK n'ont pas d'enregistrement DS associé, lorsque une zone décide de changer une de ses ZSK, il n'y a aucun travail supplémentaire pour sa zone parente, ni d'échange entre les deux zones.

Les KSK ne signent que les enregistrements DNSKEY. Comme une KSK possède un enregistrement DS associé, lorsqu'une zone change une de ses KSK, il doit y avoir un échange d'informations entre cette zone et sa zone parente afin de mettre à jour l'enregistrement DS correspondant.

Le document [23] compile un certain nombre d'informations issues de différentes expérimentations et fournit un certain nombre de conseils, notamment sur la période de renouvellement des clés, le nombre de clés à utiliser, leur taille, la fréquence de renouvellement de ces clés en fonction de leur taille, *etc.* [23] conseille d'utiliser une KSK pendant quelques mois avant de la renouveler et de renouveler une ZSK après un mois d'utilisation. Dans ce document se trouve aussi un tableau récapitulant la longueur de clé à utiliser dans les trente prochaines années.

3) L'authentification de clés en cascade : L'ensemble des éléments présentés ci-dessus (un point d'entrée sécurisé (SEP), un enregistrement DS authentifiant une clé de la zone fille qui est elle-même signée, *etc.*), permet de construire une *chaîne de confiance* [13]. Le début de la chaîne est le SEP,

chaque maillon est constitué d'un DNSKEY RRset, dont au moins une signature générée par une KSK est vérifiée et d'un enregistrement DS dont au moins une signature générée par une ZSK est vérifiée.

Le lien entre les maillons est représenté par un enregistrement DS authentifiant une KSK qui a signé le DNSKEY RRset. Le dernier maillon de la chaîne contient la ressource demandée et son enregistrement RRSIG associé. La figure 15 représente cette authentification en cascade.

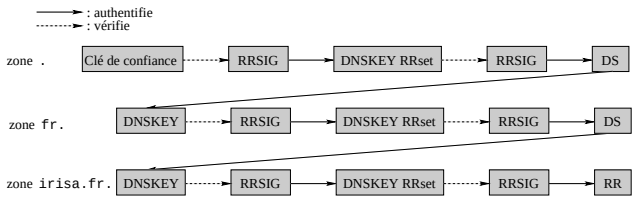

Fig. 15. L'authentification de clés en cascade.

À chaque maillon de la chaîne, nous retrouvons le même schéma : un enregistrement DS permet d'avoir confiance en une KSK, une KSK permet d'avoir confiance en l'ensemble des clés de la zone (KSK et ZSK) et les ZSK permettent alors d'avoir confiance dans les autres enregistrements.

Bien que le modèle *Delegation Signer* introduise une distinction au niveau des clés, il n'y a pas d'obligation au niveau des zones sur l'utilisation de ces deux types de clés. Nous pouvons donc nous attendre à trouver, dans les petites zones, une clé ayant les deux fonctions (KSK et ZSK), c'est-à-dire que cette clé signe tous les enregistrements de la zone et aura un DS associé dans sa zone parente.

D. La résolution de noms DNSSEC

Le principe de résolution de noms reste le même pour DNSSEC. Une des contraintes lors de la conception des extensions de sécurité du DNS était la compatibilité entre DNS et DNSSEC : un équipement ne comprenant pas DNSSEC doit être en mesure d'effectuer des résolutions de noms DNS sans avoir de problèmes. Le comportement d'un ancien équipement, face à des enregistrements qu'il ne connaît pas, est tout simplement de les ignorer.

Le schéma d'échange des messages reste donc le même, les entités communicantes sont les mêmes que sur la figure 2. Le contenu des réponses est plus important car il contient des enregistrements supplémentaires.

Un serveur de noms sécurisé (possédant un fichier de zone signé) va inclure dans ses réponses le matériel cryptographique nécessaire, tels que ses enregistrements DNSKEY. De plus, un enregistrement est toujours envoyé avec ses signatures associées.

De la même manière, si la réponse est une délégation, le serveur sécurisé enverra en réponse les enregistrements DS adéquats et leurs signatures. Si à un moment de la résolution sécurisée, il manque au serveur cache ou au résolveur certains enregistrements de sécurité, une requête spécifique sera envoyée pour récupérer les enregistrements manquants.

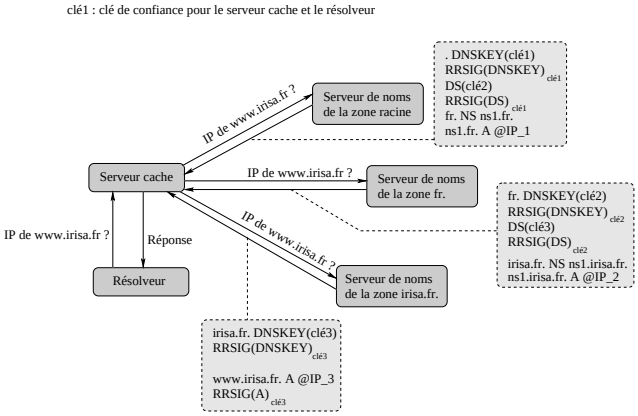

Fig. 16. La résolution de noms sécurisée.

La figure 16 montre une résolution de noms sécurisée et une partie du contenu des messages échangés. Les interactions entre les entités sont les mêmes que lors d'une résolution de noms DNS (*cf.* figure 2), mais cette fois-ci les messages contiennent le matériel cryptographique nécessaire à la vérification de signatures et à l'établissement d'une chaîne de confiance.

Deux algorithmes peuvent être utilisés pour effectuer la résolution de noms sécurisée : il s'agit des algorithmes *Top-Down* et *Bottom-Up*.

La première étape de ces deux algorithmes est la même : récupérer la ressource demandée, comme présenté sur la figure 2, sans se soucier du matériel cryptographique pour le moment. Si la ressource se trouve dans une zone sécurisée, alors la réponse contiendra la ressource et sa signature. Nous montrons dans les deux paragraphes suivants le fonctionnement de ces deux algorithmes dans l'exemple présenté sur la figure 16. Nous voulons obtenir l'adresse IP associée à `www.irisa.fr.`.

1) L'algorithme Top-Down :

1) Nous obtenons la ressource demandée grâce à une requête DNS. La ressource est dans une zone sécurisée, sa signature est donc présente dans la réponse. Nous avons donc sur notre exemple, un enregistrement A et un enregistrement RRSIG pour `www.irisa.fr.`.
2) L'algorithme regarde le nom du signataire de l'enregistrement, présent dans le champ *Signer's Name* de l'enregistrement RRSIG, il s'agit de `irisa.fr.`. Puis, il cherche la clé de confiance la plus proche de ce nom, qu'il possède. Cette recherche est effectuée en ôtant une par une les étiquettes de gauche du nom du signataire, jusqu'à obtenir le nom associé à une clé de confiance, ici nous obtenons *clé1* comme clé de confiance la plus proche.
3) *clé1* est la clé de la zone racine, nous demandons donc au serveur de noms de la zone racine ses enregistrements DNSKEY. L'algorithme utilise alors *clé1* pour vérifier les signatures. Il fait désormais confiance à toutes les clés contenues dans le DNSKEY RRset de la zone racine.
4) L'algorithme demande les clés de la zone `irisa.fr.`.

La racine ne possède pas cette information, elle transmet donc la délégation appropriée, les enregistrements DS de `fr.` et leurs signatures, ainsi que les *Glue Records*. L'algorithme utilise alors le DNSKEY RRset obtenu précédemment pour vérifier les signatures des enregistrements DS.

5) Puis l'algorithme interroge le serveur de noms de `fr.` et obtient de la même manière le DNSKEY RRset de `fr.` et la délégation `irisa.fr.`. L'algorithme vérifie qu'un enregistrement DS de `fr.` authentifie une clé du DNSKEY RRset de la zone `fr.`, puis que cette clé vérifie une signature du DNSKEY RRset de la zone `fr.`.
6) L'algorithme vérifie ensuite les signatures du DS de `irisa.fr.` et interroge le serveur de noms de `irisa.fr.` fourni dans la délégation.
7) L'algorithme obtient et vérifie le lien entre une clé et un enregistrement DS, et vérifie alors les signatures du DNSKEY RRset de la zone `irisa.fr.` avec cette clé.
8) Et finalement, ayant obtenu les clés du signataires de la ressource, l'algorithme vérifie les signatures de la ressource demandée.

À chaque étape cet algorithme valide une ressource qui vient augmenter l'ensemble des ressources en lesquelles il a confiance.

2) L'algorithme Bottom-Up : L'algorithme *Bottom-Up* est orienté signature, c'est-à-dire qu'au lieu de partir d'une clé de confiance pour arriver à la ressource, il va regarder qui a signé la ressource et remonter jusqu'à une clé de confiance.

1) Nous obtenons la ressource demandée grâce à une requête DNS. La ressource est dans une zone sécurisée, sa signature est donc présente dans la réponse. Nous avons donc sur notre exemple, un enregistrement A et un enregistrement RRSIG de `www.irisa.fr.`.
2) L'algorithme regarde le nom du signataire de l'enregistrement présent dans le champ *Signer's Name* de l'enregistrement RRSIG ; il s'agit de `irisa.fr.`. Il demande donc les clés de `irisa.fr.` et obtient le DNSKEY RRset de la zone `irisa.fr.` et ses signatures.
3) L'algorithme regarde si une des clés reçues est une clé de confiance, ce n'est pas le cas. Il demande donc les enregistrements DS existant pour la zone `irisa.fr.`. La zone `fr.` lui répond avec les enregistrements DS et leurs signatures.
4) L'algorithme regarde le nom du signataire des enregistrements DS, il s'agit de `fr.`, il demande donc les clés de `fr.`.
5) Il compare ces clés avec ses clés de confiance, aucune ne correspond il continue donc et demande les enregistrements DS pour `fr.`. La zone racine lui répond avec les enregistrements DS signés.
6) L'algorithme regarde le nom du signataire des enregistrements DS, il s'agit de la racine, il demande donc les clés de la zone racine.
7) Il compare ces clés avec ses clés de confiance ; une clé correspond : il s'agit de *clé1*. La vérification des signatures peut donc commencer.
8) L'algorithme vérifie les signatures des clés de la zone racine, puis les signatures du DS de `fr.`. Il vérifie l'existence d'un lien entre un enregistrement DS et une clé de `fr.`. Puis il vérifie les signatures des clés de `fr.` et les signatures des enregistrements DS de `irisa.fr.`. Il vérifie l'existence d'un lien entre un enregistrement DS et une clé de `irisa.fr.`. Puis il vérifie les signatures des clés de `irisa.fr.`. Finalement, il vérifie les signatures de la ressource obtenue à l'étape 1.

Ce qui différencie ces deux algorithmes est le sens de récupération des informations. La chaîne de confiance est toujours établie de la même manière. Nous pouvons remarquer que l'algorithme *Top-Down* se prête bien au comportement d'un serveur cache qui valide à la volée les enregistrements. De plus, lors de la première requête pour obtenir la ressource demandée, les serveurs autoritaires envoient généralement le matériel cryptographique nécessaire. Ceci implique donc que, lors de la première requête pour obtenir la ressource, notre serveur cache ayant suivi les délégations, il a déjà obtenu la majorité des informations nécessaires.

En revanche, pour un résolveur situé derrière son serveur cache récursif, lors de la première requête il obtient ce qu'il a demandé, même si son serveur cache a obtenu beaucoup plus d'informations. Pour ce type de résolveur [28], [29], l'utilisation de l'un ou l'autre algorithme est identique. Le trafic pour obtenir les enregistrements nécessaires se faisant entre le résolveur et le serveur cache qui normalement les possède déjà. Nous pouvons aussi remarquer une similarité entre l'utilisation de ces algorithmes et les résolutions de noms récursive ou itérative.

Dans les trois paragraphes suivants, nous présentons quelques éléments spécifiques utilisés durant une résolution de noms sécurisée : le bit *Authenticated Data* [30], le bit *Checking Disabled* [8] et l'extension EDNS0 [31].

3) Le bit Authenticated Data (AD) : Le bit AD [30] est un bit de l'en-tête des messages DNS. Ce bit est positionné dans une réponse uniquement si tous les enregistrements contenus dans cette réponse ont été cryptographiquement vérifiés.

Un résolveur DNSSEC appelé par une application peut lui retourner, grâce à la vérification des signatures, le statut de sécurité du RRset reçu. Néanmoins, certains résolveurs ne possèdent pas les ressources suffisantes pour effectuer la vérification cryptographique des enregistrements. Ces résolveurs font généralement confiance à leur serveur cache récursif qui effectue la vérification pour eux.

Le serveur cache utilise alors le bit AD dans l'en-tête de ses réponses pour indiquer que les enregistrements envoyés ont été vérifiés.

Un résolveur ne doit faire confiance au bit AD, positionné par son serveur cache, que s'il possède un canal d'échange sécurisé avec son serveur cache. Il faut aussi que la réponse soit passée par ce canal et que le résolveur soit explicitement configuré pour faire confiance au serveur cache. Ce canal sécurisé permet d'empêcher un attaquant de placer de faux enregistrements dans une réponse dont le bit AD est à 1, ce qui aurait pour conséquence que le résolveur fasse confiance

à cette fausse réponse.

4) Le bit Checking Disabled (CD) : Un autre bit de l'en-tête des messages DNS peut être utilisé par un résolveur pour spécifier un comportement particulier de la part du serveur qui reçoit la requête : il s'agit du bit *Checking Disable* (CD) [8].

Le bit CD signifie que l'entité ayant envoyé la requête accepte des enregistrements même si ceux-ci n'ont pas été cryptographiquement vérifiés par l'émetteur de la réponse. Ce bit est généralement utilisé par les résolveurs dont la politique de sécurité locale est d'effectuer eux-mêmes la vérification des signatures.

5) La taille des messages DNSSEC et l'extension EDNS0 : Historiquement, le protocole DNS fonctionne au-dessus du protocole UDP avec une taille de paquet de 512 octets. Cette limite était largement suffisante pour contenir une réponse DNS.

Néanmoins, pour une réponse devant contenir tout le matériel cryptographique nécessaire à DNSSEC, cette limite est très souvent dépassée. Pour pallier ce problème, DNSSEC possède un pseudo-enregistrement, il s'agit de EDNS0 [31] qui n'est pas conservé en cache. Ce pseudo-enregistrement est ajouté dans la partie optionnelle d'une requête : il permet de spécifier entre autre la taille de message que supporte l'émetteur de la requête et s'il supporte DNSSEC. L'entité recevant la requête peut alors adapter la taille de la réponse en fonction de cette information et ainsi placer dans sa réponse des enregistrements supplémentaires.

Un cas de dysfonctionnement assez subtil peut apparaître avec une mauvaise configuration de EDNS0, comme le montre la figure 17.

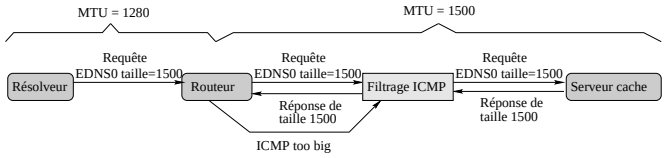

Fig. 17. Une mauvaise configuration de EDNS0.

Prenons un résolveur sur un brin de réseau dont le Maximum Transfert Unit (MTU) vaut 1280 octets et son serveur cache récursif sur un brin de réseau dont le MTU vaut 1500 octets. Un équipement filtrant le trafic ICMP existe entre le résolveur et le serveur cache. Si le résolveur indique une taille de 1500 octets en utilisant EDNS0 alors sa requête, de petite taille, parvient au serveur cache. Celui-ci utilise l'information de taille pour construire une réponse de 1500 octets. Cette réponse arrive au routeur qui indique alors que le paquet est trop gros grâce à un message *ICMP too big*. Ce message est filtré par l'équipement filtrant et n'arrive jamais au serveur cache.

Du point de vue du serveur cache tout s'est passé correctement et du point de vue du résolveur la réponse n'est jamais arrivée, il considère donc le serveur cache comme inaccessible. La mauvaise configuration de l'extension EDNS0 dans ce cas implique tout simplement un deni de service.

E. La sécurisation des transactions DNS

Les enregistrements de sécurité présentés jusqu'à présent permettent de protéger les enregistrements DNS en tant qu'unité, c'est-à-dire que toute manipulation d'un enregistrement peut être détectée grâce aux signatures numériques.

Néanmoins, ces signatures ne protègent pas l'ensemble d'un message DNS, l'en-tête des messages n'est pas protégé. Or, cet en-tête contient des informations comme le nombre d'enregistrements présents dans le message. Ainsi, même si grâce aux signatures numériques il n'est pas possible de modifier le contenu d'un enregistrement sans que cela ne soit détecté, il est toujours possible de retirer du message certains enregistrements ou bien d'en rejouer d'autres. De telles manipulations peuvent avoir de lourdes conséquences lorsqu'il s'agit de messages de mise à jour ou de transfert de zone.

Pour éviter de telles manipulations, il existe deux mécanismes assurant l'intégrité des messages envoyés. Il s'agit de TSIG [32] basé sur un chiffrement à clé secrète et de SIG(0) [33] basé sur un chiffrement à clé publique.

1) TSIG : TSIG [32] repose sur l'utilisation d'un secret partagé et de fonction de hachage à sens unique. Le problème de la cryptographie à clé secrète est que le nombre de clés nécessaires pour un groupe augmente de manière quadratique en fonction du nombre de membres. C'est pourquoi l'utilisation de TSIG est généralement limitée à la protection des transactions entre un serveur primaire et ses serveurs secondaires lors d'un transfert de zone, ou pour la protection des mises à jour dynamiques. Le nombre de serveurs présents dans la zone et d'entités autorisées à effectuer des mises à jour étant limité, le nombre de clés secrètes à gérer reste raisonnable.

Le contenu d'un enregistrement TSIG est obtenu par la combinaison d'un HMAC (*Keyed-Hashing for Message Authentication* [34]) et d'une fonction de hachage à sens unique. Cette combinaison est appliquée à l'intégralité du message DNS : enregistrements et en-tête ainsi qu'aux variables TSIG, telles que l'intervalle de temps pendant lequel l'enregistrement TSIG est utilisable. Comme un enregistrement TSIG est créé dynamiquement pour répondre à un besoin ponctuel, il n'est pas conservé dans le fichier de zone et il n'est jamais mis en cache.

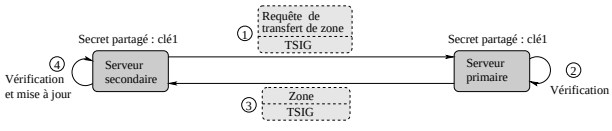

Fig. 18. Un exemple d'utilisation de TSIG.

La figure 18 montre un exemple d'utilisation de TSIG lors d'un transfert de zone. Un serveur secondaire envoie une requête de transfert de zone dans laquelle il aura inclus un enregistrement TSIG ①. À la réception de la requête, le serveur primaire vérifie le contenu de l'enregistrement TSIG grâce à la clé qu'il partage avec son serveur secondaire ②. Après cette vérification, le serveur primaire peut envoyer un message contenant le fichier de zone à jour protégé de la même manière

par un enregistrement TSIG ③. Le serveur secondaire vérifie l'enregistrement TSIG et met à jour sa copie du fichier de zone ④.

TSIG n'est pas un protocole d'échange de clé : il se base sur l'utilisation d'un secret partagé déjà en place. Pour établir un tel secret partagé, l'enregistrement TKEY [35] peut être utilisé.

2) SIG(0) : Un autre mécanisme existe pour signer les transactions, il s'agit de SIG(0) [33]. SIG(0) fonctionne exactement comme l'enregistrement RRSIG mais il contient une signature numérique de tout le message y compris l'en-tête. SIG(0) porte l'ancien nom des enregistrements permettant de conserver les signatures pour des raisons de compatibilité.

Tout comme TSIG, un enregistrement SIG(0) n'est jamais conservé en cache et possède une durée de vie relativement petite pour éviter les attaques par rejeu. La figure 19 décrit le fonctionnement de l'enregistrement SIG(0) sur une demande de transfert de zone. Cette signature est générée par un algorithme de chiffrement à clé publique utilisé avec la partie privée d'une clé de la zone. Une attention particulière doit être portée à la protection de la clé privée qui doit alors être gardée disponible en permanence sur le serveur de noms pour permettre la signature à la volée des messages.

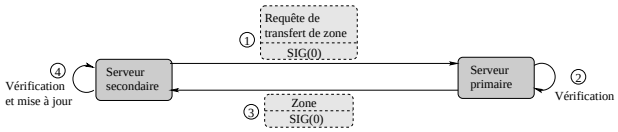

Fig. 19. Un exemple d'utilisation de SIG(0).

Nous pouvons remarquer que dans le processus d'échange des messages il n'y a pas de différence fondamentale entre TSIG et SIG(0). Ce qui différencie ces deux processus de signature est que TSIG utilise le chiffrement à clé secrète tandis que SIG(0) utilise le chiffrement à clé publique. Cela donne l'avantage à SIG(0) de ne pas avoir à utiliser en plus un protocole d'échange de clés. Néanmoins, SIG(0) possède aussi le désavantage du chiffrement asymétrique, à savoir un temps de calcul plus long et une plus grosse consommation de ressources.

Si TSIG ou SIG(0) sont utilisés, une personne malicieuse ne peut plus ajouter ou retirer des enregistrements dans le message sans en modifier la signature. Si la signature est modifiée, la vérification échouera et le message DNS sera rejeté.

Il faut aussi noter que TSIG et SIG(0) possèdent une très courte durée de vie pour éviter les attaques par rejeu : 300 secondes est la durée recommandée. Cela a une conséquence au niveau de leur utilisation. Il faut que les deux entités utilisant ces mécanismes pour protéger leurs transactions soient plus ou moins synchronisées, ce qui n'est pas forcément trivial sur l'Internet, le recours à un serveur NTP [36] étant souvent requis.

Bien que TSIG et SIG(0) protègent les messages DNS, ils ne sont malheureusement pas utilisables systématiquement. En effet, l'utilisation de la cryptographie pour signer chaque message puis vérifier les signatures prend du temps et impliquerait une forte dégradation des temps de réponse et des performances du DNS. Ces mécanismes de signature sont donc réservés à la protection des transferts de zone et des messages de mise à jour dynamique [37].

V. CONCLUSION

Dans cette article, nous avons présenté l'architecture et le fonctionnement de système de noms de domaine. La description des différentes entités intervenant au sein de ce protocole ainsi que les différentes interactions entre ces entités ont permis d'exposer les différentes vulnérabilités du protocole DNS. Aux vues de ces vulnérabilités il apparaît que les extensions de sécurité DNS sont une nécessité et devrait être bientôt largement déployées au niveau mondial afin de garantir l'intégrité et l'authenticité des données DNS.

DNSSEC est d'ores et déjà une technologie opérationnelle, mais quelques problèmes restent à résoudre. Notamment, la gestion automatiques des clés de zones [38]. En effet, à cause du grand nombre de zone DNS, la gestion manuelle de la cohérence des clés de zone est impossible. Des solutions sont à l'étude pour pour permettre de conserver la cohérence entre des clés de zones et des enregistrement DS lorsqu'un administrateur décide de renouveler une de ses clés. Des solutions voient aussi le jour pour garder la cohérence entre les clés de confiance configurées statiquement et les clés de zone lorsque ces dernières sont renouvelées. Il ne fait aucun doute que DNSSEC deviendra totalement automatisé dans sa gestion et pleinement déployé, très rapidement.

REFERENCES

[1] Mockapetris, P.: Domain Names - Concept and Facilities. RFC 1034 (1987)
[2] Mockapetris, P.: Domain Names - Implementation and Specification. RFC 1035 (1987)
[3] Albitz, P., Liu, C.: DNS and BIND. 4th edn. O'Reilly & Associates, Inc., Sebastopol, Californie (2002)
[4] Bellovin, S.M.: Using the Domain Name System for System Break-Ins. In: Proceedings of the 5th Usenix UNIX Security Symposium. (1995) 199–208
[5] Schuba, C.L.: Addressing Weaknesses in the Domain Name System. Master's thesis, Purdue University, Department of Computer Sciences (1993)
[6] Atkins, D., Austein, R.: Threat Analysis of the Domain Name System. RFC 3833 (2004)
[7] Guette, G., Cousin, B.: Les faiblesses du DNS. In: 2ème rencontre francophone sur la Sécurité et Architecture Réseaux (SAR). (2003) 235–244
[8] Eastlake, D.: Domain Name System Security Extensions. RFC 2535 (1999)
[9] Gundmundsson, O.: Delegation Signer Resource Record. RFC 3658 (2003)
[10] Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: DNS Security Introduction and Requirements. RFC 4033 (2005)
[11] Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: Resource Records for the DNS Security Extensions. RFC 4034 (2005)
[12] Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: Protocol Modifications for the DNS Security Extensions. RFC 4035 (2005)
[13] Gieben, R.: Chain of trust: The parent-child and keyholder-keysigner relations and their communication in dnssec. Master's thesis, University of Nijmegen (2001)
[14] Kolkman, O., Schlyter, J., Lewis, E.: Domain Name System KEY (DNSKEY) Resource Record (RR) Secure Entry Point (SEP) Flag. RFC 3757 (2004)
[15] Sit, E.: A Study of Caching in the Internet Domain Name System. Master's Thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Sciences (2000)

[16] Jung, J., Sit, E., Balakrishan, H., Morris, R.: DNS Performance and the Effectiveness of Caching. In: Proceedings of the ACM SIGCOMM Internet Measurement Workshop '01. (2001) 153–167

[17] Cohen, E., Kaplan, H.: Proactive Caching of DNS Records: Addressing a Performance Bottleneck. In: Symposium on Applications and the Internet. (2001) 85–94

[18] Lioy, A., Maino, F., Marian, M., Mazzocchi, D.: DNS Security. In: Terena Networking Conference. (2000)

[19] Brownlee, N., Claffy, K., Nemeth, E.: DNS Measurements at a Root Server. In: IEEE GLOBECOM 2001, San Antonio, Texas (2001)

[20] Naraine, R.: Massive DDoS Attack Hit DNS Root Servers (2002) http://www.internetnews.com/dev-news/article.php/1486981.

[21] Thurrott, P.: Microsoft Suffers Another DoS Attack (2001) http://www.windowsitpro.com/Article/ArticleID/19770/19770.html?Ad=1.

[22] Vixie, P., Thomson, S., Rekhter, Y., Bound, J.: Dynamic Updates in the Domain Name System (DNS UPDATE). RFC 2136 (1997)

[23] Kolkman, O., Gieben, R.: DNSSEC operational practices. Draft IETF, work in progress (2006)

[24] Massey, D., Rose, S.: Limiting the Scope of the KEY Resource Record (RR). RFC 3445 (2002)

[25] IDsA: Infrastructure DNSSEC et ses Applications. http://www.idsa.prd.fr (2004)

[26] Schlyter, J.: DNS Security (DNSSEC) NextSECure (NSEC) RDATA Format. RFC 3845 (2004)

[27] Laurie, B., Sisson, G., Arends, R.: DNSSEC Hash Authenticated Denial of Existence. Draft IETF, work in progress (2005)

[28] Pfleger, J.: DNSSEC Resolver Algorithm. Master's Thesis, Fachhochschulen, Autriche (2003)

[29] Pels, M.: DNSSEC Validator. Master's Thesis, Hogenschool van Amsterdam (2004)

[30] Wellington, B., Gundmundsson, O.: Redefinition of DNS AD bit. RFC 3655 (2003)

[31] Vixie, P.: Extension Mechanisms for DNS (EDNS0). RFC 2671 (1999)

[32] Vixie, P., Gudmundsson, O., Eastlake, D., Wellington, B.: Secret Key Transaction Authentication for DNS (TSIG). RFC 2845 (2000)

[33] Eastlake, D.: DNS Request and Transaction Signatures (SIG(0)s). RFC 2931 (2000)

[34] Krawczyk, H., Bellare, M., Canetti, R.: HMAC: Keyed-Hashing for Message Authentication. RFC 2104 (1997)

[35] Eastlake, D.: Secret Key Establishment for DNS (TKEY RR). RFC 2930 (2000)

[36] Mills, D.: Network Time Protocol (NTP). RFC 958 (1985)

[37] Wellington, B.: Secure Domain Name System (DNS) Dynamic Update. RFC 3007 (2000)

[38] Guette, G.: Gestion de clés dans les extensions de sécurité DNS. Thèse de doctorat, Université de Rennes 1 (2005)