Evaluation du coût de la sécurisation du système DNS

Daniel Migault[1] and Bogdan Marinoiu[2]

[1] France Telecom R&D
Laboratoire Sécurité des Services et Réseaux
38-40 Rue du Général Leclerc
92794 Issy-les-Moulineaux Cedex 9 - France
daniel.migault@francetelecom.com
[2] Polytechnique
bogdan.marinoiu@polytechnique.org

Résumé L'objet de cet article est de mesurer les coûts liés à l'utilisation des extensions de sécurité du DNS. Cet article présente les résultats d'une étude sur les performances des diverses extensions de sécurité du protocole DNS (DNSSEC et TSIG), ainsi que d'autres protocoles de sécurité comme IPsec. Les tests ont été faits sur une plateforme, de manière à permettre une comparaison entre ces diverses extensions. Le coût de ces diverses extensions de sécurité ayant été mis en évidence, il est alors possible d'implémenter une base DNS sécurisée en choisissant les mécanismes de sécurité de manière adéquate en fonction des performances que le système doit atteindre, l'application spécifique à laquelle doit répondre l'architecture DNS, son environnement réseau... Inversement, on pourra également décider, connaissant le coût généré par les extensions de sécurité, de ne pas les implémenter dans un premier temps, et modifier l'architecture progressivement de manière à pouvoir implémenter, par la suite, les extensions choisies.

1 Introduction

Internet repose en grande partie sur le système DNS, qui permet d'établir le lien entre un nom de domaine et une adresse IP. Ce système de nommage a été construit à l'heure où la sécurité ne constituait pas une préoccupation importante dans l'élaboration des protocoles. Par la suite des options et des extensions de sécurité (DNSSEC rfc2535 [**?**], rfc4033 [**?**], rfc4034 [**?**], rfc4035 [**?**], TSIG rfc2845 [**?**], SIG(0) rfc2931 [**?**]) ont été normalisées. Toutefois, le déploiement de ces options n'est pas encore très répandu à l'échelle de l'Internet, et le système DNS à l'heure actuelle ne peut être considéré comme sécurisé.
Les attaques de plus en plus nombreuses perpétrées à l'encontre du système de nommage montrent que, du point de vue sécurité, le système DNS constitue en lui même une faiblesse de l'Internet. Le déploiement de DNSSEC se fait de plus en plus pressant. Toutefois, ce protocole normalisé en mars 2005 implique des changements radicaux sur la structure et l'architecture des serveurs DNS actuels. En effet, le protocole DNSSEC utilise des mécanismes plus lourds qui modifient

considérablement le protocole DNS.
Le système DNS est utilisé à l'heure actuelle principalement pour assurer le service de nommage sur l'Internet. Toutefois, l'émergence de nouveaux protocoles comme ENUM, par exemple, offre de nouvelles perspectives de services et donc une utilisation nouvelle des bases de données DNS, avec de nouvelles contraintes de sécurité, comme la confidentialité, l'authentification des clients...
L'objet de cet article est donc de mesurer les coûts liés à l'utilisation des extensions de sécurité du DNS. Cet article présente les résultats d'une étude sur les performances des diverses extensions de sécurité du protocole DNS (DNSSEC et TSIG), ainsi que d'autres protocoles de sécurité comme IPsec (rfc2401 [**?**], [**?**]). SIG(0) ne sera pas testé ici car ce protocole utilise de la cryptographie asymétrique, et est essentiellement utiliser afin d'authentifier les requêtes émises par une personne spécifique, comme un administrateur. Les tests ont été faits sur une plateforme, de manière à permettre une comparaison entre ces diverses extensions. Le coût de ces diverses extensions de sécurité ayant été mis en évidence, il est alors possible d'implémenter une base DNS sécurisée en choisissant les mécanismes de sécurité de manière adéquate en fonction des performances que le système doit atteindre, l'application spécifique à laquelle doit répondre l'architecture DNS, son environnement réseau... Inversement, on pourra également décider, connaissant le coût généré par les extensions de sécurité, de ne pas les implémenter dans un premier temps, et modifier l'architecture progressivement de manière à pouvoir implémenter, par la suite, les extensions choisies.

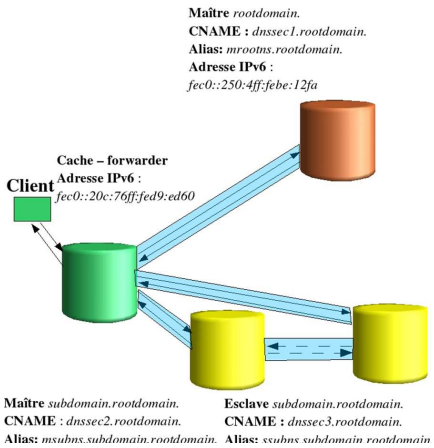

Fig. 1. Plateforme de test DNS/DNSSEC avec des tunnels IPsec

2 Description de la Plateforme DNS

L'ensemble des tests DNSSEC ont été effectués sur une architecture (voir Illustration 1 et Illustration 2) à deux zones :

- Une zone mère correspondant au domaine rootdomain., et
- Une zone fille pour subdomain.rootdomain.

La plateforme DNS servant cette architecture comprend trois serveurs :

- Un pour le domaine rootdomain., et
- Deux pour le sous-domaine subdomain.rootdomain : un serveur maître et un esclave qui se synchronise avec le maître.

L'ensemble de la plateforme est relié au réseau IPv6 ([?]) de France Télécom R&D. Pour effectuer les tests, un quatrième serveur cache-forwarder qui sert de relais pour les clients a été introduit.
Les serveurs sont des Pentium II et III à 500 Mhz avec 128 MB de mémoire, utilisant le système d'exploitation Linux Ubuntu (kernel 2.6).

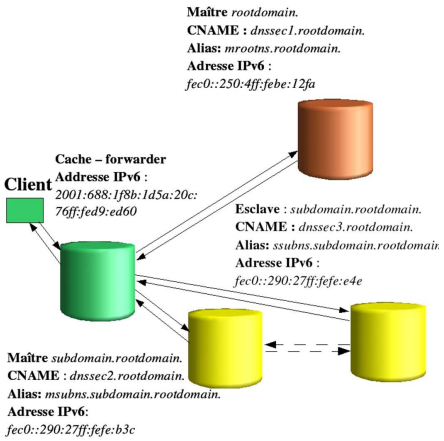

Fig. 2. Architecture de test DNS/DNSSEC

3 Objectifs et stratégie

Cette partie vise à donner une brève description des différentes extensions et protocoles qui peuvent concerner la sécurité du système DNS. A la suite de ce tour d'horizon, nous déterminerons la stratégie de test à suivre.

3.1 Description des mécanismes de sécurité DNS

DNSSEC est une extension sécurisée de DNS standardisée par l'IETF (par la rfc4033 [?], la rfc4034 [?] et la rfc4035 [?]). Elle permet de vérifier l'intégrité des

données des échanges du protocole DNS. Elle permet également d'authentifier la source de la donnée. Pour cela, elle utilise un système de clé qui signe les données et qui permet à partir d'un point sécurisé de parcourir une chaîne de confiance (les serveurs DNS) et de sécuriser le lien entre les données envoyées entre les différents serveurs de l'architecture du DNS.
TSIG et SIG(0) (DNS Request and Transaction Signatures) sont des extensions DNS qui permettent d'assurer des services d'authentification.
TSIG (rfc2845 [**?**]) signe les données de manière symétrique. Les clients et le serveur s'authentifient donc grâce à une clé partagée. Ce système d'authentification est donc envisageable pour des architectures où le nombre de clients est réduit. En effet, soit ces derniers doivent partager une clé symétrique, soit le serveur doit posséder une clé symétrique par client différent. Un mécanisme qui permet l'échange de clefs symétriques pour TSIG est TKEY (rfc2930 [**?**]).
SIG(0) (rfc2931 [**?**]) permet de chiffrer la transaction avec une clef privée (cryptographie asymétrique). Sa contrepartie publique devrait être stockée dans la zone. A la différence de DNSSEC, c'est plutôt l'échange qui est authentifié/signé que la donnée DNS elle même.
IPsec (rfc2401 [**?**]) (Internet Protocol Security) permet de gérer la confidentialité et l'authentification des datagrammes IP. L'authentification ne porte pas sur les données DNS spécifiquement, mais bien plus sur une « connexion » entre deux adresses IP. La vérification porte donc sur la provenance immédiate de la donnée. Il est possible d'authentifier le fournisseur de la donnée, par exemple un serveur cache, mais pas toujours la source initiale sauf s'il y a un lien IPsec vers le serveur de nom primaire. IPsec permet le chiffrement symétrique entre le client et le serveur, ce qui, par rapport à SIG(0) permet une authentification plus facilement gérable. De plus, IPsec est muni d'un protocole de négociation de clefs et d'Associations de Sécurité (IKE, Internet Key Exchange).

3.2 Analyse de ces mécanismes

Afin de distinguer ces mécanismes de sécurité, il est nécessaire de considérer les aspects suivants :

Quel est le mode de chiffrement utilisé ? Les opérations de chiffrement peuvent mettre en jeu de la crypto symétrique ou asymétrique. La crypto symétrique nécessite que les deux entités échangeant l'information possèdent cette clé. Le gros avantage de ce mode de chiffrement est qu'il est très peu coûteux, pour un niveau de sécurité donné, en termes de ressources nécessaires par rapport à l'utilisation la cryptographie asymétrique. Le premier inconvénient de l'utilisation de la cryptographie symétrique est qu'il faut trouver un moyen sécurisé pour communiquer la clé partagée par les deux entités. Le second inconvénient est que l'utilisation d'une même clé par les entités partageant des informations ne permet pas d'authentifier l'entité émettant l'information. En effet toutes les entités utilisent le même mécanisme de chiffrement et la même clé. Il est donc

impossible de connaître la provenance de l'information.

Quelle sécurité est implémentée ? La sécurité reste un terme générique, et peut être implémentée selon différents mécanismes. Les mécanismes en question sont :

La confidentialité : cette caractéristique permet de rendre toute donnée chiffrée inexploitable par une quelconque entité n'ayant pas la clé nécessaire au déchiffrement. Les opérations de chiffrement peuvent impliquer aussi bien de la cryptographie symétrique que de la cryptographie asymétrique.

L'intégrité : cette caractéristique permet de s'assurer que la donnée n'a pas été changée entre le moment où elle a été émise par l'entité source, et reçue par l'entité destinataire. On utilise souvent, pour cela un mécanisme de hachage, qui permet à partir d'une donnée quelque soit sa taille initiale, de fournir un certain nombre d'octets que l'on appelle hash. Une propriété essentielle du hash est que deux donnée initiale doivent avoir, en termes de probabilité, des hashs différents !

L'authentification : L'authentification permet au destinataire de s'assurer de la provenance de l'information. L'authentification nécessite dans la plus part des cas l'utilisation de la cryptographie asymétrique. En effet, une donnée chiffrée a par la clé privée d'une entité, ne sera déchiffrée que par les entités possédant la clé publique associée. L'authentification d'une donné ce fait généralement par l'intermédiaire de signatures, qui correspondent au chiffrement par une clé privé d'un résumé (ou hash) de la donnée.

Quelle est la donnée chiffrée ? Avec une même méthode de chiffrement, il est possible de chiffrer des données différentes. Chiffrer les paquets IP, dans le cadre d'IPsec permet d'intégrer la sécurité au niveau transport, i.e. sur la communication entre deux entités, alors que chiffrer les données DNS, i.e. au niveau application permet de s'abstraire de l'implémentation réseau. La données est signée par une autorité, et quelque soit la manière dont une entité obtient cette donnée signée, i.e. directement de l'autorité, ou non, cette entité est capable de vérifier la provenance de cette donnée. Ainsi la donnée chiffrée au niveau application permet l'utilisation de nœuds intermédiaires (des serveurs que l'on appelle ici serveurs caches), car l'authentification ne se fait pas au niveau de serveur.

Outre la position de la donnée dans le modèle ISO/OSI [**?**] la donnée peut également différer selon sa nature. En effet, traditionnellement, une communication entre deux entités est dites chiffrée (resp. signée) lorsque la sortie de l'entité source est chiffrée (resp. signée) et que le flux est déchiffré (resp. la signature vérifiée) par l'entité destinataire. Dans ce cas, le flux est chiffré, et chaque communication nécessite une opération de chiffrement. On peut également considérer, qu'on donnée soit chiffrée (resp. signée) une fois pour toutes, et donc une opération

de chiffrement (resp. de signature) ne soit pas nécessaire à chaque nouvelle transaction. Cette vision permet d'effectuer un pré-calcul, et donc d'éviter l'opération de chiffrement (resp de signature) lors des charges.

Mécanisme	Crypto	Donnée chiffrée	Sécurité			Couche OSI
	Sym / Asym	dynamique / statique	Conf	Int.	Ayth.	
DNSSEC	Asym	Statique	Non	Oui	Oui	Appli
TSIG	Sym	Dynamique	Non	Oui	Non	Appli
IPsec	Sym	Dynamique	Oui	Oui	Oui	Transport

Tab. 1. Tableau comparé des mécanismes

Tableau comparé des mécanismes Ainsi, il apparaît clairement que par rapport à TSIG, DNSSEC utilise de la cryptographie symétrique, plus coûteuse que la cryptographie symétrique. En revanche DNSSEC pré-calcule des signatures données DNS, alors que TSIG calcule les signatures des datagrammes transmis. Le fait d'utiliser la cryptographie asymétrique permet au protocole DNSSEC au destinataire d'authentifier la source émettrice de l'information. Cette source est identifiée par son couple de clé privée / clé publique, et non pas par l'adresse IP d'un serveur. En effet, la donnée concernée par les opérations de sécurité est la donnée DNS. Tout comme DNSSEC, TSIG n'implémentent pas la fonction de confidentialité. En effet, les données DNS sont considérées comme publiques et n'ont pas conséquent pas à être chiffrées. TSIG doit être associé à un mécanisme de distribution de clés. Un déploiement du mécanisme TSIG sur un grand nombre de client ne semble par conséquent pas envisageable.

IPsec, par rapport à DNSSEC permet d'implémenter la confidentialité. Par contre IPsec associe les opérations de sécurité à des adresses IP. Ce mécanisme ne permet l'utilisation de tiers, comme les serveurs caches, distribuant l'information. En effet, les données DNSSEC, i.e. la donnée et sa signature associée permettent au client d'en vérifier la provenance de l'information indépendamment de l'entité dont elle est envoyée.

IPsec, comme TSIG utilise le chiffrement symétrique rapide, et moins coûteux que le chiffrement asymétrique utilisé dans DNSSEC. TSIG n'implémente que l'intégrité et ne permet ni l'authentification, ni la confidentialité. Par contre IPsec bénéficie de mécanismes de négociation de clés que sont IKEv1 et IKEv2. Ces coûts de négociation lors de l'établissement de liens IPsec, ne sont d'ailleurs

pas pris en compte dans les tests.

Enfin notons également que DNSSEC est un protocole qui vise à sécuriser l'ensemble du système DNS, alors que TSIG et IPsec sécurise davantage des transactions. Ainsi DNSSEC est muni de mécanismes permettant, grâce aux champs spécifiques NSEC, et à un classement lexicographique des données DNS (i.e. des noms de domaine), de fournir au client une preuve de non existence d'une donnée faisant l'objet d'une requête. De plus, DNSSEC est muni de mécanismes de sécurité permettant d'établir une chaîne de confiance entre les différents serveurs composants l'architecture de nommage, un peu comme une PKI.

3.3 Les objectifs de l'étude

L'objectif des tests est de comparer les performances de la plateforme dans plusieurs configurations possibles. Certaines des extensions présentées ci dessous peuvent paraître redondantes dans certains cas au niveau de leur fonctionnalité, notamment lorsque l'on confronte des mécanismes de sécurité s'attachant à vérifier l'intégrité des enregistrements, ou à authentifier soit au niveau DNS, soit à des niveaux plus bas. L'objectif des tests est donc de permettre à un administrateur de connaître le coût lié à ces différentes configurations sécurisées, et d'effectuer un choix en fonction de ses besoins et de ce coût.

Les principales configurations étudiées ont été :

DNS : Cette configuration est la référence à laquelle nous allons nous référer de manière à évaluer le coût de la sécurité.

DNSSEC : Cette extension de sécurité, permet de garantir l'intégrité des données hébergées au sein du serveur DNS, et de garantir que la source des enregistrements est une entité de confiance. Cette option est plus à considérer dans le cadre d'une relation client - serveur, car le serveur fournit au client les informations nécessaires pour valider l'information. Coté serveur, le client reste anonyme, et n'est pas authentifié.

DNS + IPsec : Cette configuration est davantage utilisé dans le cadre des relations serveur - serveur ou client - serveur dans le cas où le nombre de clients est restreint. En effet, l'établissement d'un tunnel IPsec nécessite une configuration préalable et n'est pas envisageable dans le cas d'un nombre élevé de clients. Il s'agit donc ici d'une configuration mettant en oeuvre des serveurs DNS avec la sécurisation des liens serveur maître - serveur esclave ou clients - serveurs DNS. Dans cette configuration, on profitera des propriétés d'IPsec qui permettent non seulement d'authentifier les paquets par signature électronique, mais aussi de chiffrer les données assurant ainsi leur confidentialité.

DNSSEC + IPsec : Cette configuration utilise IPsec afin de sécuriser les transactions entre les serveurs, et DNSSEC afin de sécuriser les transactions entre les clients et le serveur DNS.

TSIG : Cette configuration n'est utilisée que pour des liens serveurs - serveurs ou client - serveur dans le cadre d'un nombre restreint de clients. En effet, TSIG utilise un chiffrement symétrique et nécessite alors une configuration entre les deux acteurs, à la différence de IPsec qui peut utiliser IKE pour la négociation de clefs. Toujours a la différence d'IPsec, TSIG n'implémente pas le chiffrement de données et ne permet donc pas d'assurer la confidentialité des données lors de la transaction.

3.4 Stratégie de tests

Pour chacune des configurations mentionnées, nous allons essayer d'évaluer l'impact de la sécurité, en terme de coût sur les points suivants :

1. Les temps de réponse à des requêtes.
2. L'occupation CPU par le processus serveur DNS.
3. Les temps de mise à jour.

Les tests sont effectués sur deux environnements distincts :
- Un serveur particulier
- La plateforme

On peut mener les tests sur un serveur particulier et mesurer ainsi l'impact dû à la sécurité sur un serveur. On parlera alors de *tests serveur*. Toutefois, l'architecture DNS est constituée de plusieurs serveurs, et l'impact de la sécurité doit être mesuré par rapport à l'ensemble de ces serveurs. On parlera alors de *tests de plateforme*.

Par ailleurs les tests peuvent avoir trois natures distinctes :

Tests unitaires : il s'agit de mesurer et caractériser le comportement de la plateforme ou d'un serveur dans le cas d'une requête unitaire. Pour cela les protocoles de mesures et de probabilité nous amènent à effectuer une succession de requêtes.

Tests en charge : Ces tests visent à observer le comportement d'un serveur ou de la plateforme dans le cas d'une utilisation réelle, i.e. par plusieurs clients indépendants.

Tests de mise à jour de serveurs utilisant `nsupdate`

Les tests se feront en trois parties. Nous commencerons par les tests unitaires, puis nous continuerons par les tests en charge. Enfin, nous terminerons par les tests de mise à jour.

Pour les tests portant sur l'évaluation des temps de réponse et l'occupation mémoire et CPU des serveurs, deux clients DNS ont été utilisé : le client classique `dig` qui fait partie de la distribution `BIND 9.3.1` ([**?**]) et un client Java

développé notamment pour les tests. A la différence du client `dig`, qui se bloque dans l'attente d'une réponse après avoir envoyé la requête, le client Java permet d'envoyer des requêtes de manière asynchrone (le programme a deux threads d'exécution : il envoie les requête sur le premier et reçoit et analyse les réponses sur le deuxième). De ce fait, il est adapté aux tests à charge variable. Pour les tests portant sur les mises à jour, le programme `nsupdate` de `BIND 9.3.1` a été utilisé pour tester le temps de mise à jour. Pour générer de manière automatique les objets de requêtes, on a crée des scripts `bash`.

4 Tests unitaires

4.1 Description des tests unitaires

Le premier test consiste à faire répondre la plateforme à des requêtes DNS lancées par un client `dig`. Ce client générera un nombre de requêtes. La requête $N+1$ est envoyée après la réception de la réponse à la requête N. On se contente de mesurer le temps de réponse à toutes les questions et d'évaluer les ressources utilisées. Les requêtes sont les mêmes dans toutes les configurations.

Un serveur DNS est un processus (appelé named) sur une machine. Ce processus emploie des ressources de la machine : une partie du temps CPU et une partie de la mémoire vive.

4.2 Coût sécurité sur le temps de réponse

Le Tableau 2 présente dans sa deuxième ligne les temps de réponse en millisecondes. Des différences entre les performances obtenues dans des diverses configurations sont tout de suite visibles. Pour l'instant, seul le temps de résolution de la tâche est pris en compte. Des dégradations importantes apparaissent déjà dans les configurations mettant en œuvre des mécanismes de sécurité. Le coût du à la sécurité est mis en évidence dans la troisième ligne du tableau.

Configuration	DNS	$(DNS + IPsec)$	$DNSSEC$	(DNSSEC+IPsec)
Temps de réponse (ms)	192	207	258	299
Coût de la sécurité sur le temps de réponse par rapport au DNS classique (%)	0%	+8%	+34%	+55%

Tab. 2. Temps de réponse selon les extensions de sécurité

Le temps de réponse a été jugé comme paramètre pertinent pour faire des comparaisons entre les différentes configurations analysées. Les résultats obtenus montre clairement la classification suivante en terme répercussion sur le

temps de réponse : DNS < (DNS+IPsec) < DNSSEC < (DNSSEC + IPsec). Le coût inférieur généré par l'utilisation d'IPsec par rapport à DNSSEC est dû au fait que les opérations de chiffrement symétrique sont nettement moins coûteuses que les opérations de chiffrement asymétriques. Bien que les signatures des champs soient pré-calculée par le serveur, et donc une requêtes, ne demande pas d'opération de chiffrement de la part du serveur, le client doit vérifier la signature. Cette opération de vérification comprend, un calcul de hash, et une comparaison de ce dernier avec la valeur fournie par le serveur DNSSEC. Cette valeur est chiffrée à l'aide de la clé privée du serveur DNSSEC, et placée dans le champ de type SIG. Le client doit donc d'une part calculer un hash, puis déchiffrer avec la clé privée celui contenu dans le champ SIG, et comparer ces deux valeurs. Ces deux opérations nécessitent, pour une même machine client, plus de temps de calcul que les opérations de chiffrement symétrique d'IPsec.

Le coût du à l'utilisation simultanée de DNSSEC et d'IPsec est supérieur à la somme des coûts générés par DNSSEC et IPsec séparément. Ceci peut être en partie expliqué par le fait que DNSSEC nécessite l'envoie de plus de données que DNS, et donc une opération de chiffrement plus lourde que dans le cas de l'utilisation de DNS est mise en œuvre.

4.3 Coût sécurité sur le temps CPU

Le Tableau 3 présente le coût (en temps de calcul) engendré par les différentes configurations. Ce coût est mesuré par la charge CPU. Il apparaît évident que le relais (le cache-forwarder) est plus chargé que les autres entités de l'architecture lors de l'utilisation d'un client unique : autour de 8% de la charge CPU pour le relais contre à peine 2,9% charge CPU pour le serveur de rootdomain, les machines ayant une configuration similaire. Le cache-forwarder est un serveur fonctionnant en mode récursif : toutes les requêtes passent par lui et il se charge de garder des contextes pour toutes les requêtes. Pour chacune des requêtes reçues, il peut envoyer plusieurs requêtes envers les serveurs de la plateforme dans le but d'obtenir la réponse finale car les serveurs de la plateforme fonctionnent en mode itératif : ils donnent la réponse la plus appropriée qu'ils possèdent. Nous représenterons entre parenthèses une quantification du coût généré par la sécurité.

Les charges CPU des serveurs sont assez faibles. En effet, le client dig envoie les requêtes les unes après les autres, dès qu'il a obtenu la réponse de la précédente requête. Ce client est donc incapable de charger de façon suffisante les serveurs et met ainsi en évidence la haute disponibilité des serveurs DNS, par rapport aux coûts réseaux.

Ces résultats montrent que le coût de la sécurité se répercute différemment selon le type de serveur.
Ainsi pour le serveur DNS racine, interrogé lors de chaque interrogation, le coût de la sécurité peut être considéré, par ordre de croissance : DNS << (DNS + IPsec) < DNSSEC <<< (DNSSEC + IPsec). La différence de coût entre

Config. / Serveur testé	DNS	(DNS + IPsec)	DNSSEC	(DNSSEC + IPsec)
dnssec1 (serveur rootdomain)	0.82% [+0 %]	1.21% [+47%]	1.25% [+52%]	2.89% [+252%]
dnssec2 (serveur maître subdomain)	0.20% [+0%]	0.36% [+80%]	0.30% [+50%]	0.90% [+350%]
dnssec3 (serveur esclave subdomain)	0.20% [+0%]	0.35% [+75%]	0.30% [+50%]	0.73% [+265%]
Cache - forwarder	7.20% [+0%]	7.40% [+2%]	8.80% [+22%]	8.80% [+22%]

Tab. 3. Charge CPU en fonction des extensions de sécurité

DNSSEC et la combinaison (DNS + IPsec) est relativement faible. Il serait alors raisonnable de penser que le coût de chiffrement symétrique dans (DNS et IPsec) est compensé par la les opérations propres à DNSSEC. Les opérations propres à DNSSEC ne comportent pas ici de calculs induits par le chiffrement. Les fichiers de zone DNSSEC sont beaucoup plus volumineux que les fichiers de zone DNSSEC (environ 7 fois plus volumineux). En effet, les fichiers de zone DNSSEC contiennent, en plus des champs du DNS traditionnel, des champs de type propre à DNSSEC (NSEC, SIG...). Lors que le serveur reçoit une requête, il doit procéder à la recherche d'informations Le serveur doit alors procéder à la recherche d'informations spécifiques au sein d'un fichier plus volumineux d'une part. De plus, à une information DNS sont associées plusieurs informations DNSSEC, elle-même relativement volumineuses. On peut estimer qu'au volume d'une information DNS de type adresse (A), le volume DNSSEC de la même information est environ 14 fois plus important. En effet, à une donnée DNS, il faut dans le cadre de DNSSEC lui associer une signature ainsi qu'un champ de type NSEC également signé. L'ensemble des données DNSSEC que le serveur DNSSEC doit traiter et envoyer est alors beaucoup plus volumineux que dans le cadre de DNS. Ainsi, le coût CPU d'IPsec correspond au traitement de deux champs DNS auquel on associe les champs DNSSEC.
Concernant les serveurs de la zone subdomain maître et esclave, le classement est DNS < DNSSEC < (DNS + IPsec) << (DNSSEC + IPsec).
La différence entre DNSSEC et (DNS + IPsec) n'est pas, au regard des valeurs des %CPU flagrante. En effet, dans le cadre d'une association maître - esclave, il faut tenir compte des messages échangés entre les maîtres et les esclaves. Ces messages ne sont nullement impactés par DNSSEC. Par contre un lien IPsec a été établi entre les serveurs « maître » et « esclave », et ces communications sont chiffrés. Ces messages, ne sont certes pas volumineux, mais dans le cas de tests unitaires, leur impact, avec les valeurs du champ paramétrant ces échanges (champ SOA), n'est pas négligeables. Ainsi IPsec est plus consommateur en ressources CPU car nous sommes dans le cas de tests unitaires et que les dialogues entre les clients et les esclaves sont également chiffrés par IPsec alors qu'ils ne

sont nullement impacté par DNSSEC.
Analysons la différence entre le comportement des serveurs de la zone rootdoamain et ceux de la zone subdomain. Dans le cadre de la configuration de la maquette, les serveurs DNS de la zone subdomain traitent deux fois moins de requêtes que celui de rootdomain. En effet, le trafic est réparti sur les serveurs « maîtres » et les serveurs « esclaves ». Toutefois, la nature des trafics générés n'est pas identique. Les paquets réponses du serveur rootdomain comprennent l'association de subdomain à un serveur DNS (un champ de type NS), et l'association de ce serveur DNS à une adresse IP (champ de type A). Il faut dans le cadre de DNSSEC tenir compte des champs DNSSEC associé, i.e. NSEC et SIG pour chacun d'eux. Dans le cas des serveurs « maître » et « esclave » traitant la zone subdomain, chacun renvoi dans sa réponse l'association entre le nom de domaine et l'adresse IP (type A). Dans le cas de DNSSEC, il faut lui associer les champs de type NSEC et SIG. Ainsi, les serveurs de la zone rootdomain ont des réponses deux fois plus importantes que les serveurs de la zone subdomain. Les serveurs de la zone subdomain traitent également deux fois moins de requêtes que ceux de la zone rootdomain. Ceci est confirmer par un rapport d'environ $\frac{1}{4}$ entre le serveur de la zone rootdomain et ceux de la zone subdomain.

Concernant le serveur cache-forwarder, il est difficile de mener une analyse comparative avec les autres serveurs de la plateforme, car la machine est différente. Par contre l'analyse entre les différents protocoles de sécurité peut être menée. Le classement en termes de consomation CPU est alors DNS < IPsec << DNSSEC < (DNSSEC + IPsec). Les coûts induits par IPsec sur le cache forwarder sont négligeables en comparaison de ceux induits par IPsec sur les autres serveurs (en pourcentages). La principale différentiation de ce serveur est qu'il est soumis à une charge et des opérations de vérification de signature qui permettent de mettre en évidence les coûts relatifs dus à la sécurité, en présence de trafic ou de charge réseau.
Ainsi, ces tests mettent en évidence que les coûts générés par la sécurité doivent également être évalués en charge, que DNSSEC permet de recentrer la sécurité sur les relations client serveur, alors qu'IPsec tient compte de l'ensemble du trafic (interrogation client, gestion des serveurs...). Toutefois il est raisonnable de considérer DNS < (DNS + IPsec) < DNSSEC << (DNSSEC + IPsec).

4.4 Coût sécurité sur la ressource mémoire

Le Tableau 4 présente le pourcentage mémoire utilisé pendant les tests par les trois serveurs et par le cache-forwarder.

La charge des serveurs est proportionnelle au nombre d'enregistrements de la zone (un serveur `BIND` charge au démarrage toute la zone en mémoire, sous réserve d'une capacité de mémoire suffisante sinon il crache). Des tests antérieurs ont montré que la taille de zone n'a aucun impact sur le temps de résolution d'une requête. Dans l'exemple considéré, les zones ont des dimensions réduites

Config. / Serveur testé	DNS	(DNS + IPsec)	DNSSEC	(DNSSEC + IPsec)
dnssec1 (serveur rootdomain)	1.60%	1.60%	1.60%	1.60%
dnssec2 (serveur maître)	1.70%	1.70%	1.70%	1.80%
dnssec3 (serveur esclave)	1.70%	1.70%	1.80%	1.80%
cache - forwarder	0.40%	0.40%	0.80%	0.90%

Tab. 4. Utilisation de la mémoire en fonction des extensions de sécurité

(peu d'enregistrements) ce qui explique les faibles pourcentages de mémoire utilisée sur les serveurs de nom.

L'occupation de la mémoire d'un serveur cache-forwarder devrait être assez importante dans la situation où les temps de vie (TTL) des enregistrements DNS sont non nuls. En raison des tests répétitifs, le TTL des enregistrements que nous utilisons est fixé à 0 (ils sont jetés de la mémoire du cache-forwarder juste après la fin de la résolution d'une requête). Ceci explique le faible pourcentage mémoire utilisée. Toutefois, la quantité de mémoire nécessaire dans le cas d'une résolution DNSSEC est deux fois plus élevée, en raison des enregistrements supplémentaires (SIG, NSEC) introduits par DNSSEC. Cette différence n'est visible que sur le serveur cache-forwarder car il se comporte comme un client, effectuant des opérations de vérification, alors qu'un serveur de nom primaire utilise peu de ressources mémoires supplémentaires dues à des opérations de calcul.

4.5 Conclusion

Ces premiers tests permettent de mettre en évidence le coût de la sécurité sur des requêtes unitaires en fonction des configurations. En terme de ressources CPU et de temps de réponse, le fait qu'IPsec utilise la cryptographie symétrique rend ce protocole moins gourmand en ressources que l'utilisation de chiffrement asymétrique de DNSSEC. Cette différence est relativement importante en terme de temps de réponse unitaire mais la différence ne semble plus significative lorsque la charge augmente un peu.
En terme de ressource mémoire, les tests unitaires font apparaître que DNSSEC, nécessite plus de ressources, à la fois en raison des calculs générés par cette solution, et par les données nécessaires et supplémentaires.
Notons également que IPsec est ici testé uniquement dans le cas où les tunnels ont déjà été configurés. La négociation n'est pas prise en compte.
Dans ce premier test, le client `dig` fourni par la distribution `BIND 9.3.1` a montré ses limites : il s'avère intéressant de l'utiliser dans des tests unitaires, mais il ne permet pas de faire varier le nombre de requêtes envoyées à la plateforme afin

d'étudier la capacité maximale de traitement des serveurs.

Pour cette raison, de nouveaux tests ont été faits avec un client DNS développé en Java, à la place de `dig`.

5 Tests en charge

A partir de ce test, le client utilisé n'est plus dig, mais le client développé pour les besoins des tests. Il nous permet de faire varier le nombre de requêtes envoyées par seconde vers un serveur et d'estimer le nombre de réponses correctes reçues. Il simule de cette manière l'existence de plusieurs clients qui interagissent de façon asynchrone avec les serveurs DNS, de manière à étudier leur comportement dans un environnement réseau. Les paramètres regardés attentivement seront le taux de réponses correctes et la charge CPU dans le cas d'une surcharge. Par la suite des tests sur un serveur et sur l'ensemble de la plateforme seront faits.

5.1 Coût CPU et taux de réponses correctes en mode mono - serveur

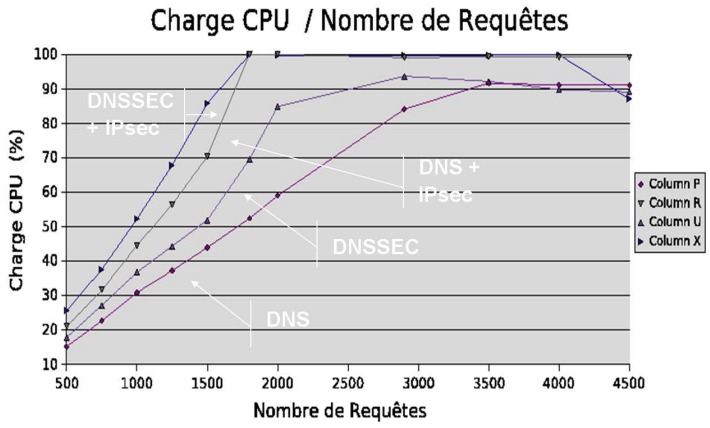

Fig. 3. Charge CPU fonction de nombre de requêtes, en fonction des différentes configurations DNS, (DNS + IPsec), DNSSEC, (DNSSEC + IPsec)

L'objectif de ce test est de mettre en évidence la capacité maximale de traitement d'un serveur de nom, et de quantifier l'impact des configurations de sécurité sur les serveurs DNS. Chaque configuration met en évidence un point de rupture correspondant à un nombre de requêtes à partir duquel le serveur ne

sera plus capable d'effectuer la moindre opération. On parlera alors de *point de rupture machine*. L'Illustration 3 met en évidence ce point de rupture en traçant la charge CPU en fonction du nombre de requêtes envoyées.
Le graphe met alors deux valeurs de point de rupture machine selon les configurations de sécurité. Les configurations implémentant (DNSSEC + IPsec) et (DNS + IPsec) montent jusqu'à 100% CPU alors que les configurations DNSSEC et DNS n'atteignent que 90% CPU. Ainsi, la mise en place d'IPsec peut mettre la machine en surcharge, voir la rendre indisponible, non seulement pour le service DNS, mais également pour les autres fonctionnalités.
Le graphe montre également qu'en charge au niveau consommation de temps CPU pour un même nombre de requêtes, les coûts des différents mécanismes de sécurité sont les suivants : DNS < DNSSEC < (DNS+IPsec) < (DNSSEC + IPsec). En effet, il apparaît alors que la configuration DNSSEC en charge est moins consommatrice en ressources CPU. Ceci s'explique car un contexte IPsec implique une opération de chiffrement, qui pour des tests unitaires est moins consommatrice que la gestion d'un contexte DNSSEC, mais qui par contre avec une croissance du nombre de contextes, et donc une gestion de ces derniers, s'avère plus consommatrice.

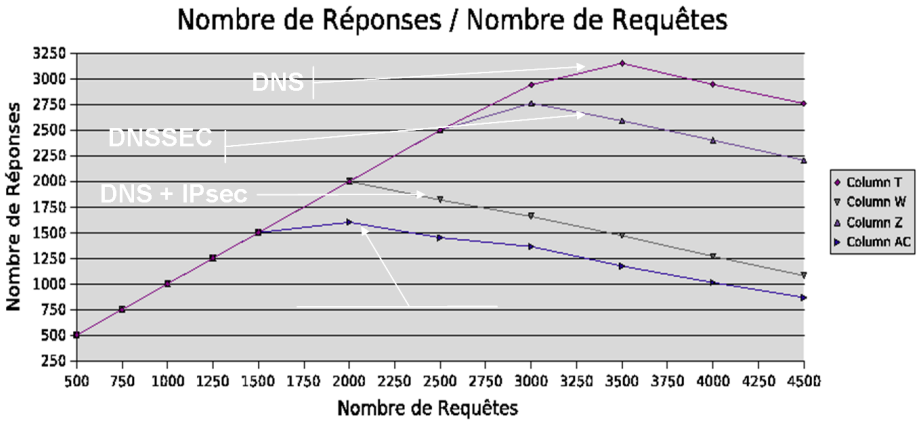

Fig. 4. Nombre de bonnes réponses fonction de nombre de requêtes, selon les différentes configurations DNS, (DNS + IPsec), DNSSEC, (DNSSEC + IPsec)

Toutefois, avant d'arriver à la saturation du serveur en tant que machine, on passe par un état intermédiaire qui correspond à une rupture du service DNS. Ce point de rupture a lieu à partir du moment où le serveur n'est plus capable de répondre à toutes les requêtes qui lui sont envoyées. On parlera dans ce cas de *point de rupture service*. L'Illustration 4 met en évidence ces points de rupture service sur chacune des configurations testées.

Dans ces tests le client envoie des requêtes directement sur le serveur dnssec1 (le cache - forwarder n'est pas utilisé).
L'Illustration 3 montre que pour les configurations IPsec les niveaux de charge peuvent atteindre 100% alors qu'elle n'atteint que 90% pour les autres configurations. On considère alors qu'un serveur commence à être surchargé à 90% de son niveau de charge maximale soit 90% CPU dans le premier cas et 81% CPU dans le deuxième cas. La position de point critique sur ce deuxième graphe dépend également de configuration (voir Tableau5).

Configuration	DNS	(DNS + IPsec)	DNSSEC	(DNSSEC + IPsec)
Point de rupture à 90% (90% de la charge maximale au point de rupture machine)	3000	2000	1800	1500
Coût de la sécurité	0%	+33%	+40%	+50%

Tab. 5. Les points de surcharge CPU pour un serveur dans les différentes configurations

Le Tableau 6 permet alors d'évaluer en termes de requêtes la charge maximale supportée par les serveurs DNS selon différentes configurations. Ce tableau confirme le coût supérieur en période de charge d'IPsec par rapport à DNSSEC. L'illustration 4 met en évidence que pour des petites valeurs de la charge, le nombre de réponses est égal au nombre de requêtes pour toutes les configurations. Pour des valeurs plus grandes, le graphe n'est plus linéaire. Le point de rupture service correspond au taux de requêtes maximal que le serveur est capable de traiter correctement. La position du point critique dépend de la configuration et est résumée dans le Tableau 5.

Configuration	DNS	DNSSEC	(DNS + IPsec)	(DNSSEC + IPsec)
Point de rupture service	3000	2500	2000	1500
Coût de la sécurité	0	+17%	+33%	+50%

Tab. 6. La capacité maximale de traitement pour un serveur de noms dans les différentes configurations

Ainsi, l'évaluation des différentes configurations de sécurité montre que le protocole DNSSEC est plus robuste qu'IPsec lors d'une situation de charge, Ainsi DNSSEC semble coûter deux fois moins chère qu'IPsec en terme de nombre maximal de requêtes DNS traitées.

5.2 Coût CPU et taux de réponses correctes en mode plateforme

Il s'agit toujours d'un test en charge, cette fois-ci l'objet du test étant la plateforme. Le client développé en Java envoie des requêtes à destination de la plateforme en utilisant comme point intermédiaire un relais.
Lors de ce test, le relais devient un goulot d'étranglement. Pour un taux de requêtes réduit, il gère bien le processus de résolution (près de 100%). Pour des taux plus élevés, il échoue de temps en temps pendant le processus de résolution en envoyant des réponses de type `SERVFAIL` à son client ou il rejette directement la requête et il n'envoie plus de réponses. Le client devrait distinguer maintenant parmi les réponses qu'il reçoit entre les messages correctes (`NO ERROR`) et les messages d'erreur (`SERVFAIL`).
Pour les tests des configurations utilisant DNSSEC, il y a deux possibilités : le client peut décider de demander au relais de ne pas faire une vérification de signature ou le laisser faire (option par défaut). La première option peut être spécifiée en mettant le bit `CD` (Checking Disable) à 1 dans la requête. Au lieu de quatre types de tests à faire, maintenant il y en a six car six combinaisons possibles.

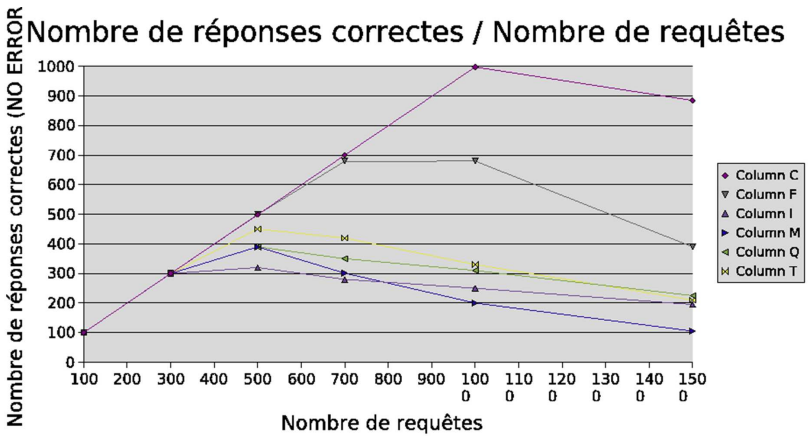

Fig. 5. Nombre de réponses correctes en fonction de nombre de requêtes, des configuration et de la vérification ou non de signatures DNS (C), (DNS + IPsec) (F), (DNSSEC + IPsec) avec vérification de signature (I), (DNSSEC + IPsec) sans vérification de signature (M), DNSSEC avec vérification de signature (Q), et DNSSEC sans vérifictaion de signature (T).

Tout comme dans le test antérieur, le graphe Nombre de Réponses correctes en fonction de Nombre de Requêtes (Illustration 5) fait ressortir un point de rupture par configuration étudiée. Les positions de ces points sont visibles dans le Tableau 7.

Config.	DNS	(DNS + IPsec)	DNSSEC (avec vérification de signature)	DNSSEC (sans vérification de signature)	(DNSSEC + IPsec) (avec vérification de signature)	(DNSSEC + IPsec) (sans vérification de signature)
Nombre maximal de requêtes répondues correctement	1000	700	400	450	300	400
Coût de la sécurité	0	+30%	+60%	+55%	+70%	+60%

Tab. 7. Capacité maximale de traitement de requêtes par une plateforme - test en charge de la plateforme

Le classement sur le coût de la sécurité que fait apparaître la comparaison du taux de réponse correctes / le nombre de requêtes envoyées est le suivant : DNS < (DNS+IPsec) < (DNSSEC sans vérifications) < (DNSSSEC avec vérifications) < (DNSSEC sans vérifications + IPsec) < (DNSSEC avec vérifications + IPsec). Les valeurs qui apparaissent dans le tableau peuvent paraître surprenantes à un premier regard, par rapport à celles trouvée lors des tests mono serveurs. La baisse de performances dans le cas de la plateforme est très importante si on sécurise le protocole DNS. Les nombres de requêtes résolues correctement sont beaucoup plus petits que les valeurs qui apparaissent dans le test d'un serveur individuel. Cette situation s'explique par le fait que le relais qui se charge de la résolution des requêtes ouvre des contextes et procède à des vérifications de signatures (si on lui le demande). Il devient vite surchargé, beaucoup plus vite que le reste des serveurs de l'architecture.

Vu le graphe dans l'Illustration 6 et en faisant un raisonnement similaire à celui du test précédent, il devient clair que les valeurs du nombre maximal de requêtes répondues correctement dans chaque configuration correspondent bien à des régimes de surcharge pour le relais. Ce test met en évidence l'importance de la prise en compte de l'architecture dans son ensemble. Dans ce cas particulier, il y a un goulot d'étranglement : le relais.
Les tests en charge ont nécessité le développement d'un outil spécifique. Ils ont mis en évidence le fait que les relais DNS sont des véritables goulots d'étranglement dans l'architecture et que, si on veut sécuriser le protocole DNS, cela va alourdir encore plus leur situation ce qui engendrera des chutes de performances importantes. Lors de tests, il ne suffit pas de se contenter d'étudier les serveurs DNS primaires. Il faudrait prendre en compte la totalité des éléments, notamment les relais.

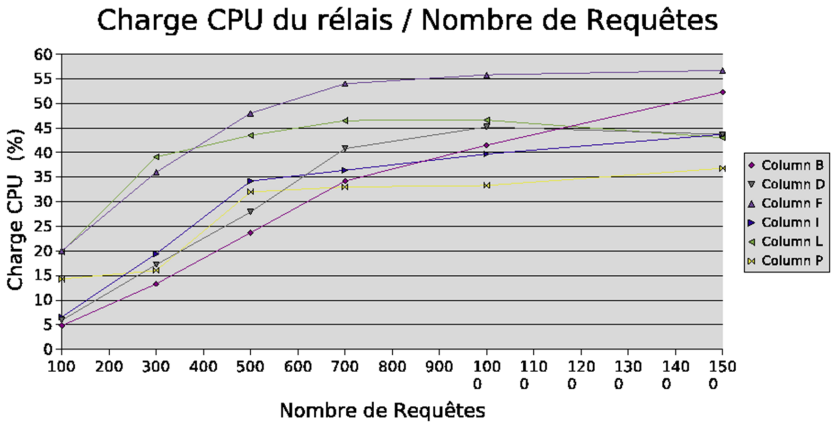

Fig. 6. La charge CPU du relais en fonction du nombre de requêtes, des configuration et de la vérification ou non de signatures DNS (B), (DNS + IPsec) (D), (DNSSEC + IPsec) avec vérification de signature (F), (DNSSEC + IPsec) sans vérification de signature (I), DNSSEC avec vérification de signature (L), et DNSSEC sans vérifictaion de signature (P).

6 Impact de la sécurité sur les temps de mise à jour

Mis à part les paramètres déjà vus dans les tests antérieurs, un élément jugé important dans le processus de choix de la démarche pour sécuriser le protocole DNS est le temps de mise à jour des données hébergées par les DNS. Ce paramètre peut être défini comme la durée de temps écoulée entre le moment où la mise à jour a été demandée et le moment où elle devient effective, i.e. visible dans les réponses des serveurs.
Dans nos tests le serveur maître met à jour son fichier de zone et notifie ensuite l'esclave des changements produits. Celui-ci récupère ces changements et les intègre à son tour dans sa copie de fichier de zone. Les deux redeviennent ainsi synchronisés. Le logiciel utilisé pour faire des mises à jour (respectant la rfc2136 [**?**]) est `nsupdate` qui vient avec la distribution `BIND 9.3.1`.
Les tests seront effectués sur des configurations DNS, (DNS+IPsec), (DNS+TSIG) et DNSSEC. TSIG utilise des clefs symétriques partagées pour signer des messages et authentifier de cette manière le client qui met à jour et l'esclave auprès du serveur maître. Le mécanisme TSIG est généralement utilisé pour sécuriser les transactions entre les serveurs. En effet, il n'est pas envisageable de passer à l'échelle un tel mécanisme, et de l'implémenter dans le cadre de consultations client, où chaque client serait authentifié par une clé symétrique.

Le test consiste à envoyer des commandes d'effacement pour N objets de type AAAA (IPv6) qui existe dans la zone et ensuite à envoyer des commandes pour la suppression des N objets supprimés.
Deux variantes sont possibles : envoyer toutes les commandes d'un même type

d'un coup, ou par plusieurs appels à la fonction `nsupdate`. Après l'envoi des N commandes de suppression, on lance la commande dig pour vérifier que toutes les suppressions ont été faites. Si on suppose que les commandes de mise à jour sont effectuées dans l'ordre de présentation au serveur, il suffit d'envoyer juste une requête ayant comme objet le $N^{\text{ième}}$ objet à supprimer. Si la réponse n'est pas `NXDOMAIN`, cela signifie que l'objet existe encore et la requête `dig` est répétée. La procédure est la même pour l'ajout, à la différence que l'appel à la fonction dig sera répété tant qu'on n'obtient pas de réponse `NO ERROR`. En pratique notre hypothèse de travail selon laquelle les mises à jour s'effectuaient dans l'ordre a été validé en utilisant un deuxième test plus coûteux qui consistait à observer la présence de tous les N objets (au lieu de un seul le $N^{\text{ième}}$).

Les Illustrations 7 et 8, montrent le fait que les différences entre les temps de

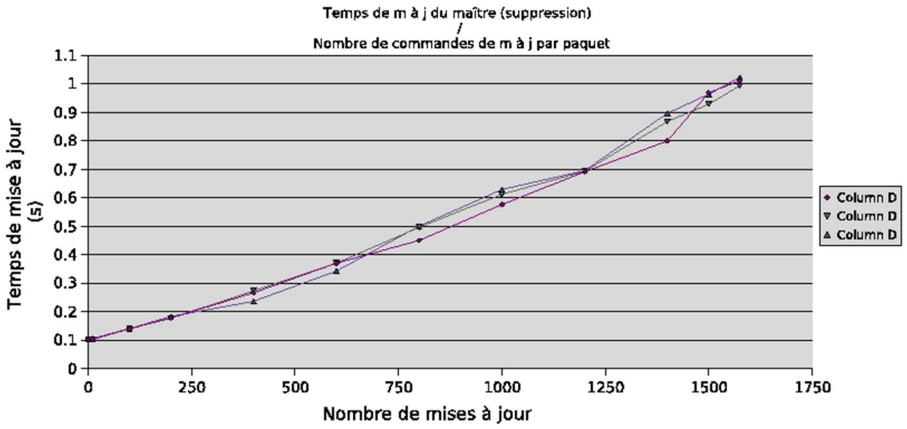

Fig. 7. Temps (en secondes) de mise à jour (suppression) du serveur maître en fonction du nombre de commandes dans le cas d'un seul appel nsupdate. DNS (rouge), (DNS + IPsec) (vert), (DNS + TSIG) (violet)

mise à jour (suppression) dans les trois configurations basées DNS ne sont pas significatives ni pour le serveur maître ni pour le serveur esclave. Pour le serveur esclave il y a un retard de 0.5 secondes par rapport au maître.

Pour l'ajout des enregistrements, les valeurs de temps de mise à jour sont à peu près les mêmes que dans le cas de la suppression et considérons que ce n'est pas important de les présenter.

En revanche, avec l'utilisation de DNSSEC (voir Illustration 9), il apparaît une grande différence entre les performances de la plateforme par rapport aux configurations DNS mais aussi entre les temps de la suppression et ces de l'ajout pour cette même configuration DNSSEC. Pour un test de 1000 enregistrements,

il y a un facteur de multiplication de 15 pour le temps de suppression et 75 pour l'ajout. Ces facteurs augmentent avec le nombre d'enregistrements concernés.

L'explication consiste dans le fait que le serveur DNSSEC doit signer de nouveau des enregistrements de la zone (tels que ceux de type NSEC) chaque fois qu'un changement est fait. L'effort est d'autant plus important dans le cas de l'ajout car il faut signer aussi les enregistrements ajoutés.

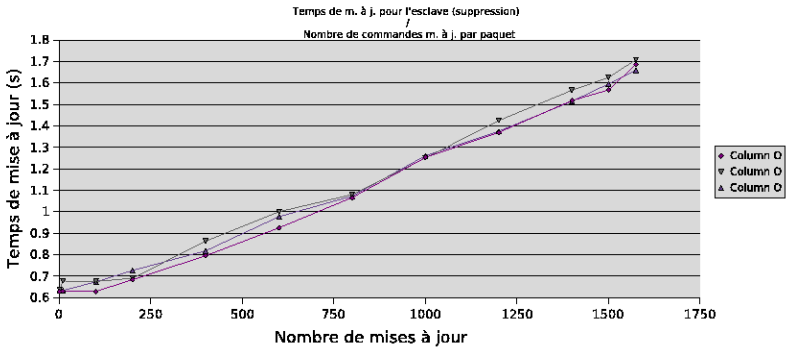

Fig. 8. Temps (en secondes) de mise à jour (suppression) du serveur esclave en fonction du nombre de commandes dans le cas d'un seul appel nsupdate. DNS (rouge), (DNS + IPsec) (vert), (DNS + TSIG) (violet)

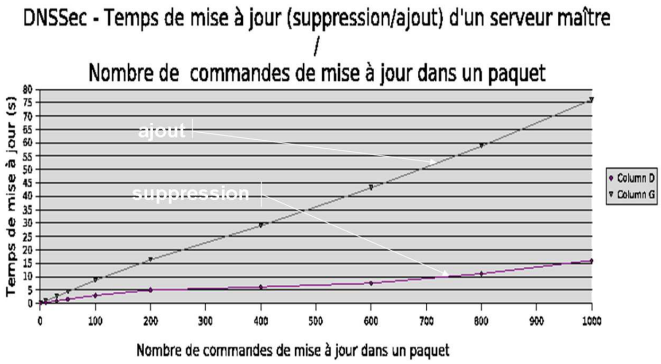

Fig. 9. Temps (en secondes) de mise à jour (suppression/ajout) du serveur maître en fonction du nombre de commandes dans le cas d'un seul appel nsupdate dans le cas d'une configuration DNSSEC, suppression (D), ajout (G)

7 Conclusion

Ces tests ont montré que la sécurité a un impact non négligeable sur les performances du système de nommage. Des paramètres tels que : le temps de réponse d'un serveur/une plateforme, la capacité maximale de traitement de requêtes pour un serveur/une plateforme et le temps de mise à jour des serveurs sont à prendre en compte lors du choix de ce type d'architecture.
TSIG s'est avéré assez peu coûteux lors de nos tests de mise à jour et il semble adapté pour sécuriser les transactions entre les serveurs. Assez simple à configurer, TSIG ne permet que d'offrir une garantie d'intégrité, et d'authentification. Il nécessite une configuration spécifique et manuelle, entre les entités impliquées auxquelles on attribue une clé symétrique. Aucun protocole de négociation n'est envisageable, et le passage à l'échelle de TSIG semble proscrit. IPsec par contre permet d'offrir un service d'intégrité et d'authentification des données. Il dispose d'un mécanisme de négociation de clefs (IKE). Une fois le tunnel établi, son coût est comparable à celui de TSIG. DNSSEC par contre, ne semble pas du tout adapté à des besoins de mises à jour fréquentes.
En ce qui concerne le temps de réponse d'une architecture peu chargée, la solution DNS + IPsec est meilleure par rapport à la solution DNSSEC. Rien d'étonnant, (DNSSEC+IPsec) est la pire en terme de dégradation de performances.
Pour la capacité de traitement de requêtes par un serveur de nom, la solution DNSSEC est meilleure que (DNS + IPsec). Par contre, quand il s'agit d'un test de plateforme, les performances chutent plus rapidement dans le cas de la configuration DNSSEC.
Ces tests on porté sur des technologies permettant de sécuriser le lien entre un nom de domaine et une adresse IP. DNSSEC permet d'assurer l'authentification des sources des enregistrements DNS et leur intégralité tandis que TSIG et IPsec permet d'authentifier le partenaire lors d'un échange de messages. IPsec propose également un service de confidentialité de données.
Pour aller encore plus loin et sécuriser le lien entre une adresse IP et une adresse Ethernet, SEND est un choix possible. Dans le cas de l'utilisation d'IPsec, les adresses IP peuvent être authentifiées. Par contre il est nécessaire de sécuriser le lien entre la couche IP et la couche Ethernet, sans quoi même avec la bonne adresse IP, il est impossible de garantir que le hôte qui reçoit le paquet est bien le hôte désiré. SEND et IPsec permettent de sécuriser les niveaux 2 et 3 du modèle ISO/OSI [**?**].

8 Remerciements

Les auteurs tiennent à remercier tout particulièrement Jean Michel Combes, pour ses conseils et ses remarques pertinentes.