Utilisation d'identifiants cryptographiques pour la sécurisation IPv6

Jean-Michel Combes

THESE DE DOCTORAT CONJOINT TELECOM SUDPARIS et L'UNIVERSITE PIERRE ET MARIE CURIE

Spécialité : Informatique

Ecole doctorale : Informatique, Télécommunications et Electronique de Paris

Présentée par

Jean-Michel Combes

Pour obtenir le grade de
DOCTEUR DE TELECOM SUDPARIS

Utilisation d'identifiants cryptographiques pour la sécurisation IPv6

Soutenue le 28 Septembre 2012

devant le jury composé de :
Directeur de thèse
LAURENT Maryline, Professeur HDR, Telecom SudParis (France)
Rapporteurs
BONNIN Jean-Marie, Professeur HDR, Telecom Bretagne (France)
VIHO César, Professeur HDR, IRISA (France)
Examinateurs
TIXEUIL Sébastien, Professeur HDR, Université Pierre et Marie Curie – Paris 6 (France)
EL KALAM Anas Abou, Professeur HDR, ENSA de Marrakech (Maroc)
SOUISSI Mohsen, Docteur et Responsable R&D, AFNIC (France)

Thèse n° 2012TELE0034

A mes parents.

"RTFM ! Sinon y'a le code ..."

Remerciements

L'aboutissement de ces quatre années de doctorat n'aurait pu avoir lieu sans l'aide et le soutien de nombreuses personnes que je tiens à remercier. Mais avant toute chose, je tiens à m'excuser auprès des personnes que j'oublie de citer.

Tout d'abord je tiens à remercier Maryline Laurent d'avoir accepté de devenir ma directrice de thèse et de m'avoir encadré pendant ces quatre années.

Je remercie aussi mes parents qui m'ont fait confiance tout au long de mes études et n'ont cessé de m'encourager dans mon travail. Je n'oublie pas le reste de ma famille, en particulier Alain, Mahaut et Laurent sur "Paris", qui me fournit, à chaque fois que je la vois, une bouffée d'air me redonnant des forces pour mon travail.

Je tiens à remercier aussi Jean-Luc Pons, mon ancien maître de stage à la Direction Départementale de France Télécom, à Albi, qui m'a fait découvrir cette entreprise et m'a conseillé de me renseigner sur les travaux de son centre de recherche, le CNET.

Je remercie Olivier Charles, mon ancien maître de stage au CNET, qui m'a fait confiance en me prenant comme stagiaire et m'a défendu âprement lors de ma soutenance de fin de parcours d'école d'ingénieur qui fut houleuse.

Je remercie aussi Thierry Baritaud et David Arditti qui m'ont proposé d'intégrer les "gars d'en bas" du bâtiment F à la suite de mon stage.

Je remercie ma hiérarchie (Matthieu, Sébastien, Pierre et Julie) qui a fait en sorte de me donner les meilleurs moyens afin de réaliser dans les meilleures conditions possibles cette thèse.

Je remercie aussi les "gars d'en haut" du bâtiment F (Henri, Bob, Côme, Gilles, Yannick, Thomas et Ryad) pour avoir supporté mes (trop ?) nombreuses questions en cryptographie.

Je remercie mon ancien collègue de bureau, Laurent, avec qui j'ai eu de bonnes rigolades au sujet d'un certain Kardogan. Celles-ci m'ont relaxé plus d'une fois pendant des périodes de stress.

Je remercie aussi les membres de la communauté IETF et de la délégation IETF de France Télécom avec qui j'ai passé de bons moments, très enrichissants sur le plan scientifique et humain, durant les meetings.

Je remercie enfin les stagiaires avec qui j'ai travaillé sur le sujet de ma thèse (Silviu, Tony, Aurélien et Ghada).

En dernier point, je m'excuse auprès de ma famille, mes amis et mes collègues, pour mes absences lorsque je faisais passer cette thèse avant eux.

Résumé

IPv6, protocole succédant à IPv4, est en cours de déploiement dans l'Internet. Il repose fortement sur le mécanisme *Neighbor Discovery Protocol* (**NDP**). Celui-ci permet non seulement à deux nœuds IPv6 de pouvoir communiquer, à l'instar du mécanisme *Address Resolution Protocol* (**ARP**) en IPv4, mais il apporte aussi de nouvelles fonctionnalités, telles que l'autoconfiguration d'adresse IPv6. Aussi, sa sécurisation pour le bon fonctionnement de l'Internet en IPv6 est critique. Son mécanisme de sécurité standardisée à l'*Internet Engineering Task Force* (**IETF**) se nomme *Secure Neighbor Discovery* (**SEND**). Il s'appuie à la fois sur l'utilisation d'identifiants cryptographiques, adresses IPv6 appelées *Cryptographically Generated Addresses* (**CGA**) et qui sont générées à partir d'une paire de clés publique/privée, et de certificats électroniques X.509. L'objet de cette thèse est l'étude de ces identifiants cryptographiques, les adresses CGA, ainsi que le mécanisme SEND les employant, et leurs réutilisations potentielles pour la sécurisation IPv6.

Dans une première partie de cette thèse, tout d'abord, nous posons l'état de l'art. Aussi, nous rappelons les principales caractéristiques du protocole IPv6 et en particulier son adressage qui inclut les adresses CGA. Ensuite, nous détaillons le mécanisme NDP et son utilisation pour l'autoconfiguration d'adresses IPv6. Enfin, nous rappelons le mécanisme de mobilité des nœuds IPv6, *Mobile IPv6* (**MIPv6**).

Dans une deuxième partie de cette thèse, nous nous intéressons à la fiabilité du principal mécanisme connu employant les adresses CGA, le mécanisme SEND. En premier lieu, nous décrivons les failles de sécurité du mécanisme NDP et présentons le mécanisme SEND. Ensuite, nous analysons les limites de ce mécanisme, ainsi que des adresses CGA, et nous proposons des améliorations pour y pallier.

Dans une troisième et dernière partie de cette thèse, nous présentons des utilisations des identifiants cryptographiques pour la sécurisation IPv6. Dans un premier temps, nous abordons la problématique de la lutte contre l'usurpation d'adresses IP source. Nous rappelons les menaces et les techniques existantes aujourd'hui, dont leurs limites, ne pouvant que les atténuer. Alors, nous décrivons les solutions *Source Address Validation Improvements* (**SAVI**), sur lesquelles nous avons contribué, qui permettent une amélioration de la protection contre l'usurpation d'adresses IP source et qui reposent, entre autres, sur l'utilisation des mécanismes NDP et SEND. Dans un deuxième temps, nous abordons la sécurisation au niveau de la couche IPv6 grâce à IPsec et en particulier au mécanisme *Internet Key Exchange version 2* (**IKEv2**) qui permet la configuration dynamique des connexions IPsec. Ce mécanisme emploie des méthodes d'authentification ayant certaines limites. Nous décrivons alors une méthode alternative reposant sur l'emploi des adresses CGA ainsi que son implémentation. Nous présentons finalement un cas concret d'utilisation dans le contexte de MIPv6. Dans un dernier temps, nous nous intéressons à la sécurisation de la mise à jour dynamique de l'architecture *Domain Name System* (**DNS**). Nous montrons que celle-ci souffre de limites, en particulier dans un contexte d'autoconfi-

guration d'adresses IPv6, et nous proposons une nouvelle solution de sécurité y remédiant. Cette dernière est basée sur deux types d'identifiants cryptographiques : l'*Identity-Based Cryptography* (**IBC**), où la partie publique d'une paire de clés publique/privée est l'identité, et les adresses CGA. Nous décrivons finalement comment nous avons implémenté cette solution.

Mots-clefs

IPv6, NDP, DNS, mobilité, sécurité, IPsec/IKEv2, cryptographie, CGA, IBC

Abstract

IPv6, next Internet protocol after IPv4, is under deployment in the Internet. It is strongly based on the *Neighbor Discovery Protocol* (**NDP**) mechanism. First, it allows two IPv6 nodes to communicate, like the *Address Resolution Protocol* (**ARP**) mechanism in IPv4, but it brings new functions too, as IPv6 address autoconfiguration. So, the security of this mechanism is critical for an Internet based on IPv6. The security mechanism standardized by the *Internet Engineering Task Force* (**IETF**) is *Secure Neighbor Discovery* (**SEND**). It is based on the use of cryptographical identifiers, IPv6 addresses named *Cryptographically Generated Addresses* (**CGA**) and generated from a public/private keys pair, and X.509 certificates. The goal of this PhD thesis is the study of such cryptographical identifiers, CGA addresses, as well as SEND using them, and their potential re-use to secure IPv6.

In a first part of this thesis, we recall the main features of the IPv6 protocol and especially its addressing, which includes the CGA addresses. Next, we describe the NDP mechanism and its use for the IPv6 autoconfiguration. Finally, we recall the protocol providing the IPv6 nodes mobility, *Mobile IPv6* (**MIPv6**).

In a second part of this thesis, we are interested in the reliability of the main known mechanism using the CGA addresses, SEND. First, we describe the security threats of the NDP mechanism and we present SEND specifications. Next, we analyze its limitations and we propose improvements for SEND and CGA addresses.

In a third and last part of this thesis, we present different uses of cryptographical identifiers to secure IPv6. In a first step, we look at protection against source IP address spoofing. We recall current security threats and methods, as well as their limitations, to mitigate them. Then, we describe *Source Address Validation Improvements* (**SAVI**) solutions, on which we contributed, that improve the protection against source IP address spoofing. These solutions are based on IPv6 addresses assignment protocols, like NDP and SEND mechanisms. In a second step, we look at the security of the IPv6 layer, provided with IPsec, and especially the *Internet Key Exchange version 2* (**IKEv2**) mechanism that configures dynamically IPsec connections. This mechanism uses authentication methods having limitations. So, we describe a new alternative method based on CGA addresses and its implementation. Finally, we present a concrete use case in the IPv6 mobility context. In a last step, we look at the security of the *Domain Name System* (**DNS**) dynamic update. We show this one suffers from limitations in an IPv6 autoconfiguration context and we propose a new method solving them. This method is based on two types of cryptographic identifiers: *Identity-Based Cryptography* (**IBC**), where the public part from a public/private keys pair is the identity itself, and the CGA addresses.

Keywords

IPv6, NDP, DNS, mobility, security, IPsec/IKEv2, cryptography, CGA, IBC

Table des matières

Introduction

Contexte

Dans les années 1970, des universitaires américains ont spécifié et expérimenté un nouveau moyen de communiquer à distance. Ces travaux ont été financés par le Ministère de la Défense américaine, *Department of Defense* (**DoD**). En effet, jusqu'à cette époque, les techniques de communication reposaient sur une architecture centralisée. Or, en cas de guerre, il aurait été simple de paralyser les communications d'un pays en frappant ces points vitaux. Suite à ces travaux, les universitaires ont proposé une nouvelle méthode de communication. Les messages échangés sont découpés en paquets qui sont acheminés dans une architecture regroupant de multiples réseaux. L'*Internet* était né et allait révolutionner le monde.

Cette nouvelle technologie a été standardisée, au sein de l'*Internet Engineering Task Force* (**IETF**), sous le nom de *Internet Protocol version 4* (**IPv4**). Avec le temps, de nombreux services ont été créés en se reposant sur IPv4, comme le courriel et le *Web* dans un premier temps, ou encore, la téléphonie et la télévision sur IP dans un passé proche. En parallèle, l'*Internet* s'étendait dans le monde avec de plus en plus d'utilisateurs qui s'y connectaient. Malheureusement, l'IETF n'avait pas prévu un tel engouement et les spécifications d'IPv4 n'étaient pas adaptées pour y répondre.

Aussi, l'IETF a spécifié un nouveau protocole qui doit succéder à IPv4 et qui se nomme *Internet Protocol version 6* (**IPv6**). IPv6 devrait permettre de répondre aux besoins toujours plus grands des utilisateurs de l'*Internet*. Les premiers déploiements d'IPv6 ont eu lieu mais le chemin pour avoir tout l'Internet reposant uniquement sur IPv6 est encore long. La principale raison est généralement dûe à la méconnaissance par les principaux acteurs de l'Internet (i.e, constructeurs, opérateurs) de ce nouveau protocole et à la peur qu'il entraîne, en particulier, vis-à-vis de sa sécurité.

Nos travaux précédents ont porté sur l'étude des différents matériels de sécurité pouvant être utilisés pour la sécurisation IPv6. Tout d'abord, lors du projet RNRT MobiSecV6 [1], nous nous sommes intéressés dans un premier temps à l'utilisation des clés secrètes partagées. Nous nous sommes rendus compte la distribution de telles clés devenait complexe lors d'un passage à grande échelle. Aussi, dans un deuxième temps, nous nous sommes penchés sur l'utilisation de certificats électroniques de type X.509. Même si plus faciles à fournir, ils obligent le déploiement d'une infrastructure de confiance dédiée, une *Public Key Infrastructure* (**PKI**). Celle-ci engendre des coûts supplémentaires et introduit de potentielles failles de sécurité. Plus important, chacun des protagonistes participant à une communication sécurisée doit avoir une relation de confiance avec la PKI de l'interlocuteur. Or, ceci n'est pas toujours le cas. Aussi, lors du projet RNRT IDsA [2], nous avons désiré savoir s'il était possible de se reposer sur une unique infrastructure de

1. http://www.inrialpes.fr/planete/people/ccastel/MOBISECv6/rnrt.html
2. http://idsa.irisa.fr/

confiance qui serait universelle. Celle-ci est l'infrastructure *Domain Name System* (**DNS**) mais, auparavant, il est nécessaire de la sécuriser grâce au mécanisme *Domain Name System Security Extensions* (**DNSSEC**). Malheureusement, le déploiement de DNSSEC n'est pas encore globale. Enfin, lors du projet Nautilus6 [3], nous avons étudié l'utilisation du mécanisme *Extensible Authentication Protocol* (**EAP**) comme matériel de sécurité [BVB+06][LBC+06][BCLML07]. Malheureusement, ce mécanisme nécessite aussi généralement une infrastructure dédiée de type *Authentication, Authorization and Accounting* (**AAA**) impliquant les mêmes inconvénients qu'une PKI.

L'objectif de cette thèse est d'étudier cette sécurisation du protocole IPv6, et plus spécialement en nous intéressant à l'utilisation d'identifiants cryptographiques.

Le mécanisme Neighbor Discovery Protocol (NDP) est le cœur du protocole IPv6. Non seulement, il permet, comme le mécanisme *Address resolution protocol* (**ARP**) en IPv4, à deux nœuds IPv6 de communiquer mais en plus il apporte une nouvelle fonctionnalité : l'autoconfiguration d'adresse IPv6. En IPv4, il est nécessaire de mettre en place dans les réseaux des entités intégrant le mécanisme *Dynamic Host Configuration Protocol* (**DHCP**) pour qu'un nœud puisse acquérir une adresse IP. L'autoconfiguration IPv6, elle, permet à un nœud IPv6 de s'attribuer une adresse IPv6 sans avoir à déployer des entités dédiées dans les réseaux. Ceci a facilité la spécification de mécanismes de mobilité des nœuds en IPv6.

Comme le mécanisme NDP est essentiel au bon fonctionnement du protocole IPv6, il est nécessaire que sa sécurité soit bien assurée. Aussi, l'IETF a standardisé le mécanisme *Secure Neighbor Discovery* (**SEND**) pour sa sécurisation. Celui-ci repose principalement sur l'utilisation d'identifiants cryptographiques, qui ne sont autres que des adresses IPv6 spécifiques, les adresses de type *Cryptographically Generated Addresses* (**CGA**). Une adresse CGA fournit des propriétés cryptographiques permettant de prouver sa possession.

Nos travaux se sont portés dans un premier temps sur l'analyse du mécanisme SEND, et donc des adresses CGA, afin de déterminer les faiblesses existantes de ce mécanisme et proposer des solutions d'amélioration.

L'Internet est le lieu d'attaques ciblant différents services ou utilisateurs s'y trouvant. Afin d'éviter de se faire localiser, les attaquants camouflent la localisation de la source de leurs attaques. Que cela soit en IPv4 ou en IPv6, cette localisation est l'adresse IP. Pour s'en prémunir, l'IETF a standardisé la technique appelée *Ingress Network Filtering*. Elle a réussit à limiter les attaques mais comporte des limites, en particulier concernant la précision de la source de potentielles attaques. Aussi, l'IETF a décidé de standardiser, dans le groupe de travail *Source Address Validation Improvements* (**SAVI**), des solutions y remédiant et s'appuyant sur la signalisation des mécanismes d'assignation d'adresses IP, dont le mécanisme NDP et sa version sécurisée SEND.

Dans un deuxième temps, nos travaux nous ont conduit à participer à la standardisation de ces mécanismes.

La sécurité des communications au niveau de la couche IP, que cela soit pour IPv4 ou IPv6, repose sur la famille de protocoles nommée *Internet Protocol security* (**IPsec**). Celle-ci permet de fournir authentification, confidentialité et intégrité des données. Des mécanismes, tels que *Internet Key Exchange Protocol Version 2* (**IKEv2**), permettent de configurer dynamiquement la sécurisation de ces communications. IKEv2 utilise différentes méthodes d'authentification quand deux nœuds désirent mettre en place une connexion sécurisée avec IPsec.

Nos travaux se sont alors intéressés à l'analyse des méthodes d'authentification utilisées par IKEv2 et nous avons recherché à fournir une nouvelle méthode palliant les défauts de

3. http://www.nautilus6.org/

celles existantes, en utilisant les propriétés cryptographiques des adresses CGA.

Bien que l'adresse IP, que cela soit en IPv4 ou IPv6, désigne un nœud dans l'*Internet*, pour des raisons de commodité, il est plus simple de donner un nom à un nœud, *Fully Qualified Domain Name* (**FQDN**), et d'utiliser ce nom (e.g., `example.foo.com`) lors de manipulations. L'architecture *Domain Name System* (**DNS**) permet de stocker le lien entre le FQDN d'un nœud et son adresse IP. Ce lien peut être mis à jour de manière dynamique et sécurisée. Dans un contexte d'autoconfiguration IPv6, les méthodes fournissant une telle sécurisation sont mal adaptées.

La dernière partie de nos travaux s'est penchée sur l'emploi des identifiants cryptographiques pour proposer une nouvelle méthode de sécurisation de la mise à jour DNS. Dans cette partie, en plus de l'utilisation des adresses CGA, nous avons élargi le domaine des identifiants cryptographiques à l'*Identity-Based Cryptography* (**IBC**).

Contributions de la thèse

A travers cette thèse, après un rappel des principales caractéristiques du protocole IPv6, du mécanisme NDP et des adresses CGA (Chapitre 1), nous avons analysé l'utilisation de ces dernières avec le mécanisme SEND (Chapitre 2). Puis, nous avons étudié l'emploi du mécanisme NDP et SEND pour la spécification de solutions SAVI (Chapitre 3). Finalement, nous avons désiré utiliser les propriétés cryptographiques des adresses CGA pour pallier les limites des méthodes de sécurisation de services : IPsec/IKEv2 (Chapitre 4) et la mise à jour DNS (Chapitre 5).

Découverte de limites au mécanisme SEND

Dans le Chapitre 2, nous avons analysé le mécanisme SEND et nos travaux ont permis de découvrir plusieurs faiblesses dans celui-ci, basées soit sur les algorithmes cryptographiques employés, soit sur l'immaturité des spécifications. Nous avons implémenté en Python certaines d'entre elles pour en vérifier la véracité. Tout d'abord, nous avons pu démontrer qu'il était possible de rejouer certains messages durant une procédure d'autoconfiguration en IPv6, malgré la sécurisation apportée par le mécanisme SEND [CC08]. Ceci entraîne que le nœud IPv6 ciblé par l'attaque ne pourra pas s'assigner une adresse IPv6. Ensuite, nous avons montré les limites de ce mécanisme lors de la sécurisation d'un service en mobilité IPv6 [DCLM08]. De plus, nous avons informé l'IETF de l'impossibilité d'utiliser le mécanisme SEND lors de certains scénarios [CKD10]. Ceci a conduit l'IETF à créer le groupe de travail *CGA & SEND maIntenance* (**CSI**) pour standardiser une solution à la problématique que nous avons soulevée. Enfin, nous avons proposé une solution permettant de limiter l'utilisation de la cryptographie asymétrique entre un nœud IPv6 et un routeur [XKH+08], afin d'optimiser les ressources de ce dernier.

Supervision du groupe de travail SAVI à l'IETF et proposition d'intégration de solutions SAVI dans l'architecture IPv6 ADSL/Fibre

Le groupe de travail SAVI à l'IETF standardise des solutions permettant de lutter contre l'usurpation d'adresse IP source et s'appuyant sur la signalisation des mécanismes d'assignation d'adresse IP, et en particulier les mécanismes NDP et SEND [CL12]. En tant que président de ce groupe de travail, nous avons supervisé, contribué et porté ces solutions pour qu'elles deviennent les standards de demain : à savoir, FCFS SAVI, SEND SAVI, DHCP SAVI (dont nous ne parlons pas dans cette thèse) et MIX SAVI. Par ailleurs, nous avons proposé à l'IETF comment intégrer FCFS SAVI, et SEND SAVI dans le cas

où le mécanisme SEND est déployé, dans une architecture IPv6 de type ADSL/Fibre standardisée au Broadband Forum (BBF) [CPLC12]. Notre proposition a été adoptée et deviendra prochainement une RFC (d'ici Octobre 2012, si tout se déroule normalement). Notre solution sera ainsi mise en œuvre par tout opérateur déployant IPv6 dans une telle architecture. Le Chapitre 3 détaille les résultats de nos travaux.

Méthode d'authentification pour IKEv2 reposant sur les adresses CGA

Désirant réutiliser les propriétés cryptographiques des adresses CGA afin de sécuriser des services en IPv6, nous nous sommes tout d'abord intéressés à la sécurisation du protocole IPv6 lui-même. Celle-ci repose sur l'utilisation de la famille de protocoles IPsec et dont la configuration dynamique s'effectue grâce au mécanisme IKEv2. A la suite de nos travaux, nous avons spécifié et implémenté une nouvelle méthode d'authentification pour IKEv2 utilisant les adresses CGA [CWL11] [CWL12]. En particulier, elle permet d'utiliser IKEv2 sans avoir à mettre en place une infrastructure de confiance, telle qu'une *Public Key Infrastructure* (**PKI**), comme c'est le cas avec les certificats X.509, tout en assurant le même niveau de sécurité. de Par ailleurs, nous avons proposé à l'IETF une solution de sécurisation, basée sur IPsec, de la mobilité IPv6 pouvant reprendre cette méthode d'authentification [DC08]. Cette solution apporte un niveau de sécurité supérieur à celle standardisée actuellement, qui peut faire l'objet d'une attaque de type *Man in the Middle* (**MitM**). Cette contribution est décrite dans le Chapitre 4.

Solution de sécurisation de la mise à jour DNS reposant sur les adresses CGA et IBC

Enfin, notre dernière contribution, décrite dans le Chapitre 5, concerne la sécurisation d'un autre service en IPv6 qui est la mise à jour dynamique des DNS. Nous avons spécifié et implémenté une nouvelle solution de sécurisation de cette mise à jour, plus adaptée dans un contexte d'autoconfiguration IPv6 que les solutions existantes [CAL12]. Communément en IPv4, cette sécurisation repose sur la présence d'un serveur DHCP, qui n'est plus présent dans le contexte d'autoconfiguration IPv6. Cette solution utilise encore une fois les propriétés cryptographiques des adresses CGA mais aussi l'IBC. Avec elle, il n'est plus nécessaire de stocker dans le DNS le matériel de sécurité nécessaire à l'authentification des données lors des mises à jour. De plus, elle permet de vérifier la possession de l'adresse IPv6 mise à jour, en plus de celle du FQDN. A notre connaissance, aucune autre méthode de sécurisation à ce jour n'a une telle fonctionnalité.

Chapitre 1

IPv6 et protocole de découverte des voisins

Le protocole IPv4 [Pos81] souffre de nombreuses faiblesses. Le principal problème est l'espace d'adressage. En effet, les adresses IPv4 sont d'une longueur de 32 bits, ce qui représente environ 4 milliards d'adresses possibles. Suite à l'explosion de la croissance du réseau Internet et le gaspillage des adresses dû à la structure en classes, le nombre d'adresses IPv4 est devenu insuffisant. Un autre problème se pose sur la saturation des tables de routage dans les routeurs principaux de l'Internet. Même si dès 1993 [EF94][SE01], des mesures d'urgence ont été prises, cela ne permet que de retarder l'échéance. Aussi, l'*Internet Engineering Task Force* (**IETF**) a lancé des travaux en 1994 afin de spécifier le protocole Internet qui remplacera IPv4 : ce protocole est IPv6.

Dans ce chapitre, tout d'abord, nous rappelons les différents types et classes d'adresse IPv6 spécifiés à l'IETF. Ensuite, nous décrivons le protocole de découverte des voisins, *Neighbor Discovery Protocol* (**NDP**), ainsi que le mécanisme d'autoconfiguration d'adresses IPv6, *Stateless Address Autoconfiguration* (**SLAAC**), reposant sur ce dernier. Par la suite, nous présentons le mécanisme de Neighbor Discovery Proxy nécessaire pour le bon fonctionnement de NDP dans certaines architectures. Enfin nous rappelons le mécanisme de mobilité en IPv6.

1.1 Classes d'adresses IPv6

Les adresses IPv6 [DH98] sont codées sur 128 bits alors que les adresses IPv4 l'étaient sur 32 bits, comme le montre la Figure 1.1 illustrant les en-têtes IPv4 et IPv6.

Il existe plusieurs classes d'adresses IPv6, correspondant à la portée de l'adresse, et plusieurs types d'adresses IPv6, correspondant à la procédure ayant servi à générer l'adresse IPv6. La Figure 1.3 résume les propriétés de ces différentes classes d'adresses.

IPv6 définit 3 types de classes [HD06] : *unicast*, *anycast* et *multicast*. La classe *multicast* remplace la classe *broadcast* en IPv4 et la classe *anycast* est une nouvelle classe par rapport à IPv4.

Au passage, IPv6 définit `::0/128` comme l'adresse non spécifiée, *unspecified address*, et `::1/128` comme l'adresse dite de *loopback*, équivalente à `127.0.0.1` en IPv4.

1.1.1 Unicast

Une adresse *unicast* IPv6 a la même propriété qu'en IPv4 : un paquet, dont l'adresse destination est *unicast*, sera réceptionné par un seul nœud, le nœud dont cette adresse

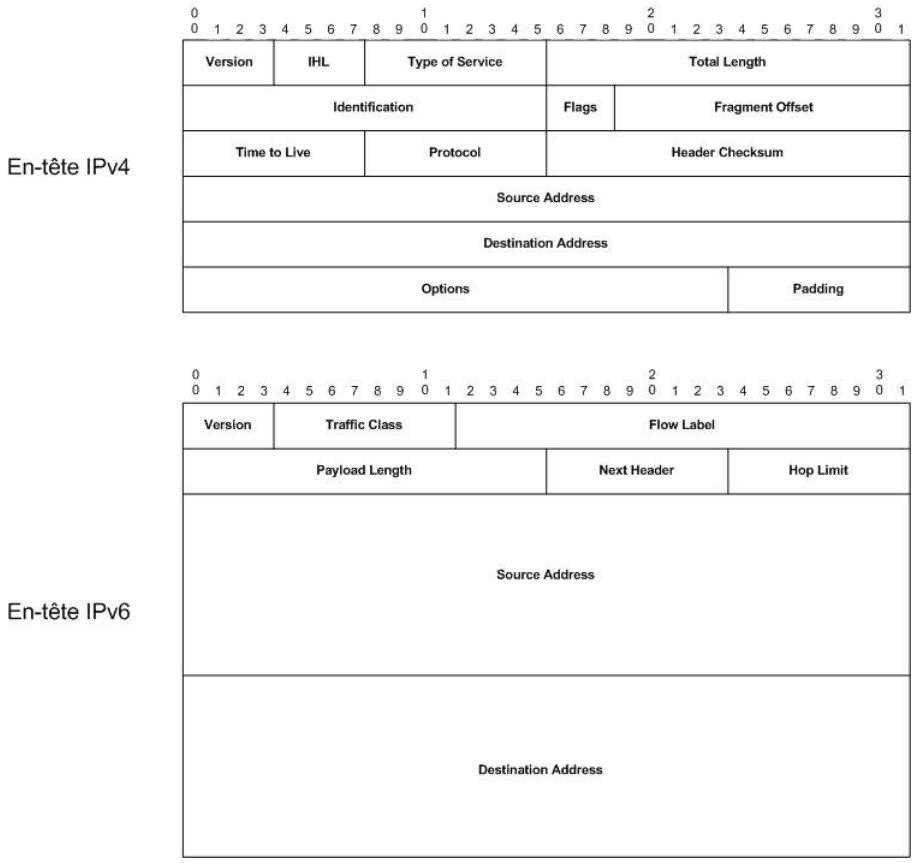

Figure 1.1 – En-têtes IPv4 et IPv6

a été assignée. Une adresse *unicast* est constituée de 2 parties, comme illustré dans la Figure 1.2, la première renseignant sur le préfixe réseau et la deuxième indiquant ce qui est appelé *Interface Identifier* (**IID**). Sauf exceptions (e.g., pour les liens réseau de type *Point-to-Point* présentés dans la Section 2.2.2), chacune de ces parties fait 64 bits.

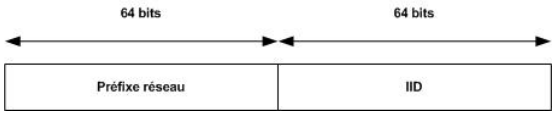

Figure 1.2 – Format d'une adresse unicast IPv6

Lien-local

Lorsque le préfixe réseau de l'adresse source ou destination est `FE80::/64`, cela indique que la communication est restreinte au lien réseau local (i.e., le paquet ne doit pas être acheminé via un routeur) et ce genre d'adresse est appelé adresse IPv6 lien-local, *Link-local Unicast Address* (**LUA**).

Privée

Comme en IPv4 [RMK+96], IPv6 spécifie un adressage "privé". Une adresse IPv6 de ce genre est appelée *Unique Local Address* (**ULA**) [HH05] et son préfixe réseau commence par `FC00::/7`. Elle ne peut être "routée" qu'à l'intérieur d'un périmètre spécifique.

Globale

Une adresse IPv6 est dite globale, équivalente à l'adresse IPv4 dite "publique", lorsque le préfixe réseau commence par `2000::/3`[1].

1.1.2 Multicast

Une adresse *multicast* IPv6 a la la même propriété qu'en IPv4 : un paquet, dont l'adresse destination est *multicast*, sera réceptionné par un groupe de nœuds dont l'adresse *multicast* a été assignée.

1.1.3 Anycast

Cette nouvelle classe d'adresse en IPv6 désigne une adresse qui est utilisée par plusieurs nœuds à la fois. Contrairement à la classe *multicast*, via des mécanismes non explicités ici, un paquet dont l'adresse destination est *anycast* ne sera reçu que par un des nœuds possédant cette adresse. Le format d'une adresse *anycast* est le même que celui d'une adresse *unicast* : il est impossible de différencier une adresse *anycast* d'une adresse *unicast*. Seul le nœud, dont l'adresse est assignée, sait s'il s'agit d'une adresse *anycast* ou *unicast*.

1.2 Types d'adresse unicast IPv6

Il existe plusieurs façons de générer dynamiquement les IID pour les adresses *unicast* IPv6.

1.2.1 EUI-64 modifié

La plus commune des méthodes est basée [HD06] sur l'identifiant IEEE modifié EUI-64[2], qui lui même repose sur l'adresse IEEE 802, plus connue sous le nom d'adresse *Media Access Control* (**MAC**). A partir d'une adresse MAC, il faut insérer `0xFFFE` au milieu de celle-ci afin d'obtenir l'identifiant IEEE EUI-64. Ensuite, il suffit de mettre à 1 le bit "universel/local" (i.e., 7ème bit du premier octet) et à 0 le bit "individuel/groupe" (i.e., 8ème bit du premier octet). Cette méthode est résumée dans la Figure 1.4.

1.2.2 Aléatoire

L'inconvénient de la méthode précédente est que le nœud IPv6 aura toujours le même IID même s'il change de préfixe réseau. Du coup, il serait facile d'identifier le nœud en question malgré ses mouvements. Aussi, une autre méthode [NDK07] a été spécifiée et est basée sur une génération pseudo-aléatoire. Lors de son démarrage, le nœud IPv6 initie aléatoirement une variable de 64 bits, nommée *history value* et qu'il sauvegarde. Lorsque le nœud a besoin d'un IID, tout d'abord, il génére un IID de type EUI-64 qu'il concatène à la valeur actuelle de *history value*. Ensuite, il applique sur le résultat la fonction de hachage

1. http://www.iana.org/assignments/ipv6-address-space/ipv6-address-space.xml
2. http://standards.ieee.org/develop/regauth/tut/eui64.pdf

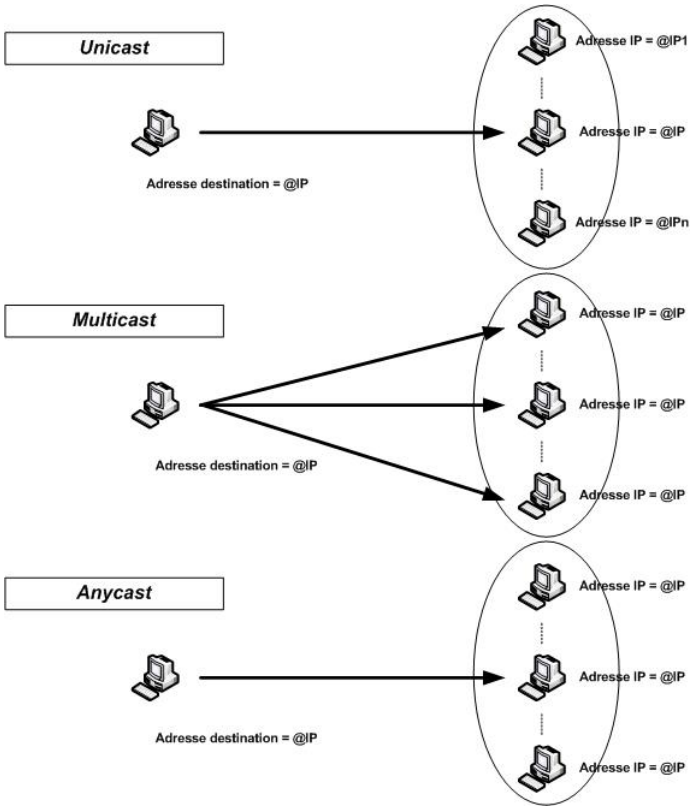

Figure 1.3 – Propriétés des différentes classes d'adresses IPv6

MD5 [Riv92]. Les 64 derniers bits en sortie sont sauvegardés comme nouvelle valeur pour *history value*. Il prend les 64 premiers bits en sortie et met le 7ème bit du premier octet à 0. Si cette valeur correspond à une valeur d'IID déjà utilisée par le nœud ou réservée, il recommence la procédure depuis son début. Sinon, il utilise la valeur calculée comme IID. Cette méthode est résumée dans la Figure 1.5, où `rand(n)` est une fonction retournant aléatoirement une valeur de n bits, `mod_EUI-64()` est une fonction retournant un IID de type EUI-64 modifié, `md5(x)` est une fonction appliquant la fonction de hachage MD5 sur la valeur x, `bit7(x)` est une fonction mettant le 7ème bit de x à 0, `firt64(x)` est une fonction retournant les 64 premiers bits de la valeur x et `last64(x)` est une fonction retournant les 64 derniers bits de la valeur x.

1.2.3 Aléatoire et stable

La méthode précédente génère à chaque fois un IID différent même si le nœud IPv6 reste dans le même réseau. Ceci peut ne pas faciliter la tâche de l'administrateur de ce réseau si celui-ci effectue une surveillance de son réseau. Aussi, récemment, une autre méthode a été proposée [Gon12a] qui permet de générer un IID aléatoire mais stable pour un même réseau (i.e., préfixe réseau). Cette méthode est encore en cours de spécification et donc pourrait être modifiée dans le futur. Pour générer un IID, le nœud dans un premier temps applique une fonction pseudo-aléatoire (e.g., MD5 comme précédemment ou SHA-1 [iost02]), *Pseudo-Random Function* (**PRF**), sur la concaténation du préfixe réseau, de l'IID basé sur EUI-64 modifié, l'identifiant du réseau s'il existe (e.g., IEEE 802.11 SSID) et une clé secrète qui a été généré au moment du démarrage du nœud IPv6. Ensuite,

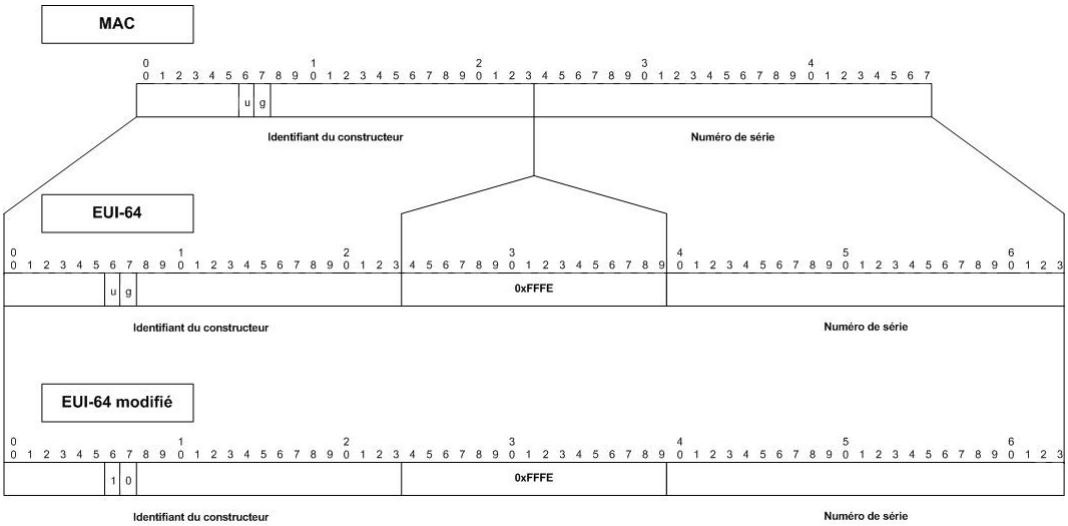

Figure 1.4 – Génération de l'IID à partir d'une adresse MAC

il prend les 64 premiers bits en sortie et met le 7ème bit du premier octet à 0. Cette méthode est résumée dans la Figure 1.6, où `gen()` est une fonction générant une clé secrète, `mod_EUI-64()` est une fonction retournant un IID de type EUI-64 modifié, `F(x)` est une fonction PRF, `bit7(x)` est une fonction mettant le 7ème bit de x à 0 et `firt64(x)` est une fonction retournant les 64 premiers bits de la valeur x.

1.2.4 Cryptographiques

Une autre méthode [Aur] de génération d'adresses IPv6 se rapproche de la méthode précédente. En effet, elle utilise la clé publique d'une paire de clés publique/privée pour générer l'IID. Les adresses produites ainsi s'appellent des adresses générées cryptographiquement, *Cryptographically Generated Addresses* (**CGA**) [Aur05]. Chaque adresse CGA est associée à des paramètres qui ont permis sa génération. Le paramètre *Sec* détermine la robustesse de l'adresse CGA contre les attaques de type brute-force et sa valeur varie de 0 à 7. Le paramètre *Modifier* permet d'améliorer l'effet aléatoire lors de la phase de génération de l'adresse CGA. Le paramètre *Subnet Prefix* est le préfixe réseau utilisé lors de la génération de la CGA. Le paramètre *Collision Count* indique le nombre de collisions rencontrées lors des procédures *Duplicate Address Detection* (cf. Section 1.4.2) pendant la génération de l'adresse CGA et sa valeur varie de 0 à 2. Le paramètre *Public Key* est la clé publique de la paire de clés publique/privée générée grâce à RSA [RSA78] par le possesseur de l'adresse CGA. Enfin, le paramètre *Extension Fields*, optionnel, contient les potentiels futurs paramètres rajoutés pour la spécification CGA. Tous ces paramètres, à part le paramètre *Sec* qui est codé directement dans l'adresse CGA, sont associés à une structure nommée *CGA Parameters*, illustrée dans la Figure 1.7.

Génération

Une adresse CGA est générée en plusieurs étapes en supposant qu'au préalable le nœud possède une paire de clés publique/privée RSA et que le paramètre *Sec* a été fixé. Tout d'abord, le paramètre *Modifier* est initialisé aléatoirement. La deuxième étape consiste à concaténer le paramètre *Modifier* à la valeur `0x000000000000000000`, suivi du paramètre

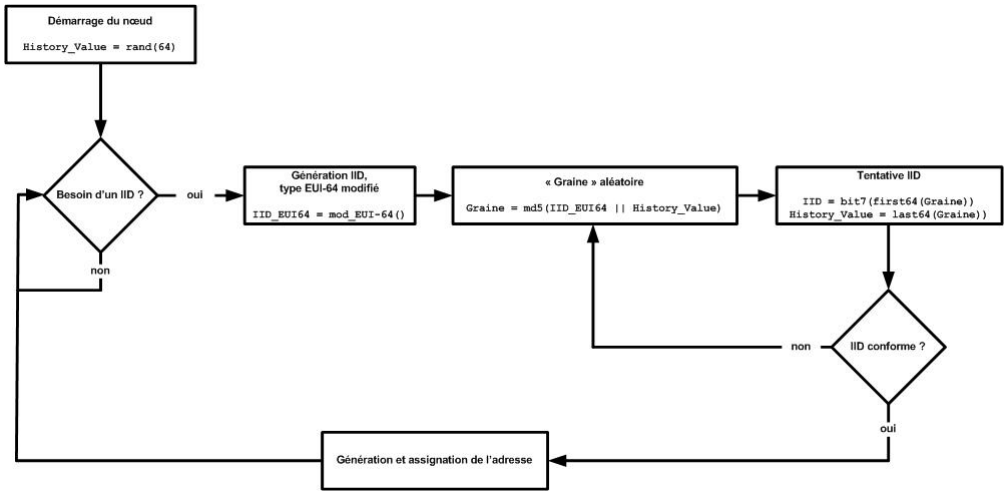

Figure 1.5 – Génération aléatoire de l'IID

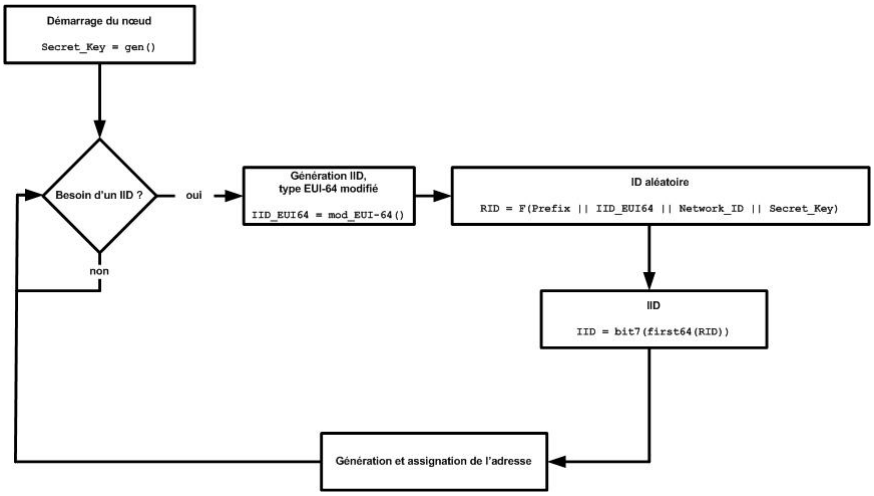

Figure 1.6 – Génération aléatoire et stable de l'IID

Public Key, ayant comme valeur la clé publique de la paire de clés précédemment citée, et finalement, s'il y en a, les données concernant les potentielles extensions du paramètre *Extension Fields*. La fonction de hachage SHA-1 est appliquée sur cette concaténation. Les 112 premiers bits du résultat sont appelés *Hash2*. L'étape suivante compare les "*Sec* x 16" premiers bits à 0. Si ce n'est pas le cas alors le paramètre *Modifier* est incrémenté de 1 et le processus revient à la deuxième étape. S'ils sont à 0 ou alors le paramètre *Sec* est égal à 0, le paramètre *Collision Count* est mis à la valeur `0x00`. La cinquième étape consiste à concaténer le paramètre *Modifier* au paramètre *Subnet Prefix*, suivi du paramètre *Collision Count*, suivi du paramètre *Public Key* et finalement, s'il y en a, les données concernant les potentielles extensions du paramètre *Extension Fields*. La fonction de hachage SHA-1 est appliquée sur cette concaténation. Les 64 premiers bits du résultat sont appelés *Hash1*. Durant la sixième étape, les 3 premiers bits de *Hash1* sont remplacés par le paramètre *Sec* et les bits 6 et 7 sont mis à 0. L'étape suivante forme l'adresse IPv6 en agrégeant le *Subnet Prefix* à la précédente valeur. La huitième étape consiste à appliquer la procédure *Duplicate Address Detection* (cf. Section 1.4.2) sur l'adresse générée. Si celle-ci n'est pas

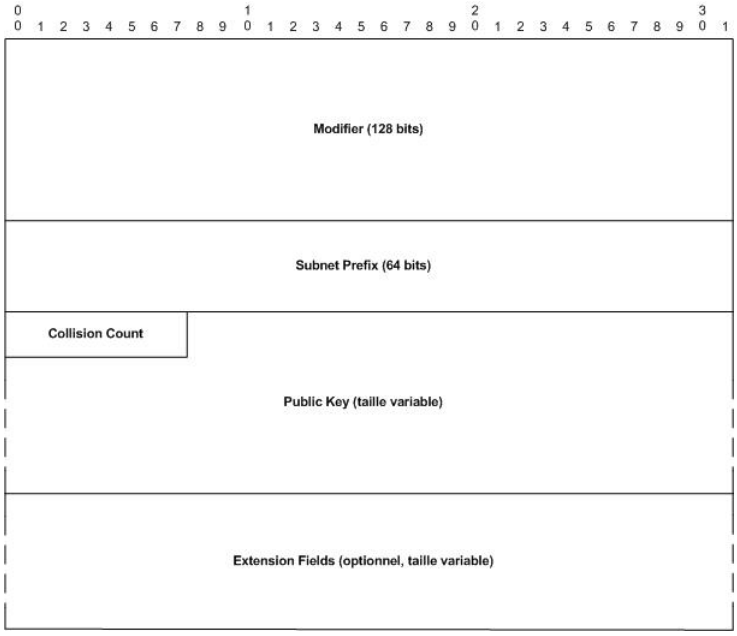

Figure 1.7 – *CGA Parameters*

unique, alors le paramètre *Collision Count* est incrémenté de 1 et on revient à la cinquième étape. Au bout de 3 tentatives infructueuses, le processus de génération CGA s'arrête. Dans le cas où une adresse CGA générée est unique (i.e., la procédure *Duplicate Address Detection* valide l'unicité de l'adresse), celle-ci est assignée et la structure de données *CGA Parameters* associée à cette adresse est sauvegardée. Le processus de génération CGA est résumé dans la Figure 1.8, où `rand(n)` est une fonction retournant aléatoirement une valeur de n bits, `sha1(m)` est une fonction appliquant la fonction de hachage SHA-1 sur le message m, `first(n,m)` est une fonction retournant les n premiers bits du message m, `sec(n,m)` retournant 0 si n est égal à 0 ou si les "n x 16" premiers bits de m sont égaux à 0, `iid_cga(m, p)` est une fonction encodant p dans les 3 premiers bits de m et mettant les 7ème et 8ème bits à 0.

Vérification

La possession d'une adresse CGA est vérifiée en deux temps. Tout d'abord, il est nécessaire de vérifier qu'il est possible de regénérer l'adresse à partir des *CGA Parameters* et de l'adresse CGA. Si cela aboutit correctement, alors il est obligatoire de vérifier que le possesseur de la CGA est bien le propriétaire de la clé privée associée à la clé publique ayant servi à générer l'adresse CGA.

Concernant la regénération de l'adresse CGA, en premier lieu, on vérifie que le paramètre *Collision Count* a une valeur comprise entre 0 et 2. Ensuite, on vérifie que le paramètre *Subnet Prefix* est égal au préfixe réseau de l'adresse CGA. Puis, on applique la fonction de hachage SHA-1 sur la structure *CGA Parameters* et les 64 premiers du résultat sont appelés *Hash1*. Alors, on vérifie que l'IID de l'adresse CGA est identique à *Hash1* en ignorant les 3 premiers bits ainsi que les 7ème et 8ème bits. Puis on récupère les 3 premiers bits de l'IID, que l'on nomme *Sec*. Ensuite, on concatène le paramètre *Modifier* à la valeur `0x000000000000000000`, suivi du paramètre *Public Key*, et finalement, s'il y en a, les données concernant les potentielles extensions du paramètre *Extension Fields*. La

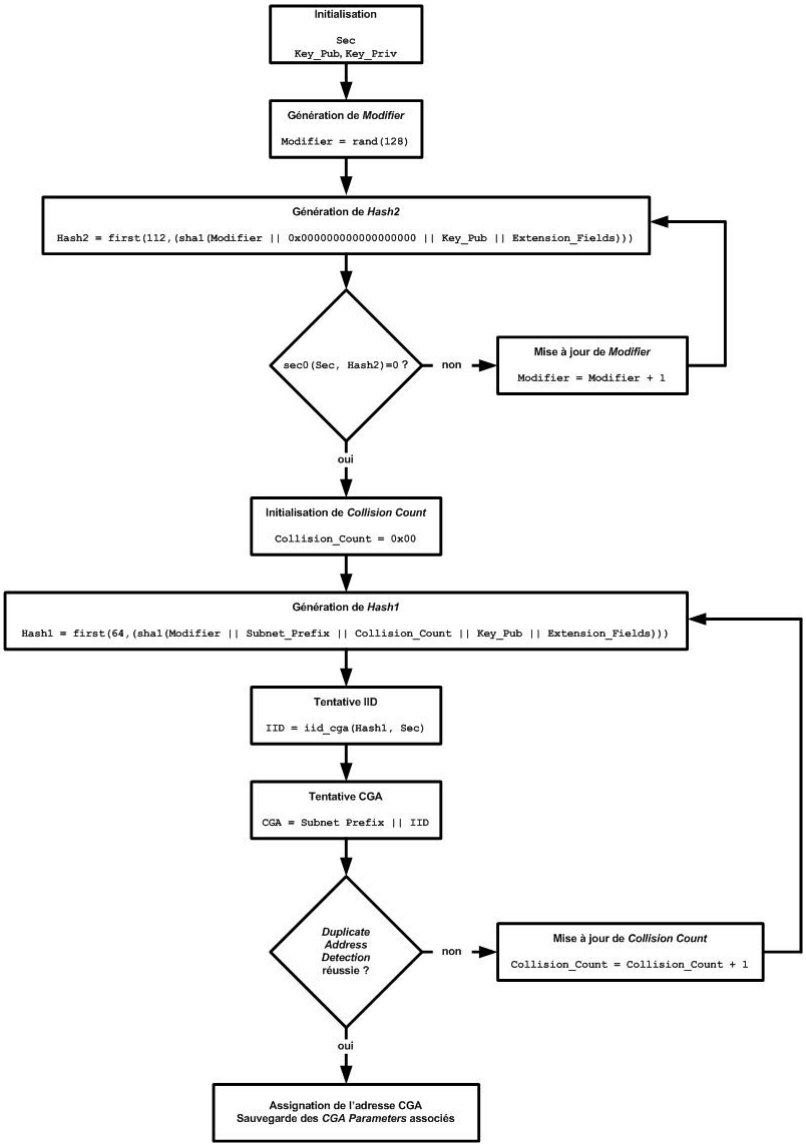

Figure 1.8 – Génération d'adresse CGA

fonction de hachage SHA-1 est appliquée sur cette concaténation. Les 112 premiers bits du résultat sont appelés *Hash2*. Alors, on vérifie que les "*Sec* x 16" premiers bits sont à 0, sauf dans le cas où *Sec* est zéro et dans ce cas, le test est toujours réussi. Le processus de regénération d'adresse CGA est résumé dans la Figure 1.9, où `sha1(m)` est une fonction appliquant la fonction de hachage SHA-1 sur le message m, `first(n,m)` est une fonction retournant les n premiers bits du message m, `sec(n,m)` retournant 0 si n est égal à 0 ou si les "n x 16" premiers bits de m sont égaux à 0.

Concernant la possession de la clé privée liée à la clé publique ayant servi à générer l'adresse CGA, le possesseur de cette adresse devra fournir au vérificateur des données qu'il aura précédemment signées avec la clé privée. Par exemple (cf. 2.3.1), ces données signées sont fournies dans une option ICMPv6 *RSA Signature* dans le cas de *Secure Neighbor Discovery*. Le vérificateur sera capable de vérifier la signature de ces données, et donc la possession de la clé publique associée à l'adresse CGA, grâce à la clé publique associée qui est incluse dans le champ *Public Key* des *CGA Parameters*.

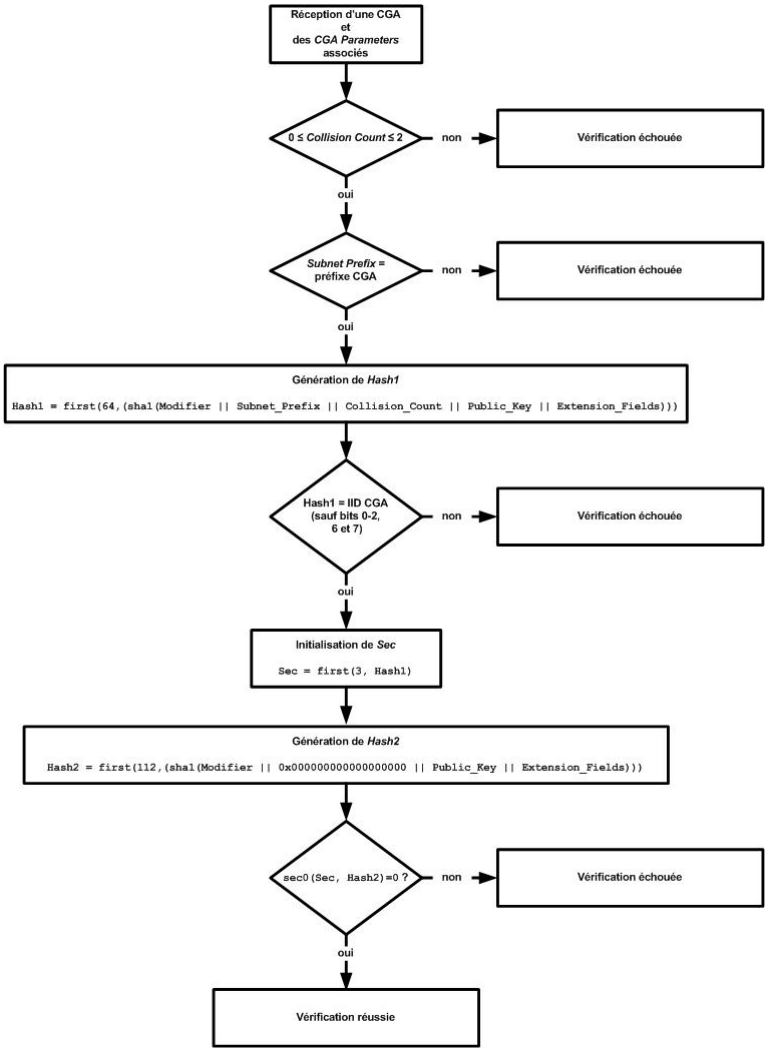

Figure 1.9 – Vérification CGA : regénération de l'adresse

1.2.5 Cryptographiques et locales

Une dernière méthode de génération d'adresses IPv6 reprend les propriétés cryptographiques des adresses CGA, présentées précédemment, mais ces adresses sont non routables en dehors d'un site précis. Cette méthode s'appelle *Overlay Routable Cryptographic Hash Identifiers* (**ORCHID**) [NLD07] et est en cours de révision à l'IETF [NLD11]. Les adresses IPv6 de ce type n'utilisent pas l'identifiant d'interface ni le préfixe réseau pour être générées mais un préfixe spécifique, qui est `2001:10::/28`, complété par une génération cryptographique. Ce type d'adresse sert principalement comme identifiant pour le protocole *Host Identity Protocol* (**HIP**) [MNJH08], en cours de révision [MTJH12], mais rien n'empêche de les utiliser comme des adresses ULA (cf. Section 1.1.1).

Pour générer une adresse ORCHID, tout d'abord, deux données sont nécessaires. La première est le *Context ID*, codé sur 128 bits et qui caractérise l'utilisation faite de l'adresse ORCHID (e.g., pour HIP) ainsi que la fonction de hachage utilisée (e.g., SHA-1 pour HIP) pour la génération de l'adresse. La deuxième est une donnée de longueur variable qui est la clé publique d'une paire de clés publique/privée. Pour générer l'adresse ORCHID, dans un premier temps, la fonction de hachage liée au *Context ID* est appliquée à la concaténation

de ces deux données. Ensuite, les 100 bits se trouvant au milieu du résultat en sortie de la fonction de hachage sont concaténés à la valeur `2001:10::/28`, ce qui forme l'adresse ORCHID.

1.3 Protocole Neighbor Discovery

Le protocole de découverte des voisins, *Neighbor Discovery Protocol* (**NDP**) [NNSS07], en IPv6 est composé de 2 parties. La première composante permet aux nœuds IPv6, sur un même lien réseau, de s'échanger des informations. En particulier, cela permet aux nœuds de connaître les adresses physiques, *Link-Layer Address* (**LLA**), associées aux adresses IPv6 (e.g., adresse MAC), comme le fait en IPv4 le protocole de résolution d'adresse, *Address Resolution Protocol* (**ARP**) [Plu82]. Cette composante est généralement appelée *Neighbor Discovery* (**ND**). La deuxième composante permet d'échanger des informations entre un nœud IPv6 et un routeur IPv6. Cette composante est généralement appelée *Router Discovery* (**RD**). Le mécanisme NDP repose sur l'utilisation de messages de type *Internet Control Message Protocol for IPv6* (**ICMPv6**) [CDG06].

1.3.1 Neighbor Discovery

La composante ND utilise 2 messages ICMPv6 lors des échanges. Le premier, *Neighbor Solicitation* (**NS**), illustré dans la Figure 1.10, permet à un nœud de demander des informations concernant une adresse IPv6 spécifique, contenue dans le champ *Target* du message NS. Ce message peut contenir la LLA de l'émetteur dans l'option *Source link-layer address*. Cette requête est envoyée en multicast sur le lien réseau.

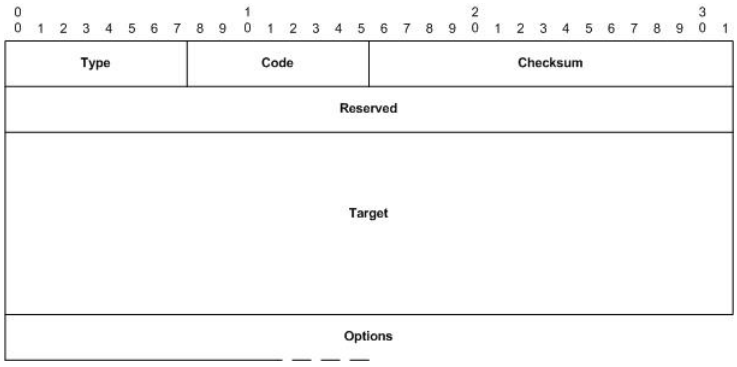

Figure 1.10 – Message NS

L'autre message, *Neighbor Advertisement* (**NA**), illustré dans la Figure 1.11, permet à un nœud de répondre à un message NA, ou alors d'avertir le lien réseau (*unsolicited NA*), concernant des informations pour l'adresse IPv6 contenue dans le champ *Target* du message NA. Ce message peut contenir la LLA de l'émetteur dans l'option *Target link-layer address*.

Les informations concernant les voisins d'un nœud sont stockées dans la base de données appelée *Neighbor Cache*.

Neighbor Unreachability Detection

La procédure *Neighbor Unreachability Detection* (**NUD**) permet à un nœud IPv6 de savoir si un autre nœud d'adresse `@IP` est encore joignable à cette adresse. Pour cela, il

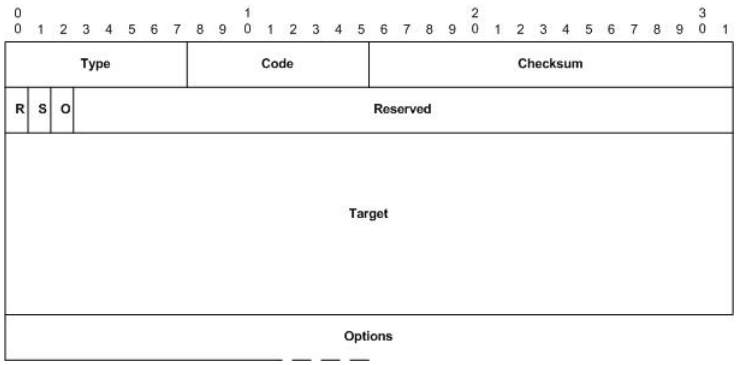

Figure 1.11 – Message NA

envoie un message NS unicasté plusieurs fois à destination de l'adresse `@IP` et s'il ne reçoit aucun message NA unicasté en retour, cela signifie que le nœud en question n'est plus joignable à cette adresse `@IP`.

1.3.2 Router Discovery

La composante RD utilise principalement 2 messages ICMPv6 lors des échanges. Le premier, *Router Solicitation* (**RS**), illustré dans la Figure 1.12, permet à un nœud de demander des informations concernant les routeurs présents sur le lien réseau. Ce message peut contenir la LLA de l'émetteur dans l'option *Source link-layer address*. Cette requête est envoyée en multicast sur le lien réseau (i.e., l'adresse destination est l'adresse `FF02::2/128`).

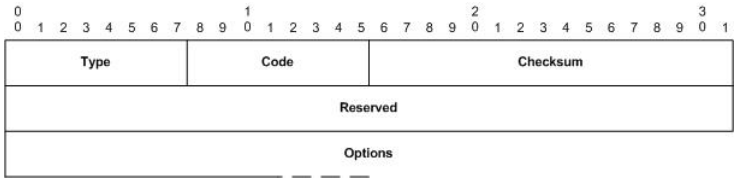

Figure 1.12 – Message RS

L'autre message, *Router Advertisement* (**RA**), illustré dans la Figure 1.13, permet à un routeur de répondre à un message RA, ou alors d'avertir le lien réseau (*unsolicited RA*), pour fournir des informations le concernant. Ce message peut contenir la LLA de l'émetteur dans l'option *Target link-layer address*.

Les messages RA permettent à un nœud de connaitre les routeurs actifs sur un lien et ainsi de pouvoir faire transiter par eux des communications à destination de nœuds ne se trouvant pas sur le même lien réseau. Les routeurs connus sont stockés dans une base de données appelée *Default Router List*.

Un message RA peut contenir les informations concernant un préfixe déployé sur le lien réseau dans l'option *Prefix Information*, illustrée dans la Figure 1.14.

Les informations concernant les préfixes sont stockées dans une base de données appelée *Prefix List*.

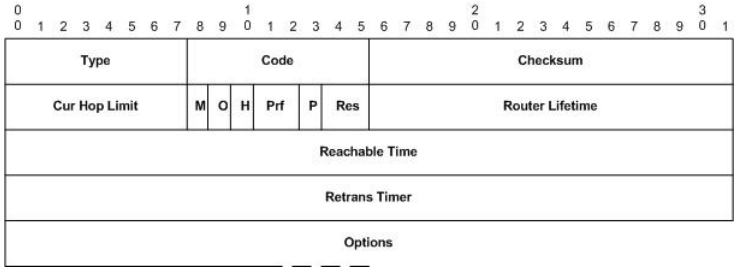

Figure 1.13 – Message RA

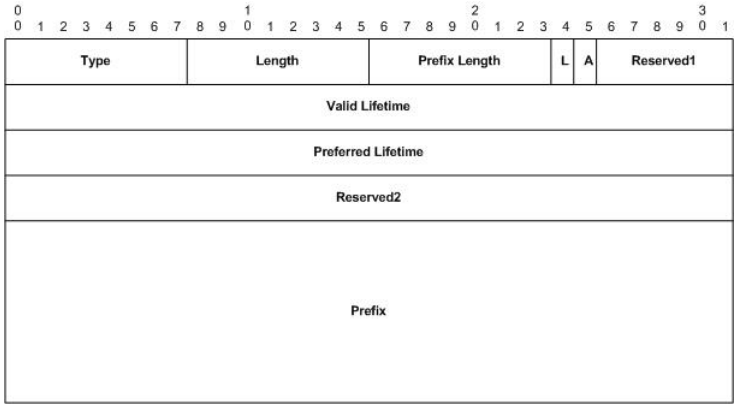

Figure 1.14 – Option Prefix Information

Redirection

Le routeur utilise un message spécifique, appelé *Redirect* et illustré dans la Figure 1.15, pour informer un nœud qu'il existe un meilleur chemin pour communiquer avec un destinataire spécifique, contenu dans le champ *Destination Address*. Cela peut être en passant par un autre routeur ou alors le destinataire est sur le même lien réseau en fait. L'information est incluse dans le champ *Target Address*. Ce message devrait contenir la LLA du nœud par lequel devra transiter les paquets (i.e., meilleur routeur ou destinataire suivant le cas) dans l'option *Target link-layer address*.

Ce type de message met à jour la base de données *Destination Cache* contenant les informations qui permettent à un nœud de savoir si un destinataire est sur le même lien que lui ou si les paquets doivent transiter par un routeur.

1.4 Autoconfiguration des adresses IPv6

Le mécanisme d'autoconfiguration d'adresse en IPv6, *Stateless Address Autoconfiguration* (**SLAAC**) [TNJ07], permet à un nœud de générer et s'assigner une adresse IPv6.

1.4.1 Adresses lien-local et globale

Tout d'abord, chaque nœud IPv6 doit posséder une adresse LUA. Pour cela, il va générer un IID suivant une des méthodes précédemment présentées (cf. Section 1.2). Ensuite, il concatène le préfixe `FE80::/64` à cet IID afin de former l'adresse LUA.

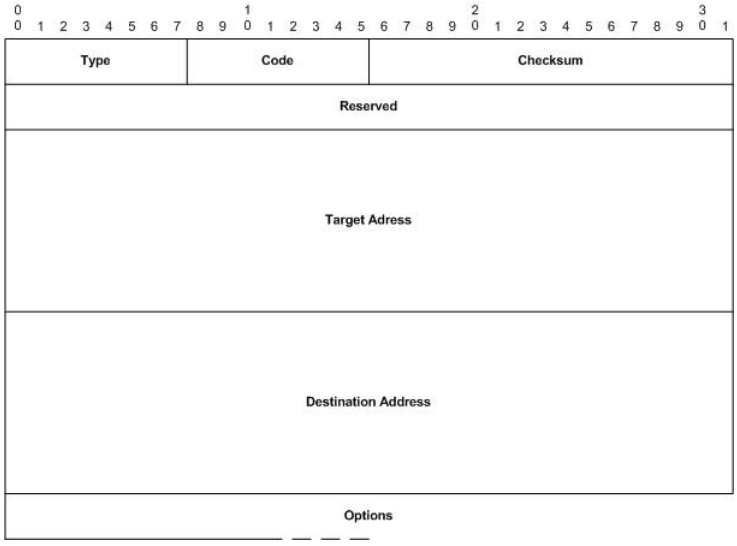

Figure 1.15 – Message Redirect

Pour former une adresse GUA, un nœud a besoin dans un premier temps de connaître le préfixe déployé sur le lien réseau. Pour cela, il doit recevoir un message RA grâce, soit à l'émission d'un message RS, soit à l'envoi régulier de RA par un routeur. Les données concernant le préfixe déployé sont incluses dans l'option *Prefix Information* (cf. Section 1.3.2), en particulier le préfixe réseau, dans le champ *Prefix*, et les durées de vie, dans les champs *Valid Lifetime* et *Preferred Lifetime*. Identiquement à la génération d'une LUA, le nœud va générer un IID auquel il concatène le préfixe reçu dans le message RA.

1.4.2 Duplicate Address Detection

Avant de pouvoir utiliser une adresse, qu'elle soit LUA ou GUA, qu'elle soit produite via SLAAC ou allouée via *Dynamic Host Configuration Protocol for IPv6* (**DHCPv6**) [DBV+03], un nœud IPv6 doit vérifier qu'aucun autre nœud sur le même lien réseau ne se l'utilise déjà. Pour effectuer la vérification, le nœud lance une procédure appelée *Duplicate Address Detection* (**DAD**) [TNJ07]. Pour cela, comme illustré dans la Figure 1.16, (1) le nœud IPv6, désirant utiliser l'adresse `@IP`, demande si un autre nœud possède déjà cette adresse grâce à un message NS multicasté, ayant comme adresse source l'adresse `::0/128`. Si c'est le cas, (2) le nœud utilisant déjà l'adresse `@IP` va l'informer, via un message NA unicasté. Le nœud IPv6, recevant ce message, ne devra pas alors l'utiliser. Il se peut aussi qu'un autre nœud génère dans le même laps de temps la même adresse `@IP` et donc (3) envoie lui aussi un message NS multicasté pour vérifier l'unicité de cette adresse. Dans ce cas, aucun des 2 nœuds ne devra utiliser cette adresse. Si au bout d'un certain temps, le nœud IPv6 n'a reçu ni de message NS ni de message NA concernant cette adresse `@IP`, (4) il pourra l'utiliser.

Optimistic DAD

Dans certains scénarios, comme le cas de la mobilité IPv6 (cf. Section 1.6.1), il peut être nécessaire qu'un nœud IPv6 puisse utiliser rapidement une adresse, générée ou allouée. La procédure *Optimistic Duplicate Address Detection* (**oDAD**) [Moo06] permet d'utiliser une adresse IPv6 avant d'avoir finalisé la procédure DAD.

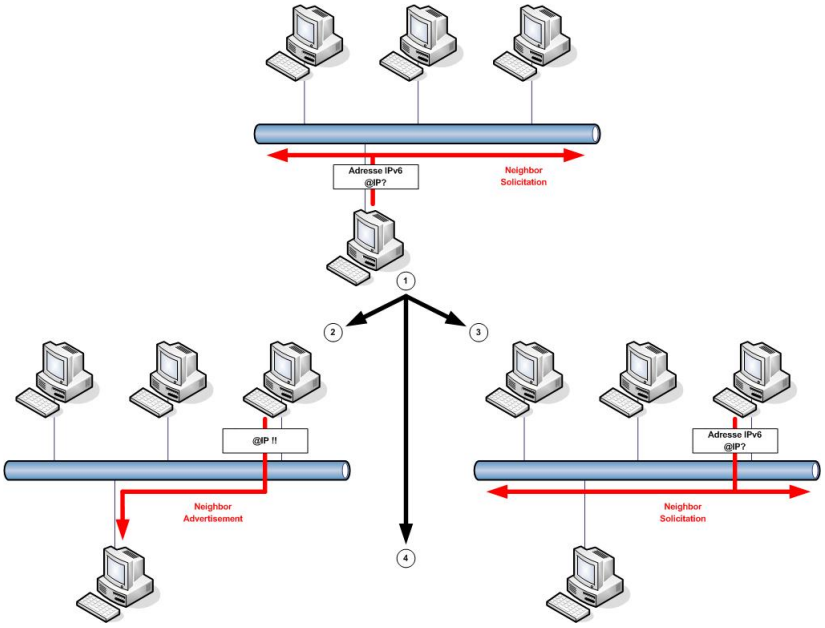

Figure 1.16 – Procédure DAD

1.5 Neighbor Discovery Proxy

Certaines architectures réseau reposent sur l'utilisation de la technologie *MAC Bridge*[3] où les nœuds d'un même réseau IP (i.e., préfixe IP identique) sont situés sur des liens réseau séparés. Afin de permettre le bon fonctionnement de NDP dans ce type d'architecture, le mécanisme *Neighbor Discovery Proxy* (**ND Proxy**) a été spécifié [TTP06]. Il permet de modifier la signalisation NDP : en particulier, les adresses LLA des nœuds IPv6 sont substituées par celles de l'entité ND Proxy dans les messages échangés. Ainsi, comme illustré dans la Figure 1.17, les nœuds A et B ont respectivement les adresses LLA `@a` et `@b`. Le ND Proxy a, comme adresses LLA, les adresses `@p_a` et `@p_b`. Supposons que le nœud A désire communiquer avec le nœud B. Pour cela, il émet un message NS incluant l'adresse IPv6 du nœud B et une option *Source link-layer address* contenant l'adresse `@a`. Le ND Proxy intercepte le message, remplace cette adresse par l'adresse `@p_b` puis transmet le message modifié au nœud B. A la réception, celui-ci sauvegarde dans son *Neighbor Cache* qu'à l'adresse IP du nœud A correspond l'adresse LLA `@p_b`. Puis, le nœud B émet un message NA avec une option *Target link-layer address* contenant l'adresse `@b`. Le ND Proxy intercepte à nouveau le message, remplace cette adresse par l'adresse `@p_a` et transmet le message modifié au nœud A. A la réception, celui-ci sauvegarde dans son *Neighbor Cache* qu'à l'adresse IP du nœud B correspond l'adresse LLA `@p_a`. Du coup, tous les messages envoyés par le nœud A à destination du nœud B seront envoyés, avec comme adresse destination LLA, l'adresse `@p_a` et réciproquement, les messages envoyés par le nœud B à destination du nœud A, seront envoyés, avec comme adresse destination LLA, l'adresse `@p_b`.

3. http://standards.ieee.org/getieee802/download/802.1D-2004.pdf

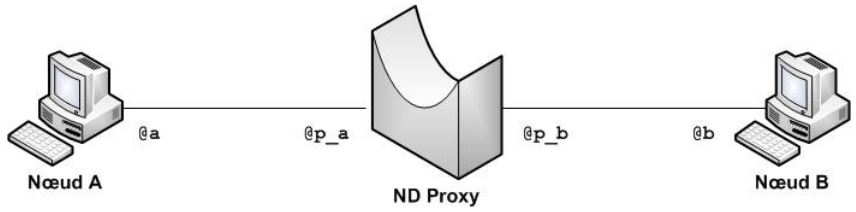

Figure 1.17 – ND Proxy

1.6 Mobilité IPv6

Le protocole de mobilité en IPv6, *Mobility Support in IPv6* (**MIPv6**) [PJA11], permet à un nœud IPv6 de changer de réseau (i.e., préfixes IPv6 différents entre l'ancien et le nouveau réseau) sans que ses connexions ne se coupent suite au changement d'adresses et tout en restant joignable pour n'importe quel correspondant.

1.6.1 MIPv6

Un nœud MIPv6, appelé *Mobile Node* (**MN**), possède deux types d'adresses. La première est la *Home Address* (**HoA**), adresse principale habituellement enregistrée dans les DNS, qui est liée à un réseau que l'on appelle *Home Network*. La deuxième est une adresse temporaire, la *Care-of Address* (**CoA**), liée à des réseaux autres que son *Home Network* et que le MN visite : cette famille de réseaux est appelée *Foreign Network*. Une entité, appelée *Home Agent* (**HA**), située sur le *Home Network* est en charge de la gestion de la mobilité du MN.

Lorsque le MN se trouve dans son *Home Network*, tout se passe comme s'il était un nœud IPv6 normal. Par contre, lorsque le MN se déplace dans un *Foreign Network*, il obtient une CoA locale au *Foreign Network*, généralement grâce au mécanisme SLAAC (cf. Section 1.4), et en informe son HA via un message de signalisation, nommé *Binding Update* (**BU**). Le HA confirme alors qu'il a bien reçu l'information grâce à un acquittement, nommé *Binding Acknowledgment* (**BA**).

Dans le cas où un nœud IPv6, appelé *Correspondent Node* (**CN**), désire communiquer avec le MN, les paquets IP qu'il envoie à destination du MN et adressés à la HoA, sont interceptés par le HA puis encapsulés dans un tunnel vers le MN à destination de la CoA. De même, en retour, le MN encapsule ses paquets IP dans le même tunnel vers le HA qui les transmet au CN. C'est ce que l'on appelle le *Reverse Tunneling*, qui est illustré dans la Figure 1.18.

Il existe une optimisation de routage, *Route Optimization* (**RO**), qui permet d'éviter que tout le trafic entre le MN et le CN ne circule via le HA. Pour cela, le MN envoie un BU contenant sa CoA au CN. Celui-ci confirme qu'il a bien l'information grâce à un BA. Ensuite, le CN envoie directement ses paquets IP au MN en y incluant une extension d'en-tête IPv6 de type *Routing Header, type 2* (**RH2**) contenant la HoA du MN. Ensuite le MN lui envoie directement ses paquets IP en y incluant une extension d'en-tête IPv6 de type *Home Address* contenant sa HoA.

1.6.2 Sécurité de MIPv6

La signalisation de MIPv6 est critique pour le bon fonctionnement de la mobilité mais surtout elle permet de monter des attaques facilement.

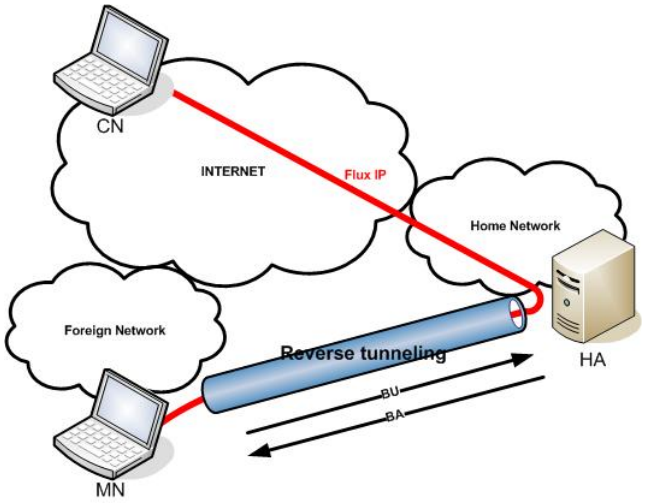

Figure 1.18 – MIPv6 - Reverse Tunneling

Entre le MN et le HA

La sécurisation de la signalisation entre le MN et le HA repose sur l'utilisation d'IPsec [KS05] (cf. Section 4.1.1) afin de protéger les BU et les BA entre le MN et le HA. Pour cela, le protocole ESP [Ken05b] en mode transport est utilisé pour authentifier les données émises : l'association de sécurité IPsec, *Security Association* (**SA**), garantit que l'émetteur est bien le propriétaire de la HoA et ESP couvre l'intégrité de la CoA qui est placée dans une option de mobilité de type *Alternate Care-of Address*. Cela permet d'éviter ainsi à l'émetteur du BU de se faire passer pour un autre MN en utilisant une fausse HoA mais aussi à un nœud se situant entre le MN et le HA de modifier la CoA en interceptant le BU.

Entre le MN et le CN

La méthode standard pour sécuriser la signalisation de la RO est plus complexe car elle part du principe qu'elle ne doit pas reposer sur une infrastructure de confiance, comme une *Public Key Infrastructure* (**PKI**) par exemple. Elle repose sur un mécanisme appelé *Return Routability* et qui est effectué avant que le MN n'envoie un BU au CN. Tout d'abord, le MN envoie un message, appelé *Home Test Init* (**HoTI**), dans le reverse tunneling et qui contient un cookie (*Home Init Cookie*), la HoA et l'adresse IP du CN. En parallèle, il envoie un autre message appelé *Care-of Test Init* (**CoTI**), directement au CN et qui contient un autre cookie (*Care-of Init Cookie*), la CoA et l'adresse IP du CN. À la réception de ces deux messages, le CN génère une clé secrète *Kcn*, un nonce, puis il renvoie un message, appelé *Home Test* (**HoT**), à destination de la HoA et qui contient le *Home Init Cookie*, l'index d'un nonce et le *Home Keygen Token*. Ce dernier est généré avec *Kcn*, le nonce précédemment cité et la HoA. Parallèlement, il envoie un message, appelé *Care-of Test* (**CoT**), à destination de la CoA et qui contient le *Care-of Init Cookie*, l'index d'un nonce et le *Care-of Keygen Token*. Ce dernier est généré avec *Kcn*, le nonce précédemment cité et la CoA. En recevant ces deux messages, le MN génère alors, à partir du *Home Keygen Token* et du *Care-of Keygen Token*, une clé appelée *Kbm*, qui sera utilisée pour sécuriser les données envoyées dans le BU. Grâce aux quatre messages qui empruntent deux chemins différents, le mécanisme *Return Routability* garantit au CN qu'un MN se trouve bien à

l'adresse CoA. De plus, il apporte la preuve de l'identité du MN (i.e. MN a bien l'adresse HoA) au CN, et ce, grâce au HA qui se porte en quelque sorte garant quant à l'authenticité des messages HoTI et HoT échangés. Aussi, il est nécessaire de protéger les messages HoTI et HoT entre le MN et le HA avec IPsec. Le mécanisme *Return Routability* est illustré dans la Figure 1.19.

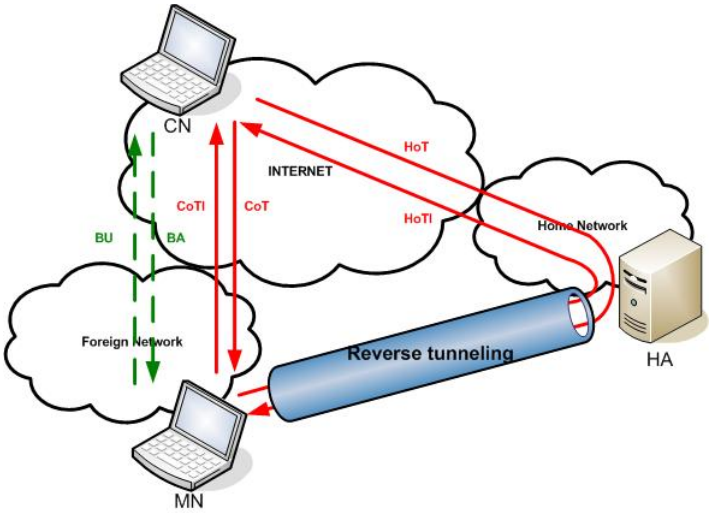

Figure 1.19 – MIPv6 - Mécanisme Return Routability

Il est à noter qu'un HA ou CN doit directement rejeter tout paquet (sauf les messages *Return Routability* dans le cas d'un CN) venant directement d'un MN s'il n'y a pas eu de BU de sa part auparavant afin d'éviter de créer des attaques de type *Bombing* ou *par rebond*. Ce genre d'attaques consiste à envoyer des BU contenant de fausses informations : la HoA ou la CoA envoyée dans le BU est en fait l'adresse IP de la victime, ce qui entraîne une attaque de type *Denial of Service* (**DoS**).

1.7 Synthèse du chapitre

Dans ce chapitre, nous avons tout d'abord rappelé l'adressage en IPv6, que cela soit les différentes classes d'adresses IPv6 (cf. Section 1.1), que les différents types d'adresses IPv6 (cf. Section 1.2). En particulier, nous avons décris les adresses CGA (cf. Section 1.2.4), identifiants cryptographiques qui nous serviront de base pour nos travaux.

Ensuite, nous avons présenté le mécanisme NDP (cf. Section 1.3), ainsi le mécanisme ND Proxy qui est son extension (cf. Section 1.5), cœur du protocole IPv6. En effet, il sert aux nœuds IPv6 à communiquer entre eux, à l'instar du mécanisme ARP en IPv4. Mais surtout, il supporte de nouvelles fonctionnalités, tel que le mécanisme d'autoconfiguration IPv6, SLAAC (cf. Section 1.4), qui permet à un nœud de s'assigner une adresse IPv6, sans qu'il soit nécessaire de mettre en place des entités dédiées, comme DHCP en IPv4. Aussi, il est nécessaire de sécuriser de manière forte le mécanisme NDP car il est critique pour IPv6. C'est ce que nous verrons au Chapitre 2. Par ailleurs, comme nous le verrons dans le Chapitre 3, une solution protégeant contre l'usurpation d'adresse IPv6 source repose sur ce mécanisme.

Finalement, nous avons rappelé le mécanisme MIPv6 (cf. Section 1.6) qui permet la mobilité de nœuds IPv6 ainsi que sa sécurisation. Celui-ci utilise le mécanisme NDP, ce qui aura des conséquences comme nous le verrons dans le Chapitre 2. De plus, par la suite, le

mécanisme MIPv6 nous servira d'exemple concret d'utilisation d'une de nos contributions dans le Chapitre 4.

Chapitre 2

Analyse de la sécurité du mécanisme Neighbor Discovery

Comme nous l'avons vu dans le Chapitre 1, le mécanisme NDP est le cœur du protocole IPv6. Il est nécessaire dès qu'un nœud IPv6 désire s'attribuer une adresse. Il permet à un nœud IPv6 de communiquer avec d'autres nœuds IPv6, y compris des routeurs. Aussi la sécurité de ce mécanisme est cruciale pour IPv6.

Dans ce chapitre, nous analysons tout d'abord les failles de sécurité du mécanisme NDP. Ensuite, nous présentons des solutions palliatives limitant ces dernières. Après quoi, nous décrivons le mécanisme *Secure Neighbor Discovery* (**SEND**) qui est la solution standardisée à l'IETF de sécurisation du mécanisme NDP. Finalement, nous analysons les limites d'une telle solution, dont certaines sont le fruit de nos travaux : attaque par rejeu dans SEND, inefficacité de SEND pour la sécurisation d'un service en mobilité IPv6, inemployabilité de SEND dans certains scénarios.

2.1 Vulnérabilités du mécanisme Neighbor Discovery

Il existe 2 catégories de vulnérabilités pour le mécanisme NDP. La première catégorie concerne les attaques externes au lien réseau où se situent les potentielles victimes. La deuxième catégorie s'appuie, elle, sur des attaques internes au lien réseau.

2.1.1 Attaques externes

Lors de la réception d'un paquet IPv6 à destination d'un nœud de ce sous-réseau par un routeur, suivant les spécifications de NDP, ce routeur doit vérifier s'il a déjà dans son *Neighbor Cache* les informations concernant le destinataire de ce paquet IPv6 afin de pouvoir le lui transmettre. Si ce n'est pas le cas, il stocke le paquet IPv6 et applique la procédure ND (cf. Section 1.3.1). Tout d'abord, il crée une nouvelle entrée dans le *Neighbor Cache* puis il envoie un message NS pour obtenir un message NA de la part du nœud destinataire contenant ces informations. D'un autre côté, en IPv6, un sous-réseau (i.e., pour un préfixe réseau donné) peut contenir jusqu'à 2^{64} adresses, représentant ainsi des milliards et des milliards d'adresses. Mais la grande majorité d'entre elles ne sont pas assignées. C'est en particulier le cas pour les architectures de type *Point-to-Point* entre 2 routeurs, où sur les 2^{64} adresses, seules 2 adresses sont assignées (i.e., une adresse par routeur). Du coup [GJK12], un attaquant, se trouvant à l'extérieur du sous-réseau et scannant celui-ci (i.e., envoi d'un grand nombre de requêtes dont chacune est à destination d'une adresse différente de ce sous-réseau), peut initier au routeur un grand

nombre de procédures ND. Cela peut résulter en une saturation du *Neighbor Cache*, vu que pour chaque procédure ND, une entrée est créée. Ceci pourra alors entraîner le rejet d'acheminement de paquets IPv6 légitimes car nécessitant eux aussi une procédure ND et donc la création d'une entrée dans le *Neighbor Cache* qui est déjà plein.

2.1.2 Attaques internes

Les potentielles attaques internes à un sous-réseau se basent principalement sur la manipulation de la signalisation NDP [NKN04]. On peut classer ces attaques en 2 catégories. La première concerne les attaques liées au routeur, et donc les procédures RD et Redirection. La deuxième catégorie concerne les attaques liées aux nœuds, et donc la procédure ND ainsi que la procédure NUD.

Les attaques, décrites par la suite, ont été implémentées avec scapy6 [1], qui est un dérivé pour IPv6 du bien connu logiciel scapy [2], dans le cadre du projet ANR MobiSEND [ECC+08] [ECBM08]. Ces attaques, et donc les implémentations, fonctionnent contre tout système d'exploitation conforme aux standards IPv6 concernant les mécanismes NDP [NNSS07] et SLAAC [TNJ07].

Partie routeur

Les principales attaques connues concernant le routeur sont principalement basées sur la falsification de messages RA (cf. Section 1.3.2).

Malicious Last Hop Router L'objectif de cette attaque est de permettre à un nœud malveillant de se faire passer pour un routeur et que d'autres nœuds sur le même lien réseau décident de le sélectionner comme routeur par défaut : toutes leurs communications vers l'extérieur passeront alors par ce routeur. Ainsi, il pourra, soit faire un *Denial of Service* (**DoS**) en rejetant ensuite toutes les communications de ces nœuds victimes de l'attaque, soit faire un *Man in the Middle* (**MitM**) interceptant toutes les données non chiffrées, comme illustré dans la Figure 2.1.

Pour réaliser cette attaque, l'attaquant va envoyer des messages RA falsifiés sur le lien réseau, conformes aux caractéristiques du sous-réseau (e.g., même préfixe réseau), soit en réponse à des messages RS de nœuds sur le lien, soit à intervalle régulier.

Default Router is killed L'objectif de cette attaque est de permettre à un nœud malveillant de faire croire à des nœuds sur le lien qu'un routeur légitime n'assure plus sa fonction de routeur. Si la *Default Router List* de ces nœds se retrouve vide, alors ils ne pourront plus envoyer de paquets IP vers l'extérieur, comme illustré dans la Figure 2.2. Cette attaque peut être combinée avec la précédente attaque décrite (i.e., *Malicious Last Hop Router*) pour augmenter les chances de réussite de cette dernière.

Pour réaliser cette attaque, l'attaquant va envoyer des messages RA usurpés, identiques à ceux envoyés par le routeur légitime ciblé mais avec le champ *Router Lifetime* fixé à la valeur 0. Cette valeur, lorsqu'elle est à 0, indique que le routeur émettant le message RA ne doit pas être considéré comme un routeur par défaut, et donc ne doit pas être dans la *Default Router List*.

1. http ://natisbad.org/scapy/
2. https ://secdev.org/projects/scapy/

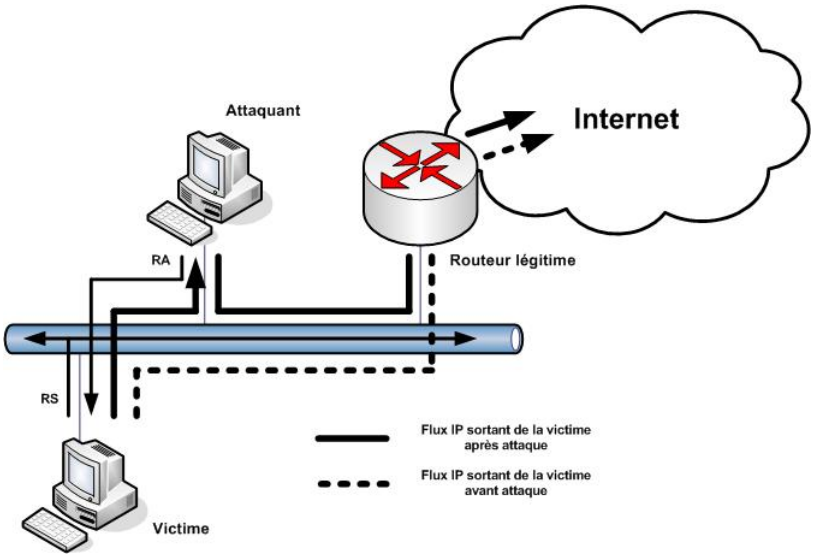

Figure 2.1 – Attaque *Malicious Last Hop Router*

Spoofed Redirect Message L'objectif de cette attaque est de permettre à un nœud malveillant de faire croire à sa victime qu'il existe un meilleur chemin, qui est erroné en fait, pour envoyer des paquets à destination d'un nœud IPv6 spécifique. Ainsi, comme illustré dans la Figure 2.3, si le meilleur chemin était d'envoyer les paquets via un routeur légitime et que l'attaquant, se faisant passer pour ce routeur, indique que le destinataire se trouve en fait sur le même lien que la victime, cette dernière ne pourra plus faire parvenir les paquets à son destinataire.

Pour réaliser cette attaque, l'attaquant va envoyer des messages *Redirect* (cf. Section 1.3.2), usurpant le routeur légitime, en indiquant une adresse IPv6 erronée dans le champ *Target Address*.

Bogus On-Link Prefix L'objectif de cette attaque est de permettre à un nœud malveillant de faire croire à sa victime qu'un préfixe IPv6 spécifique est employé sur le lien réseau. Ainsi, comme illustré dans la Figure 2.4, les communications de la victime à destination d'un nœud ayant une adresse IPv6 incluse dans ce préfixe ne passeront plus par un routeur et seront envoyées directement sur le lien.

Pour réaliser cette attaque, l'attaquant va envoyer à la victime des messages RA falsifiés, incluant une option *Prefix Information* dont le champ *Prefix* contient le préfixe IPv6 erroné.

Bogus Address Configuration Prefix L'objectif de cette attaque est similaire à la précédente car l'attaquant veut faire croire à sa victime qu'un préfixe spécifique est employé sur le lien réseau, mais de plus, que celle-ci doit autoconfigurer une adresse à partir de celui-ci (cf. Section 1.4). Du coup, comme illustré dans la Figure 2.4, non seulement les communications de la victime à destination d'un nœud ayant une adresse IPv6 incluse dans ce préfixe ne passeront plus par un routeur et seront envoyées directement sur le lien, mais la victime n'aura pas de réponses de l'extérieur concernant les autres communications (i.e., les protocoles de routage n'achemineront pas les paquets en réponse), comme illustré dans la Figure 2.5.

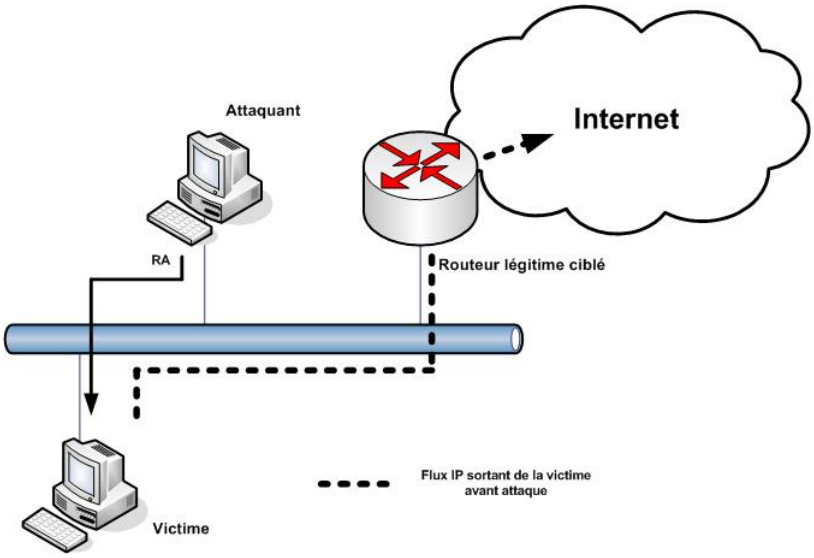

Figure 2.2 – Attaque *Default Router is killed*

Pour réaliser cette attaque, l'attaquant va envoyer à la victime des messages RA falsifiés, dont le flag M est fixé à la valeur 0 (i.e., indiquant que le nœud n'a d'autres choix que la procédure SLAAC), en incluant une option *Prefix Information* dont le champ *Prefix* contient le préfixe IPv6 erroné.

Parameter spoofing L'objectif de cette attaque est de permettre à un nœud malveillant de propager des informations erronées concernant les caractéristiques du lien réseau où se situe la victime. Cela entraînera des perturbations concernant les communications de cette dernière. La liste des conséquences est longue mais par exemple, l'attaquant peut faire croire à sa victime qu'elle doit obligatoirement passer par un serveur DHCPv6 [DBV+03] pour récupérer une adresse unicast globale.

Pour réaliser cette attaque, l'attaquant va envoyer des messages RA usurpés, identiques à ceux envoyés par le routeur légitime mais avec des informations erronées dans un ou plusieurs champs. Ainsi, s'il veut faire croire que la victime doit obtenir son adresse unicast globale via un serveur DHCPv6, il mettra le flag M du message RA à la valeur 1.

Partie ND

Les principales attaques connues concernant les nœuds sont principalement basées sur la falsification de messages NS et NA (cf. Section 1.3.1).

NA/NS Spoofing L'objectif de cette attaque est de permettre à un nœud malveillant de faire croire à sa victime qu'une adresse IPv6 est associée à une LLA spécifique (i.e., adresse MAC), qui est en fait erronée. Suivant la LLA falsifiée que l'attaquant fournit, les conséquences sont différentes. Ainsi, si personne ne la possède, les communications de la victime n'aboutiront pas (i.e., attaque DoS). Sinon, ces communications parviendront au nœud propriétaire de cette LLA. Dans ce dernier cas, si c'est l'attaquant, il pourra poursuivre par une attaque MitM, comme illustré dans la Figure 2.6. Dans le cas contraire, cela pourrait se traduire par une attaque DoS visant le destinataire des communications si celles-ci sont nombreuses.

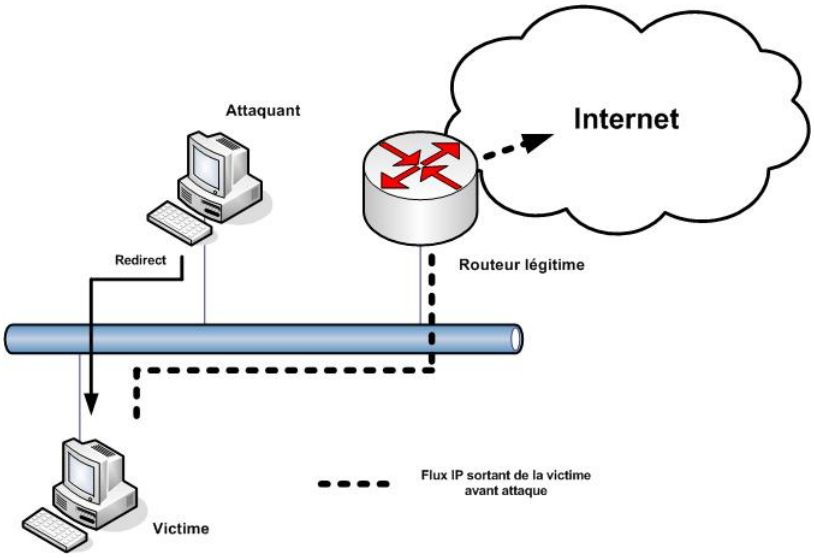

Figure 2.3 – Attaque *Spoofed Redirect Message*

Pour réaliser cette attaque, l'attaquant va envoyer des messages NA usurpés, identiques à ceux envoyés par le destinataire légitime, incluant une option *Target link-layer* qui contiendra la LLA falsifiée.

NUD failure L'objectif de cette attaque est de permettre à un nœud malveillant de faire croire à sa victime que le destinataire de ses communications est encore actif lors d'une procédure NUD (cf. Section 1.3.1). Ceci entraîne une attaque DoS car la victime continuera d'émettre des paquets en consommant des ressources pour le faire.

Pour réaliser cette attaque, l'attaquant va répondre aux messages NS envoyés par la victime avec des messages NA usurpés, identiques à ceux qu'aurait envoyé le destinataire des communications.

DAD DoS Attack L'objectif de cette attaque est de permettre à un nœud malveillant de faire croire à sa victime que l'adresse faisant l'objet d'une procédure DAD n'est pas unique. Cela a pour conséquence d'empêcher la victime de s'attribuer une adresse et est une attaque DoS.

Afin de réaliser cette attaque, l'attaquant va répondre aux messages NS envoyés par la victime, soit avec des messages NS identiques à ceux de la victime, soit avec des messages NA avec l'adresse désirée par la victime dans le champ *Target*. En reprenant la Figure 1.16, ceci correspond aux cas (2) et (3).

2.2 Solutions de protection palliatives

Afin de limiter les attaques précédemment décrites, il existe plusieurs solutions, plus ou moins efficaces.

2.2.1 Protection du *Neighbor Cache*

Une première solution [Kum12] concerne les attaques externes (cf. Section 2.1.1) et la protection du *Neighbor Cache*. Elle spécifie en particulier une politique de gestion des

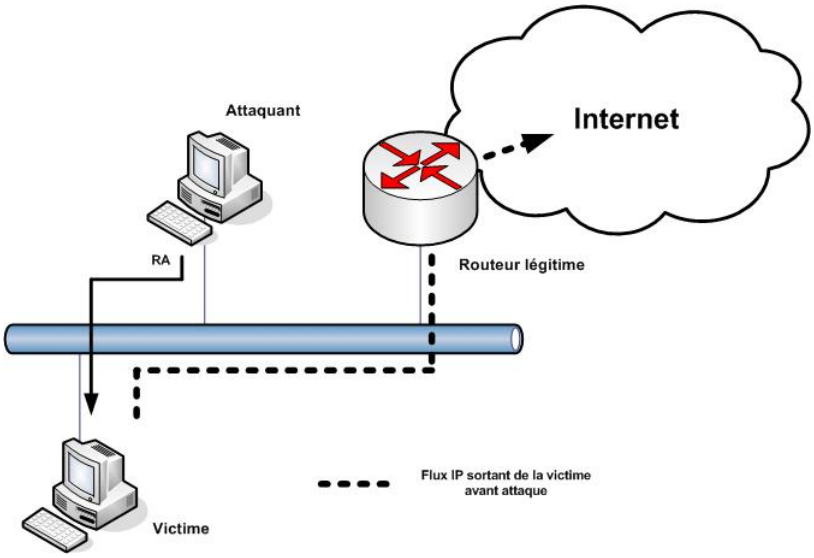

Figure 2.4 – Attaque *Bogus On-Link Prefix*

messages NDP et du *Neighbor Cache* afin d'éviter que ce dernier se retrouve saturé trop rapidement.

2.2.2 Préfixe en /127

Dans le cas particulier d'architectures de type *Point-to-Point* entre 2 routeurs, une méthode [KNB+11] simple est préconisée. Normalement toute adresse unicast doit être composée obligatoirement de 2 parties de 64 bits (cf. Section 1.1.1), la première étant le préfixe réseau et la deuxième étant l'IID. Ici, la solution consiste à déroger à cette règle et à fixer le préfixe sur 127 bits. Il ne reste plus du coup que deux adresses pouvant être assignées pour ce préfixe : une par routeur.

2.2.3 Mécanisme *RA Guard*

Cette solution [LAdVPM11] tente de limiter les attaques internes basées sur l'utilisation de messages RA (cf. Section 2.1.2). Elle repose sur l'utilisation d'une entité de couche 2 dont les ports sont configurés pour filtrer les messages RA. Comme illustré dans la Figure 2.7, suivant la configuration, soit le port les laisse passer, soit il les rejette.

La configuration peut se faire manuellement ou dynamiquement. Dans ce dernier cas, pendant un certain temps, l'entité *RA Guard* apprend où se trouvent les routeurs en se basant sur les messages RA reçus. Malheureusement, cela peut entraîner une mauvaise configuration si un attaquant réussit à lancer ses attaques durant ce temps d'apprentissage. Aussi, c'est la configuration manuelle qui sera privilégiée. Or, celle-ci peut devenir fastidieuse si un grand nombre de ports sont à configurer.

2.2.4 Mécanisme *Neighbor Discovery Shield*

Reprenant le même principe que le mécanisme *RA Guard*, une proposition [Gon12b] récente à l'IETF, *Neighbor Discovery Shield*, propose d'utiliser une entité de couche 2 dont les ports filtrent les messages NDP, exceptés les messages RA. La configuration est

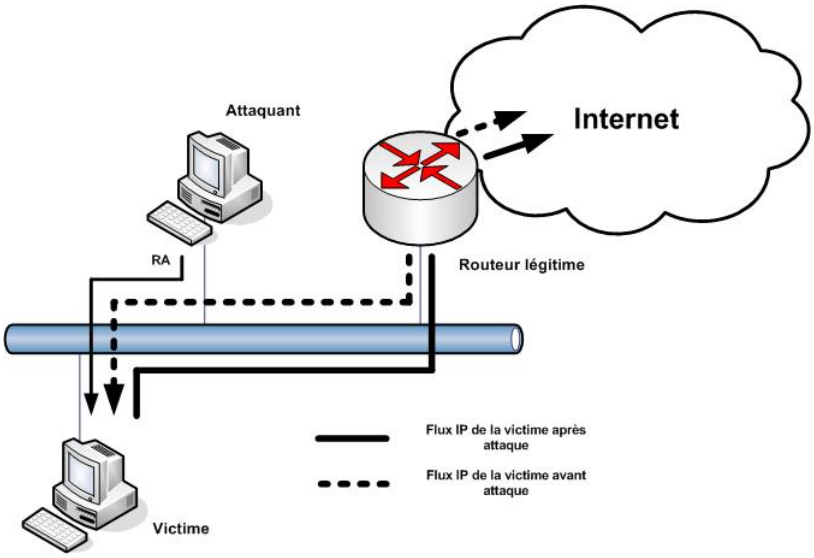

Figure 2.5 – Attaque *Bogus Address Configuration Prefix*

dynamique, mais comme précédemment, elle peut être corrompue si l'attaquant parvient à émettre des messages durant la phase d'apprentissage.

2.3 Secure Neighbor Discovery

Le mécanisme *Secure Neighbor Discovery* (**SEND**) [AKZN05] est la solution standardisée à l'IETF permettant de contrer les attaques internes (cf. Section 2.1.2). Il est composé de deux parties, une partie sécurisant les échanges entre 2 nœuds IPv6, l'autre partie sécurisant la découverte d'un routeur.

2.3.1 Sécurisation des échanges entre 2 nœuds

Cette composante de SEND permet de sécuriser les messages NDP échangés entre 2 nœuds IPv6. Elle repose sur l'utilisation des adresses CGA (cf. Section 1.2.4). Le mécanisme SEND définit 4 nouvelles options ICMPv6 pour sécuriser les échanges NDP :

- option *CGA*
  Cette option permet de transporter les CGA Parameters durant les échanges NDP. Le format de cette option est illustré dans la Figure 2.8.
- option *RSA Signature*
  Cette option permet de transporter la signature RSA [RSA78] d'un message NDP, ainsi qu'une partie du résultat de la fonction de hachage SHA-1 appliquée à la clé publique associée à la clé privée ayant servi pour la signature. Le format de cette option est illustré dans la Figure 2.9.
- option *Timestamp*
  Cette option permet de transporter la valeur temporelle indiquant le nombre de secondes écoulées depuis la date du 1er Janvier 1970 à 00 :00 (UTC) lorsque le message contenant cette option a été émis. Elle permet de contrer les attaques par rejeu. Le format de cette option est illustré dans la Figure 2.10.
- option *Nonce*

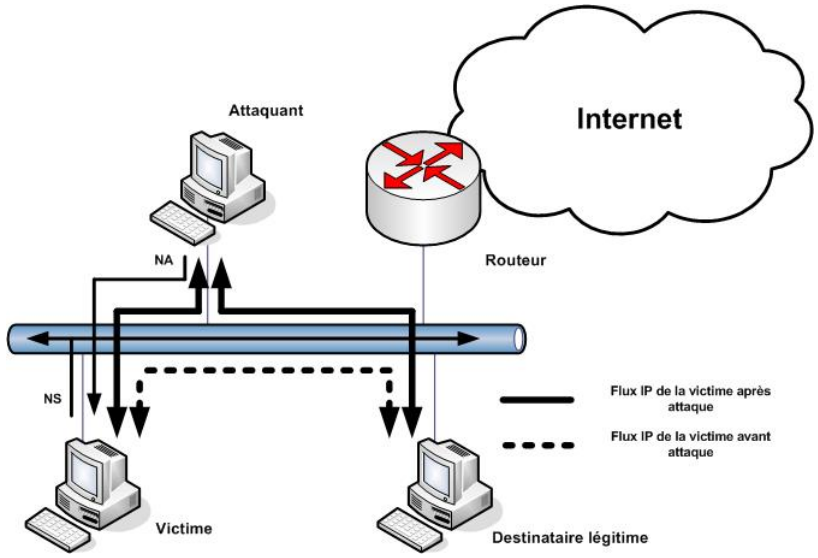

Figure 2.6 – Attaque *NA/NS Spoofing*

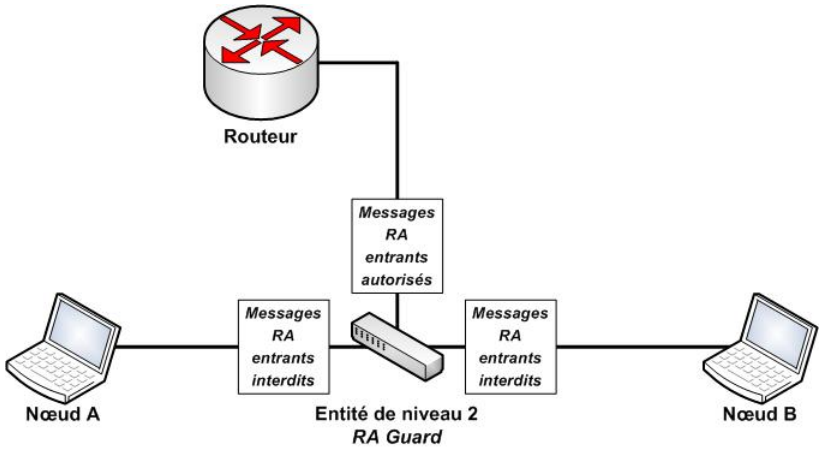

Figure 2.7 – Architecture *RA Guard*

Cette option permet de transporter une valeur, initialement tirée aléatoirement, qui permet d'associer une réponse NDP à une requête NDP antérieure. Le format de cette option est illustré dans la Figure 2.11.

Quand un nœud utilise SEND pour sécuriser un message NDP, les options ICMPv6 précédemment décrites sont incluses dans ce message initial. Ensuite le message est signé en utilisant la clé privée associée à la clé publique utilisée pour la génération de l'adresse CGA. Par contre, avec les messages RA, la signature est générée avec la clé privée associée au certificat prouvant que le nœud est autorisé à être routeur (cf. Section 2.3.2). Le message NDP ainsi signé est illustré dans la Figure 2.12.

2.3.2 Sécurisation de la découverte d'un routeur

Cette composante de SEND permet de sécuriser la découverte d'un routeur par un nœud IPv6 et est appelé *Authorization Delegation Discovery* (**ADD**). Ce mécanisme per-

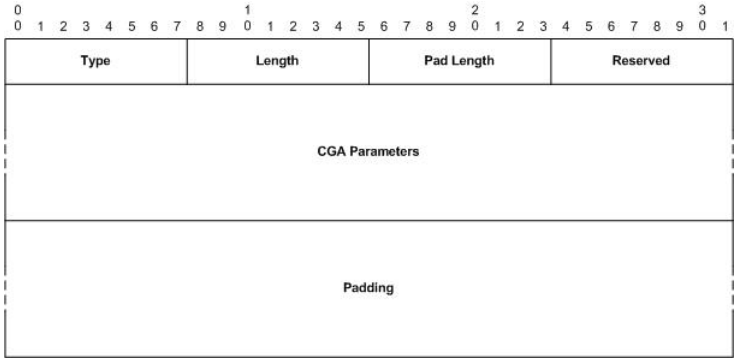

Figure 2.8 – Format de l'option *CGA*

Figure 2.9 – Format de l'option *RSA Signature*

met au nœud de vérifier si un autre nœud est bien habilité à être routeur et en particulier s'il a le droit d'annoncer un préfixe IPv6 spécifique. Ce mécanisme est bâti sur l'utilisation de certificats X.509 [GKK12] indiquant qu'un nœud est bien un routeur et incluant, grâce à l'extension IP, le ou les préfixes IPv6 qu'il peut avertir. Ces certificats peuvent reposer sur une architecture de confiance appelée *Resource Public Key Infrastructure* (**RPKI**) [LK12] et servant aussi à la sécurité du protocole de routage BGP. Comme le nœud doit pouvoir vérifier la validité du certificat aussi (i.e., chemin de certification), 2 nouveaux messages ICMPv6 et 2 nouvelles options ICMPv6 ont été définis :

- message *Certification Path Solicitation* (**CPS**)
  Ce message permet à un nœud de demander au routeur un chemin de certification jusqu'à un point de confiance, *Trust Anchor*, dont il a foi. Le format de ce message est illustré dans la Figure 2.13.
- message *Certification Path Advertisement* (**CPA**)
  Ce message permet à un routeur de fournir à un nœud le chemin de certification demandé. Le format de ce message est illustré dans la Figure 2.14.
- option *Trust Anchor*
  Cette option permet d'indiquer un point de confiance (e.g., une autorité de certification) pour un chemin de certification. Le format de cette option est illustré dans la Figure 2.15.

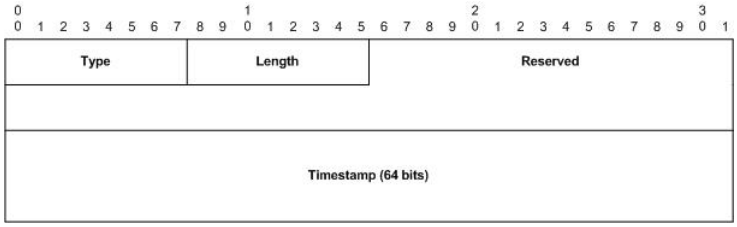

Figure 2.10 – Format de l'option *Timestamp*

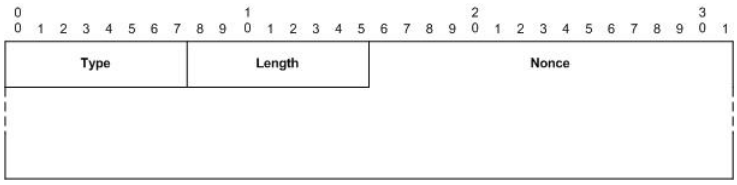

Figure 2.11 – Format de l'option *Nonce*

– option *Certificate*
  Cette option permet de transporter un certificat servant à vérifier un chemin de certification. Le format de cette option est illustré dans la Figure 2.16.

Lors de la réception d'un message RA, qui doit être signé, le nœud va demander à l'émetteur du message de prouver qu'il est bien habilité à être routeur. Pour cela, le nœud envoie un message CPS contenant son ou ses points de confiance grâce à l'option *Trust Anchor*. En retour, le routeur lui fournit un chemin de certification avec un ou plusieurs messages CPA incluant les certificats nécessaires à la vérification grâce à l'option *Certificate*.

2.4 Limitations de SEND

Le mécanisme SEND possède plusieurs limitations, tout d'abord au niveau des algorithmes cryptographiques employés. Ensuite, ce mécanisme n'est pas mature d'un point de vue standardisation et comporte du coup certaines lacunes dans ses spécifications.

2.4.1 Limitations cryptographiques

Les spécifications du mécanisme SEND ne permettent d'utiliser uniquement que 2 algorithmes cryptographiques : SHA-1 [iost02] et RSA [RSA78]. Tout d'abord, cette contrainte va à l'encontre de certaines réglementations nationales ou politiques au sein d'entreprises. De plus, le niveau de sécurité du mécanisme SEND s'en trouve affaibli. Enfin, cela ralentit le déploiement du mécanisme SEND dans certains environnements.

Aspects réglementaires

Certains pays ou sociétés réglementent l'utilisation des algorithmes cryptographiques. C'est le cas par exemple en Russie, où l'algorithme GOST R 34.11-94 [Dol10] doit être obligatoirement utilisé comme fonction de hachage. Aussi, les spécifications de DNSSEC ont été modifiées [DCU10] afin d'autoriser l'utilisation d'un tel algorithme. En l'état actuel, le mécanisme SEND ne pourrait être utilisé en Russie à moins d'une modification de ses spécifications autorisant des algorithmes alternatifs, comme GOST.

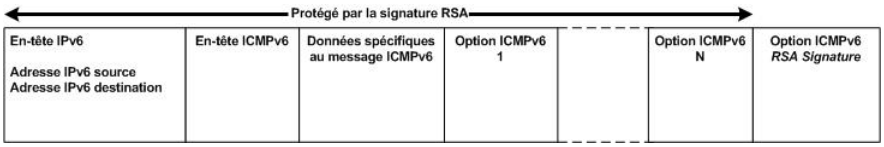

Figure 2.12 – Message NDP sécurisé avec SEND

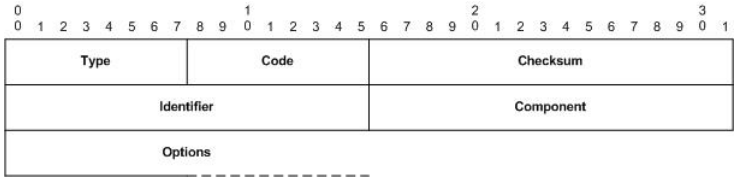

Figure 2.13 – Format du message CPS

Niveau de sécurité de SHA-1

Les dernières cryptanalyses rendues publiques concernant SHA-1 indiquent que cette fonction de hachage risque de ne plus être sûre dans un proche futur car il sera aisé de générer des collisions. Cela impactera l'utilisation des adresses CGA : pour une adresse CGA donnée, et donc une paire de clés publique/privée RSA spécifique, grâce à une collision, il sera possible de générer la même adresse CGA avec une paire de clés publique/privée RSA différente.

Aussi, il est nécessaire que les mécanismes utilisant les adresses CGA, comme le mécanisme SEND, tiennent parfaitement compte de la clé publique ayant servi à la génération d'une adresse CGA, qui est fournie dans les *CGA Parameters*. En effet, en plus de trouver une collision, l'attaquant devra utiliser la même paire de clés RSA que sa victime et donc avoir réussi à la casser pour usurper l'adresse CGA.

Maintenant, l'alternative la plus sûre, pour éviter des collisions, est de permettre l'emploi de la prochaine génération de fonction de hachage dans les spécifications des adresses CGA et du mécanisme SEND : SHA-3 [3].

Coût de l'emploi de RSA

L'algorithme à clé publique RSA est connu pour être coûteux aux niveaux ressources et temps de calcul (i.e., contraintes au niveau du processeur et de la consommation énergétique). Ainsi, cet algorithme n'est pas réellement adapté pour certains nœuds IPv6.

Tout d'abord, cela est le cas pour les nœuds à faibles capacités, comme les terminaux mobiles ou les capteurs. Malheureusement, les spécifications actuelles des adresses CGA n'autorisent pas l'usage d'algorithmes alternatifs comme ceux basés sur les courbes elliptiques, *Elliptic Curve Cryptography* (**ECC**), malgré la publication d'articles [CBL10] ou de propositions de standards à l'IETF [SXZ11].

C'est aussi le cas avec les routeurs, dont la tâche prioritaire est de router le plus rapidement les paquets : les ressources doivent être utilisées le moins possible pour d'autres tâches (e.g., effectuer des calculs cryptographiques lourds). Nous avons proposé à l'IETF une méthode [XKH+08] permettant d'utiliser la cryptographie symétrique, à la place de celle asymétrique, entre un nœud IPv6 et un routeur pour le mécanisme SEND. Dans cette méthode, le nœud IPv6 réclame au routeur une clé secrète partagée grâce à un message

3. http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf

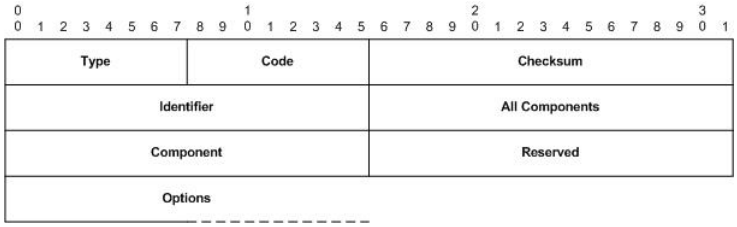

Figure 2.14 – Format du message CPA

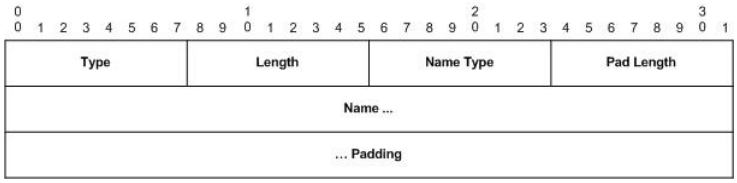

Figure 2.15 – Format de l'option *Trust Anchor*

RS sécurisé avec le mécanisme SEND (i.e., le message RS est signé avec la clé publique associée à l'adresse CGA du nœud). Le routeur lui répond avec un message RA sécurisé avec le mécanisme SEND et incluant la clé secrète partagée qui est chiffrée avec la clé publique associée à l'adresse CGA du nœud. Par la suite, tous les échanges NDP entre ce nœud IPv6 et le routeur seront sécurisés grâce à cette clé secrète partagée.

2.4.2 Faiblesses dans les spécifications

Le mécanisme SEND souffre aussi de faiblesses dans ses spécifications pouvant engendrer des failles de sécurité ou alors empêcher son utilisation.

Attaque par rejeu

La première faiblesse du mécanisme SEND concerne une faille de sécurité, que nous avons publiée [CC08], autorisant une attaque par rejeu lors d'une procédure DAD (cf. Section 1.4.2) même si elle est sécurisée grâce à ce mécanisme.

Pour rappel, lors d'une telle procédure, un nœud IPv6 s'informe avec un message NS si l'adresse IPv6 qu'il va s'assigner n'est pas déjà utilisée par un autre nœud. Dans le cas où il recevrait un message NS comportant la même adresse IPv6, et indiquant qu'un autre nœud applique une procédure DAD pour la même adresse IPv6, aucun de ces deux nœuds ne pourrait l'utiliser. Avec le mécanisme SEND, le nœud peut regénérer une adresse CGA en incrémentant le compteur *Collision Count* (cf. Section 1.2.4) et retenter la procédure DAD pour cette nouvelle adresse. Le nœud IPv6 utilisant le mécanisme SEND peut effectuer jusqu'à 3 tentatives de génération d'adresse CGA, alors qu'avec le mécanisme SLAAC, le nœud IPv6 a seulement 1 tentative de génération d'adresse IPv6. En employant le mécanisme SEND, les messages échangés durant la procédure DAD sont sécurisés et en particulier grâce à l'option *TimeStamp* qui permet de contrer le rejeu de paquets. En effet, pour qu'un paquet provenant d'un nouveau correspondant (i.e., pas d'entrée dans le *Neighbor Cache* concernant ce nœud IPv6) soit valide, il est nécessaire que la différence entre le temps indiqué par l'horloge interne lors de la réception de ce paquet et la valeur contenue dans l'option *Timestamp* de ce même paquet soit comprise dans une fenêtre de temps paramétrable. Par défaut, cette fenêtre est de 10 minutes. Ainsi, un attaquant fera échouer une procédure DAD sécurisée par le mécanisme SEND s'il arrive à rejouer durant

Figure 2.16 – Format de l'option *Certificate*

la fenêtre de temps les paquets NS envoyés par sa victime : la victime pense qu'un autre nœud a généré la même adresse CGA (i.e., les paquets NS sont correctement signés) et essaie aussi de se l'assigner.

Il est possible de détecter ce genre d'attaque. Lors des 3 tentatives de procédure DAD, la valeur de l'option *Nonce* ainsi que les *CGA Parameters* reçus dans les paquets NS seront tout le temps identiques à ceux émis par la victime : la probabilité que cela arrive est normalement négligeable. Donc, une des solutions à cette attaque est d'assigner l'adresse CGA obtenue à la troisième tentative.

Une autre solution serait d'utiliser l'option *Nonce* : durant la procédure DAD, ne tenir compte que des messages NS reçus dont la valeur de l'option *Nonce* est différente de celle dans l'option *Nonce* des messages NS émis.

Sécurisation du message RA

Une autre faiblesse du mécanisme SEND concerne la taille des clés RSA utilisées. Les spécifications CGA définissent que cette taille doit être au minimum de 384 bits. Par ailleurs, avec le mécanisme SEND, un routeur doit signer les messages RA émis avec la clé privée associée à son certificat (i.e., clé publique) prouvant qu'il est bien autorisé à être routeur. Si un routeur utilise une adresse CGA aussi, la paire de clés publique/privée ayant servi à la génération de cette adresse devra être la même que pour le certificat car il n'y a qu'une option *RSA Signature* dans un message sécurisé avec le mécanisme SEND.

Du coup, dans ce dernier cas, se pose le problème du choix de la taille des clés RSA employées par un routeur. Si celle-ci est trop petite, la génération des adresses CGA du routeur ainsi que la signature des messages émis seront rapides. Mais, il sera plus facile de compromettre les clés RSA et du coup le certificat du routeur : l'attaquant sera capable de se faire passer pour le routeur, avec les conséquences décrites auparavant (cf. Section 2.1.2). D'un autre côté, si la taille des clés est trop grande, la génération des adresses CGA et la signature des messages émis occuperont trop les ressources du routeur.

Une solution serait de modifier l'option *RSA Signature* afin d'indiquer si la signature est basée sur une adresse CGA ou sur un certificat. Il faudrait alors autoriser l'inclusion de deux options RSA Signature pour un routeur utilisant une adresse CGA : une contiendrait la signature basée sur l'adresse CGA et l'autre contiendrait la signature basée sur le certificat. Ceci permettrait d'avoir deux paires de clés RSA : une "courte" pour l'adresse CGA et une "robuste" pour le certificat. Les messages NDP, autres que les messages RA, ne seraient signés qu'avec la clé privée associée à l'adresse CGA, réduisant ainsi le temps de calcul pour le routeur.

Usurpation de privilèges dans la mobilité IPv6

Le mécanisme SEND permet à un routeur de prouver qu'il est bien habilité à l'être et qu'il a le droit d'avertir certains préfixes IPv6. Cela signifie que seules les informations concernant ces 2 points sont couvertes par la sécurité fournie par le mécanisme SEND.

Le mécanisme *Dynamic Home Agent Address Discovery* (**DHAAD**) est employé avec MIPv6 (cf. Section 1.6.1) pour permettre à un MN de découvrir dynamiquement un HA pour un préfixe IPv6 donné. Chaque HA, sur le sous-réseau ayant ce préfixe IPv6, a une base de données, la *Home Agents List*, contenant les informations concernant chaque HA dans ce sous-réseau. Cette base de données est mise à jour grâce aux messages RA envoyés par chaque HA, qui incluent une option nommée *Home Agent Information*.

Comme expliqué dans un document que nous avons soumis à l'IETF [DCLM08], le mécanisme SEND ne couvre pas les informations concernant un HA. Ainsi, sur un sous-réseau, n'importe quel routeur "légitime" peut se faire passer pour un HA et corrompre la *Home Agents List* de chaque HA légitime sur ce même sous-réseau. Le problème est encore plus critique avec le mécanisme de mobilité IPv6 *Network Mobility Basic Support Protocol* (**NEMO**) [DWPT05]. En effet, dans un tel contexte, la majorité des nœuds sur le sous-réseau où se trouvent les HA, sont des routeurs mobiles et donc des routeurs légitimes pouvant polluer la *Home Agents List* de chaque HA légitime.

Utilisation d'une même adresse par plusieurs nœuds

Une adresse CGA générée par un nœud IPv6 permet à celui-ci de prouver qu'il en est bien le possesseur. Cette propriété cryptographique est la base du mécanisme SEND. Or, comme nous l'avons décrit dans une *Request for Comments* (**RFC**) de l'IETF [CKD10], il existe des scénarios où plusieurs nœuds doivent utiliser une même adresse IPv6.

Le premier scénario concerne MIPv6 où le HA, afin d'intercepter les paquets à destination du MN quand celui-ci ne se trouve pas dans son *Home Network*, usurpe la HoA du MN en falsifiant les messages ND (cf. Section 1.3.1) : il annonce qu'à la HoA du MN correspond sa LLA.

Le deuxième scénario est ressemblant au premier et concerne IPsec et IKEv2. Lors d'une connexion sécurisée IPsec à un réseau distant, un nœud IPv6 se voit attribuer une adresse IPv6 interne à ce réseau par la passerelle IPsec le protégeant grâce à IKEv2 [ELM10]. Afin de pouvoir tunneliser les paquets provenant de ce réseau à destination du nœud IPv6, la passerelle doit se faire passer pour ce nœud. Une des techniques est de faire comme dans le cas précédent, à savoir, falsifier les messages ND.

Le troisième scénario est lors de l'utilisation du mécanisme ND Proxy (cf. Section 1.5). L'entité intégrant ce mécanisme doit falsifier les messages NDP comme expliqué précédemment.

Le dernier scénario concerne l'usage par plusieurs entités (i.e., deux ou plus) d'une même adresse : c'est avec les adresses de type *anycast* (cf. Section 1.1.3) ou alors avec le mécanisme de mobilité IPv6 *Proxy Mobile IPv6* (**PMIPv6**) [GLD+08] où les routeurs d'accès font croire aux nœuds mobiles IPv6 qu'ils n'ont pas changé de routeur.

Dans tous ces scénarios, le mécanisme SEND ne peut être utilisé en l'état car il faudrait que les différentes entités partageant une même adresse CGA, partagent aussi le matériel de sécurité associé (i.e., paire de clés publique/privée et *CGA Parameters*). Or d'un point de vue sécurité, cela n'est pas une bonne solution : il suffit qu'une des entités soit compromise pour que la sécurité apportée par le mécanisme SEND, utilisés par toutes les autres entités, soit mise en péril.

A la suite de notre information, le groupe de travail *CGA & SEND maIntenance* (**CSI**) [4] a été créé à l'IETF afin de standardiser une solution pour ces scénarios.

4. http://tools.ietf.org/wg/csi/charters

Cas des réseaux privés

Il existe une autre situation où le même matériel de sécurité pour SEND doit être employé par plusieurs entités distinctes mais cette fois-ci, cela concerne les certificats X.509 prouvant qu'un routeur est habilité à avertir des préfixes IPv6 donnés.

Le scénario dans lequel cela peut se produire est lorsque ces préfixes sont de type ULA (cf. Section 1.1.1). En effet, n'importe qui peut les utiliser dans son réseau et donc il peut y avoir plusieurs réseaux les employant. Dans ce cas là, si le mécanisme SEND est déployé, il faudrait que les routeurs dans chacun des réseaux en question soit en possession d'un certificat l'autorisant à avertir le préfixe `FC00::/7`. Du coup, comment serait-il possible de différentier, d'un point de vue sécurité, deux réseaux différents utilisant ce même préfixe ?

Dans le cadre de nos travaux, nous avons défini les spécifications d'une solution basée sur ORCHID (cf. Section 1.2.5) permettant de répondre à cette problématique et qui a été brevetée [eDM11] [CM12]. Dans cette solution, les routeurs génèrent leur propre préfixe d'une manière cryptographique comme avec ORCHID. Ensuite, ils génèrent des certificats auto-signés avec le matériel de sécurité ORCHID, fournissant l'unicité de leur préfixe "privé", qui seront utilisés dans le cadre du mécanisme SEND. Ce dernier doit être modifié afin de prendre en compte ce nouveau type de certificat.

Faute de temps, nous n'avons pas pu terminer les travaux sur cette problématique et avoir une spécification finalisée de notre solution.

2.5 Synthèse du chapitre

Dans ce chapitre, nous avons tout d'abord présenté les failles de sécurité concernant le mécanisme NDP (cf. Section 2.1), qui rendent l'utilisation du protocole IPv6 impossible. Dans le cadre du projet ANR MobiSEND, nous avons développé des scripts en Python les reproduisant.

Ensuite, nous avons présenté des solutions palliatives à ces failles (cf. Section 2.2). Elles permettent de les limiter mais ne sont pas toujours fiables et peuvent être compliquées à mettre en place. Par la suite, nous avons décrit le mécanisme SEND (cf. Section 2.3),qui est la solution de sécurisation standardisée à l'IETF et le principal mécanisme connu utilisant des identifiants cryptographiques, les adresses CGA.

Finalement, nous avons décrit différentes limites au mécanisme SEND ainsi qu'aux adresses CGA (cf. Section 2.4). Soit, elles sont dépendantes de la contrainte des algorithmes cryptographiques employés. Soit, elles sont liées à l'immaturité des spécifications du mécanismes SEND et des adresses CGA. Ainsi, comme nous l'avons montré dans une de nos contributions (cf. Section 2.4.2), il est encore possible de mettre en place une attaque par rejeu contre le mécanisme SLAAC, malgré la protection fournie par le mécanisme SEND. Autre exemple d'immaturité des spécifications, comme nous l'avons détaillé dans une RFC à l'IETF (cf. Section 2.4.2), il est impossible d'utiliser le mécanisme SEND dans de nombreux scénarios. Notre contribution a résulté à la création du groupe de travail *CGA & SEND maIntenance* (**CSI**) à l'IETF dont l'objectif est de spécifier la solution comblant cette lacune majeure. Pareillement, nous avons découvert un autre scénario où le mécanisme SEND ne peut être employé mais qui concerne cette fois-ci l'utilisation par plusieurs réseaux du même préfixe ULA, et donc d'un même certificat X.509 (cf. Section 2.4.2). Nous avons spécifié, et breveté, une solution pour y remédier mais, faute de temps, nous n'avons pu la finaliser. Enfin, une autre de nos contributions (cf. Section 2.4.2), que nous avons soumise à l'IETF, démontre les limites de la protection du mécanisme SEND dans un contexte de mobilité IPv6. Pour la majorité de ces limites, nous avons proposé

des améliorations permettant de les combler.

Ainsi, en supposant qu'elles sont appliquées, on peut considérer les adresses CGA, et donc le mécanisme SEND, comme fiables au niveau sécurité. A partir de ce constat, comme nous le verrons dans la suite de nos travaux, des solutions de sécurisation IPv6 peuvent se baser sur la réutilisation du mécanisme SEND, comme décrit dans le Chapitre 3, ou des adresses CGA, comme décrit dans les Chapitres 4 et 5.

Chapitre 3

Protection contre l'usurpation d'adresse IP basée sur SAVI

L'adresse IP dans l'Internet sert à la fois à identifier et à localiser un nœud IP. Ainsi, afin de ne pas pouvoir remonter à la source d'une attaque, des nœuds IP malveillants peuvent falsifier leur adresse IP source, en usurpant soit une adresse appartenant à un autre nœud IP, soit une adresse non attribuée. Afin de lutter contre cette falsification, l'*Internet Engineering Task Force* (**IETF**) a standardisé et encouragé le déploiement de la technique appelée "Network Ingress Filtering", dont la déclinaison la plus connue est *unicast Reverse Path Forwarding* (**uRPF**). Malheureusement, cette technique a certaines limites dont la principale est sa précision. Aussi, l'IETF a lancé des travaux, au sein du groupe de travail *Source Address Validation Improvements* (**SAVI**), pour standardiser des mécanismes complémentaires permettant une meilleure lutte contre l'usurpation d'adresses IP source. Ceux-ci reposent sur l'observation de la signalisation d'assignation d'adresses, telles que les mécanismes NDP ou SEND qui ont été présentés respectivement dans les Sections 1.3 et 2.3.

Dans ce chapitre, tout d'abord, nous rappelons les diverses attaques pouvant s'appuyer sur l'usurpation d'adresse IP source. Ensuite, nous présentons la technique de "Network Ingress Filtering" et expliquons ses limites. Ensuite, nous décrivons alors les différentes solutions standardisées dans le groupe de travail SAVI, sous notre responsabilité, à l'IETF. Après cela, nous décrivons notre proposition d'intégration de SAVI, dans le cas concret du réseau d'accès IPv6 ADSL/Fibre, à l'IETF et nous donnons les implémentations/déploiements existants à ce jour. Enfin, nous concluons avec les limites propres à SAVI et les potentiels futurs travaux sur le sujet.

3.1 Menaces liées à l'usurpation d'adresses IP source

Les attaques capables d'utiliser l'usurpation d'adresses IP source peuvent être classifiées selon leurs conséquences [MBH11]. Tout d'abord, il y a celles dont l'objectif est de corrompre des bases de données, connues sous le terme technique de *Poisoning*, afin généralement de pouvoir enchaîner avec une autre attaque. Ensuite, celles dont l'objectif est de bloquer l'accès à un service en le mettant à terre, connues sous le nom de *Denial of Service* (**DoS**). Enfin, celles dont le but est la reconnaissance et l'infiltration dans les systèmes.

3.1.1 Poisoning

L'objectif d'attaques de type *Poisoning* est d'introduire, dans une base de données, des informations erronées. Celles-ci pourront alors servir pour la mise en place d'un autre type d'attaque comme, par exemple, celle du type *Man in the Middle* (**MitM**) permettant de dérouter le trafic vers l'attaquant et ainsi d'obtenir, et modifier s'il le désire, les informations échangées entre 2 nœuds IP. Voici quelques unes des attaques de ce type les plus connues :

- *ARP Poisoning*
  L'*Address Resolution Protocol* (**ARP**) est un mécanisme [Plu82] qui permet à un nœud IPv4 de connaître l'adresse de niveau 2 (e.g., adresse MAC) associée à une adresse IPv4 attribuée à un autre nœud IPv4. La réponse à une requête ARP est stockée dans une table ARP dans le nœud IPv4. Un attaquant accomplit un *ARP Poisoning*, soit en envoyant à la victime des messages de type *Unsolicited ARP*, soit en répondant à des message de type *ARP Request* émis par la victime. Les messages envoyés par l'attaquant contiennent des informations erronées (e.g., adresse MAC de l'attaquant) corrompant la table ARP de la victime, ce qui peut entraîner par la suite une attaque de type MitM ou DoS.
- *NDP Poisoning*
  Le mécanisme *Neighbor Discovery Protocol* (**NDP**) [NNSS07] a globalement la même fonction qu'ARP mais pour le protocole IPv6. Comme pour ARP, l'attaquant envoie des messages NDP contenant des informations erronées pour corrompre les bases de données NDP (e.g., *Neighbor Cache*, *Default Router List*), aidant par la suite pour une attaque de type MitM ou DoS. La base de données NDP ainsi modifiée dépend du type de message NDP utilisé [NKN04] lors de l'attaque, comme nous l'avons déjà présenté dans la Section 2.1.
- Table de routage
  Le rôle d'un message *ICMP Redirect*, envoyé par un routeur, est d'avertir le nœud IP le recevant que le meilleur chemin pour une communication vers un nœud spécifique est via un autre routeur. Un attaquant peut envoyer un message *ICMP Redirect* [NKN04], contenant des informations erronées, pour corrompre la table de routage de la victime. Nous avons présenté plus précisément cette attaque dans la Section 2.1.
- *DNS Cache Poisoning*
  Le *Domain Name System* (**DNS**) [Moc87a] permet d'associer un nom de machine (e.g., `www.example.org`) à une adresse IP. Afin de réduire les requêtes DNS, la technique de *DNS Cache* est déployée au sein d'entreprises et de fournisseurs d'accès Internet. L'attaque *DNS Cache Poisoning* est une attaque contre l'infrastructure DNS [AA04] où l'attaquant pousse des informations erronées au sein d'un *DNS Cache*.

3.1.2 Denial of Service

L'objectif d'une attaque de type DoS est de perturber, voire d'interrompre, un service fourni. Généralement, ce genre d'attaque s'appuie sur l'envoie massif de messages sur l'entité fournissant le service, technique communément appelée *Flooding*. Voici quelques unes des attaques de ce type les plus connues :

- LAND
  L'attaque *Local Area Network Denial* (**LAND**) est basée sur l'envoi d'un paquet IP dont l'adresse IP source et destination sont l'adresse IP de la victime. A la réception

de celui-ci, la victime va continuellement s'envoyer des paquets IP. Ceci a pour conséquence une utilisation croissante des ressources de la victime (e.g., le paquet falsifié est un paquet TCP SYN), pouvant même entraîner son "crash" [1].

– *UDP Flooding*
L'attaque *UDP Flooding*[2] est basée sur l'envoi massif par l'attaquant de datagrammes UDP, dont l'adresse IP source est falsifiée et les ports UDP sont aléatoires, à destination de sa victime. Cette dernière répond alors avec des messages ICMP de type *Destination Unreachable* entraînant une consommation importante de ses ressources.
– *TCP SYN Flooding*
L'attaque *TCP SYN Flooding* [Edd07] est basée sur l'envoi massif par l'attaquant de messages TCP SYN, dont l'adresse IP source est falsifiée, à destination de sa victime. Ceci entraîne une allocation importante de ses ressources afin de maintenir les demandes de connexion provenant de l'attaquant et peut résulter à des refus de demandes de connexion légitimes provenant d'autres nœuds IP.
– *ICMP Flooding*
De nombreuses attaques DoS basées sur ICMP existent. Une des plus connues est celle nommée *Smurf attack*[3] et qui consiste à l'envoi massif par l'attaquant de messages ICMP de type *Echo Request*, dont l'adresse IP destination est une adresse de type broadcast ou multicast et dont l'adresse IP source est falsifiée et correspond à l'adresse IP de la victime. Ainsi, tous les nœuds IP recevant ce type de message ICMP répondent un message ICMP de type *Echo Reply* à destination de la victime qui se retrouve submergée de messages.

3.1.3 Reconnaissance et infiltration

Afin de préparer ses attaques, un attaquant peut scanner de potentielles vulnérabilités sur une cible spécifique. En utilisant des adresses IP source falsifiées, l'attaquant peut ainsi cacher sa localisation. Par exemple, l'application `nmap`[4], un outil bien connu de reconnaissance réseau, dispose d'une telle fonction.

De la même manière, les vers et *malwares* pourraient cacher la localisation des terminaux déjà infectés durant leur propagation. Par exemple, même si le ver nommé *SQL Slammer*[5], basé sur UDP, ne falsifie pas son adresse IP source, rien n'empêche des codes dérivés de ce ver d'ajouter une telle fonctionnalité.

3.2 Network Ingress Filtering

Suite à l'augmentation[6] d'attaques DoS basées sur l'usurpation d'adresse IP source, l'IETF a standardisé la technique de *Network Ingress Filtering* et a encouragé son déploiement. Plusieurs variantes de cette technique existent : soit statique, connue sous le terme *BCP 38* [FS00], soit dynamique, connue sous le terme *BCP 84* [BS04] ou aussi *unicast Reverse Path Forwarding* (**uRPF**). Malheureusement, cette technique a certaines limites qui permettent encore l'usurpation d'adresse IP source dans certains cas mais surtout elles

1. http://insecure.org/sploits/land.ip.DOS.html
2. http://www.cert.org/advisories/CA-1996-01.html
3. http://www.cert.org/advisories/CA-1998-01.html
4. http://nmap.org/
5. http://www.cert.org/advisories/CA-2003-04.html
6. http://www.cert.org/advisories/CA-1996-21.html

peuvent entraîner que du trafic légitime soit considéré comme illégitime (i.e., trafic ayant des adresses IP source falsifiées) et alors rejeté par l'entité appliquant la technique.

3.2.1 BCP 38

La variante statique de la technique de *Network Ingress Filtering* a été standardisée en premier afin de répondre rapidement à l'augmentation d'attaques DoS basées sur l'usurpation d'adresse IP source comme indiqué précédemment. L'idée est simplement de mettre en place des règles de filtrage au niveau des bordures du réseau des entreprises et fournisseurs d'accès Internet (e.g., au sein des routeurs de bordure). Ces règles vont vérifier si l'adresse IP source de chaque paquet, transitant par l'entité appliquant la technique, est légitime ou pas. Par légitime, cela indique que l'adresse IP source du paquet entrant sur l'interface de l'entité, est bien incluse dans le ou les préfixes IP alloués au réseau rattaché à cette interface. Tout paquet IP qui ne respecte pas ces règles est rejeté et loggé.

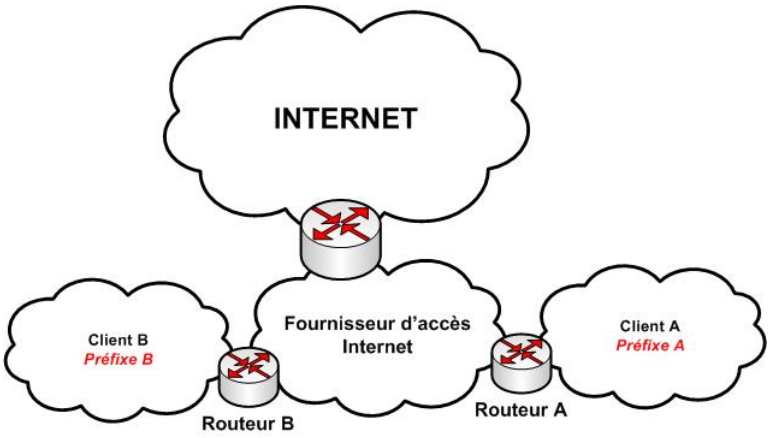

Figure 3.1 – Exemple d'architecture intégrant du *Network Ingress Filtering*

Ainsi, supposons, par exemple, que la technique de *Network Ingress Filtering* est déployée dans l'architecture réseau illustrée dans la Figure 3.1, où :

- Le préfixe `A` a été alloué au Client A par le fournisseur d'accès Internet ;
- Le préfixe `B` a été alloué au Client B par le fournisseur d'accès Internet ;
- Le fournisseur d'accès Internet a déployé du *Network Ingress Filtering* au niveau des routeurs A et B.

Alors, le Routeur A ne laissera sortir du réseau du client A que les paquets dont l'adresse IP source est comprise dans le préfixe `A`. De même, le Routeur B ne laissera sortir du réseau du client B que les paquets dont l'adresse IP source est comprise dans le préfixe `B`.

3.2.2 BCP 84

La limitation majeure du *BCP 38* est que les règles de filtrages sont configurées manuellement : cela peut entraîner des rejets de trafic légitime provenant d'un réseau, dans le cas d'une erreur humaine lors de la configuration des règles mais aussi, dans le cas où un nouveau préfixe IP serait alloué au réseau et la mise à jour des règles de filtrages ne se ferait pas assez rapidement. Le *BCP 84* [BS04], plus connu sous le nom de *unicast Reverse Path Forwarding* (**uRPF**), a été standardisé par l'IETF afin de répondre à cette problématique et est actuellement le plus répandu au niveau déploiement.

uRPF permet de mettre en place dynamiquement les règles de filtrage et pour cela, il s'appuie sur les tables de routage (e.g., BGP [RLH06] [MD99]). En effet, pour savoir si un paquet, contenant une adresse IP source spécifique et arrivant à l'entité effectuant le filtrage, est légitime, cette dernière va vérifier, grâce à sa table de routage, si dans le cas où elle aurait reçu un paquet contenant cette même adresse IP, mais cette fois-ci en adresse IP destination, elle aurait pu le router. Si ce n'est pas le cas, le paquet IP est rejeté et loggé.

Ainsi, par exemple, en reprenant l'architecture de la Figure 3.1, le Routeur A sait, grâce à sa table de routage, qu'il doit transmettre tout paquet IP contenant une adresse IP destination incluse dans le préfixe `A` vers le réseau du client A et que tout paquet avec une autre adresse IP destination doit être envoyé vers le réseau du fournisseur d'accès Internet. Aussi, en s'appuyant sur la table de routage, il considérera que tout paquet IP, sortant du réseau du client A, ne sera légitime que si l'adresse IP source est incluse dans le préfixe `A`. Toute autre adresse IP source entraînera que le paquet IP sera considéré comme illégitime (i.e., le paquet est rejeté et loggé).

Il existe 4 modes pour uRPF :

- mode *Strict uRPF*
  C'est le mode de base de uRPF. Dans ce mode, basé seulement sur le meilleur chemin de routage (i.e., *best routing path*), un paquet IP, avec une adresse IP source spécifique, reçu sur une interface de l'entité appliquant uRPF, est considéré comme légitime que si un paquet IP avec cette même adresse IP, mais cette fois ci en tant qu'adresse IP destination, serait envoyé par la même interface.
- mode *Feasible uRPF*
  Ce mode est une extension au mode *Strict uRPF* : les chemins de routage alternatifs sont pris en compte pour effectuer le test de légitimité d'un paquet IP (i.e., le paquet IP peut arriver sur plus d'une interface pour qu'il soit considéré comme légitime).
- mode *Loose uRPF*
  Ce mode est similaire au mode *Strict uRPF* mais pour que le paquet soit considéré comme légitime, il suffit qu'il existe un chemin de routage pour l'adresse IP source, et l'existence d'une route par défaut remplit la condition. De plus, l'interface sur laquelle le paquet IP arrive n'est pas prise en compte (i.e., le paquet peut arriver sur n'importe interface de l'entité exécutant uRPF).
- mode *Loose uRPF - Ignoring Default Routes*
  Ce mode est identique au mode *Loose uRPF* mais les routes par défaut ne sont pas tenues en compte pour le test de légitimité d'un paquet IP.

3.3 Limites

Même si la technique de *Network Ingress Filtering* permet d'atténuer la circulation d'un bon nombre de paquets IP contenant une adresse IP source falsifiée, cette technique a ses limites. En effet, dans certains cas (i.e., routage asymétrique et réseau *multihomé*), du trafic légitime peut être considéré comme illégitime par les entités appliquant du *Network Ingress Filtering*. Inversement, suite à une faible granularité dans les règles de filtrage, le *Network Ingress Filtering* peut considérer du trafic illégitime comme légitime.

3.3.1 Routage asymétrique

Un réseau pratiquant du routage asymétrique est un réseau où les flux IP entrants et les flux IP sortants passent par des routeurs distincts.

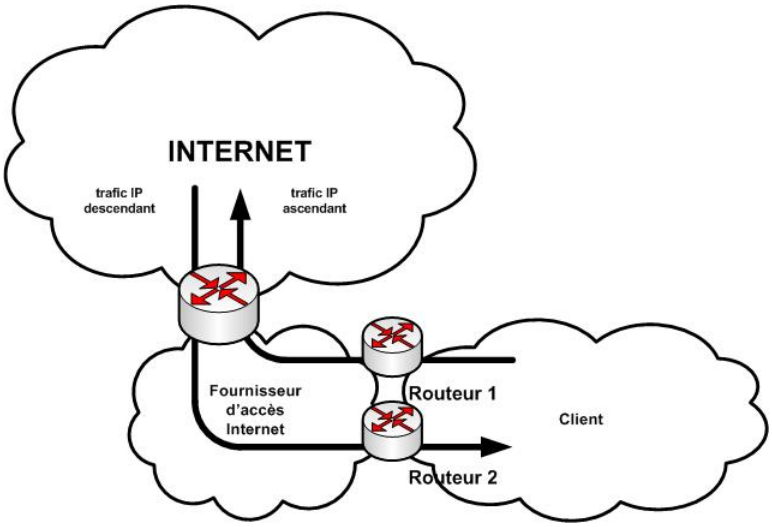

Figure 3.2 – Routage asymétrique

Ainsi, comme illustré dans la Figure 3.2, le Routeur 1 achemine le trafic sortant du réseau du client vers le fournisseur d'accès Internet et le Routeur 2 achemine le trafic entrant vers le réseau du client en provenance du fournisseur d'accès. Dans le cas où le Routeur 1 appliquerait du *Network Ingress Filtering*, si les informations de routage ne sont pas partagées convenablement entre les Routeur 1 et Routeur 2, du trafic légitime sera considéré comme illégitime et donc rejeté par le Routeur 1.

3.3.2 Réseau "multihomé"

Un réseau client dit *multihomé* est un réseau connecté à plusieurs fournisseurs d'accès Internet. Généralement, chacun de ces fournisseurs alloue un préfixe réseau différent au client. Maintenant, suivant la politique de routage du client, des paquets IP ayant une adresse IP source basée sur le préfixe d'un des fournisseurs pourraient être dirigés vers le réseau d'un des autres fournisseurs. Si ce dernier a déployé du *Network Ingress Filtering* sur le routeur connecté au client, ces paquets IP pourraient être considérés comme illégitimes et rejetés.

Par exemple, comme illustré dans la Figure 3.3, la politique de routage du client est d'acheminer tout son trafic sortant via le fournisseur d'accès Internet A (e.g., pour des raisons de qualité de service) et de recevoir tout son trafic entrant via le fournisseur d'accès B. Si le routeur A applique du *Network Ingress Filtering*, et le fournisseur B ne lui fournit aucune information, le trafic IP sortant du client, ayant une adresse IP source basée sur le préfixe réseau `B`, sera considéré comme illégitime et rejeté.

3.3.3 Granularité

Les principes du *BCP 38* et *BCP 84* reposent sur l'utilisation de règles de filtrage se basant sur des préfixes pour savoir si un paquet IP avec une adresse IP source spécifique est légitime ou pas. Cela signifie qu'un attaquant peut encore falsifier son adresse IP source si l'adresse utilisée est "topologiquement" correcte : l'attaquant usurpe une adresse incluse dans le préfixe IP alloué au réseau dans lequel il se trouve.

Ainsi, comme illustré dans la Figure 3.4, nous supposons que le fournisseur d'accès

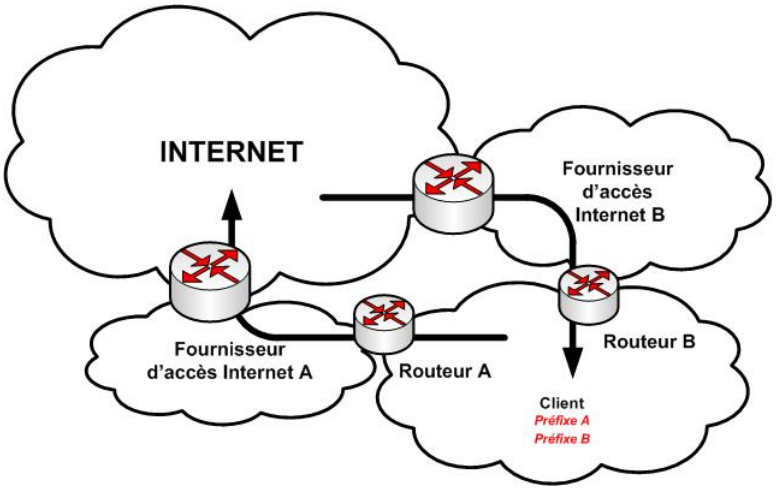

FIGURE 3.3 – Réseau *multihomé*

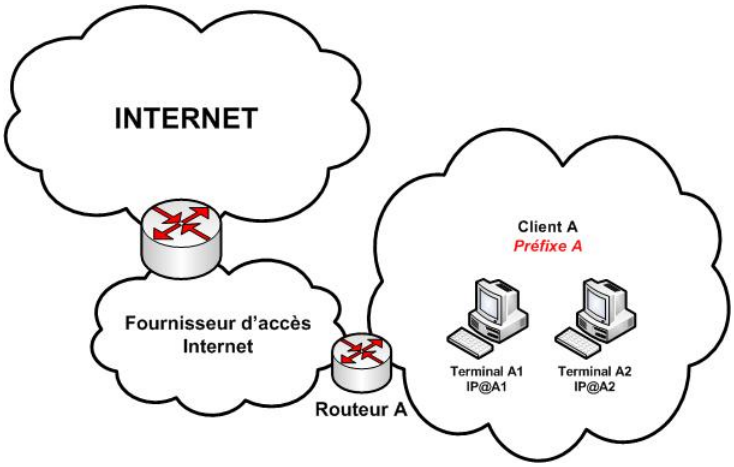

FIGURE 3.4 – Faiblesse de la granularité

Internet a alloué le préfixe IP `A` au client A et le routeur A de ce fournisseur d'accès effectue du *Network Ingress Filtering*. Alors, rien n'empêchera le terminal A2, qui a l'adresse IP `IP@A2` normalement, d'usurper l'adresse IP du terminal A1, à savoir l'adresse IP `IP@A1`. Le routeur A considérera alors le trafic de A2 comme légitime alors que ce n'est pas le cas. Avec les mêmes conséquences, le terminal A2 peut aussi usurper une adresse IP non assignée et incluse dans le préfixe `A`.

3.4 SAVI

Le but premier du groupe de travail *Source Address Validation Improvements* (**SAVI**) à l'IETF [7] est de répondre à la dernière limite explicitée précédemment : spécifier des mécanismes qui empêchent des terminaux connectés sur un même lien IP d'usurper des adresses IP de ce même lien. Il est à noter qu'il existe de tels mécanismes déjà mais ceux-ci

7. http://datatracker.ietf.org/wg/savi/charter/

sont partiels et propriétaires (e.g., *IP source guard*[8]).

En tant que président de ce groupe de travail (i.e., *WG chairman*) à l'IETF, j'ai supervisé, contribué et parrainé (i.e., *IETF document Shepherd*) les spécifications de ces mécanismes [CL12].

Il est important de signaler que, comme les solutions SAVI sont encore en cours de standardisation, leurs spécifications peuvent encore évoluer dans un proche futur.

3.4.1 Principes

Les solutions spécifiées dans SAVI [WBB+12] reposent sur l'observation du trafic des terminaux IP et l'observation, voire l'utilisation, des protocoles d'assignation/allocation d'adresses IP. Ces derniers sont :

- *Dynamic Host Configuration Protocol* (**DHCP**) [Dro97] en IPv4,
- *Stateless Address Autoconfiguration* (**SLAAC**) [TNJ07] en IPv6,
- *Secure Neighbor Discovery* (**SEND**) [AKZN05] en IPv6,
- *Dynamic Host Configuration Protocol for IPv6* (**DHCPv6**) [DBV+03] en IPv6.

Dans le cadre de ce mémoire, par la suite, nous nous intéresserons seulement aux solutions SAVI basées sur les protocoles SLAAC et SEND.

A partir de cette observation, une solution SAVI identifie quelle adresse IP est légitime pour un terminal IP. Elle peut alors lier, *SAVI Binding* (**SAVI-B**), cette adresse IP à un "point d'ancrage", *Binding Anchor* (**BAnchor**), qui identifie ce qui relie le terminal IP au réseau et ainsi au dispositif matériel intégrant la solution SAVI, l'*Entité SAVI*. Le niveau de sécurité du BAnchor conditionne le niveau de fiabilité des solutions SAVI. Comme illustré dans la Figure 3.5, si l'entité SAVI utilise comme BAnchor l'adresse MAC des terminaux IP, tout d'abord elle ne pourra différencier les adresses MAC des terminaux Terminal 1 et Terminal 2, respectivement `@MAC1` et `@MAC2`, car ces terminaux se trouvent derrière un switch de couche 2. L'entité SAVI ne peut connaître que l'adresse MAC de ce switch. Plus grave, il est facilement envisageable pour le terminal Terminal 3 d'usurper cette adresse MAC. Aussi, généralement, ce sont des points d'ancrage de l'entité SAVI qui sont choisis comme BAnchor, comme par exemple :

- si c'est un switch de couche 2, le BAnchor est un port de switch,
- si c'est un routeur IP, le BAnchor est soit une interface réseau de ce routeur, soit l'adresse MAC de cette interface réseau,
- si c'est un point d'accès réseau 802.11, le BAnchor est un *Security Association* 802.1x.

Finalement, la solution SAVI met en place la règle de filtrage suivante : un paquet est considéré comme légitime s'il existe un SAVI-B qui associe l'adresse IP source de ce paquet et le BAnchor par lequel transite ce paquet. Dans le cas contraire, le paquet est considéré comme illégitime : il sera rejeté et l'événement sera loggé.

Dans la suite de ce chapitre, il est considéré lors de la description des différentes solutions SAVI que le BAnchor est de type port de switch car les entités SAVI déployées seront généralement un switch de couche 2.

Afin d'optimiser les solutions SAVI en réduisant au minimum l'activité des entités SAVI, le principe de périmètre de protection SAVI a été défini. Ce périmètre de protection est délimité par les entités SAVI. Il permet de classifier les flux IP en 2 catégories : ceux provenant de l'intérieur du périmètre, qui seront considérés comme de confiance et donc légitimes, et ceux provenant de l'extérieur du périmètre, et qui ne seront pas considérés de confiance. Aussi, les solutions SAVI ne s'appliqueront qu'à cette dernière catégorie de

8. http://www.cisco.com/en/US/docs/switches/lan/catalyst6500/ios/12.2SY/configuration/guide/ip_source_guard.html

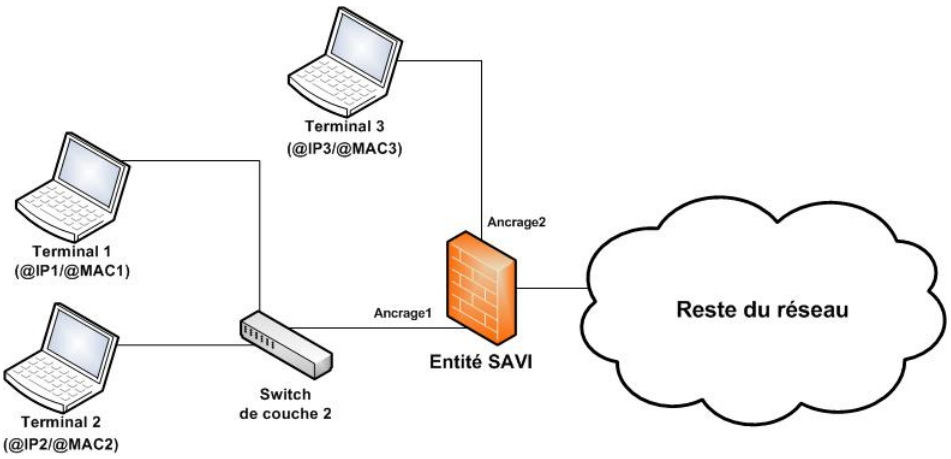

FIGURE 3.5 – Architecture SAVI

flux. Du coup, comme illustré dans la figure 3.6, l'entité SAVI `SAVI 3` n'a pas à se soucier des flux provenant du nœud `Nœud IP 1`, car les flux provenant de ce nœud sont déjà filtrés par l'entité SAVI `SAVI 1` qui fait partie du périmètre de protection SAVI.

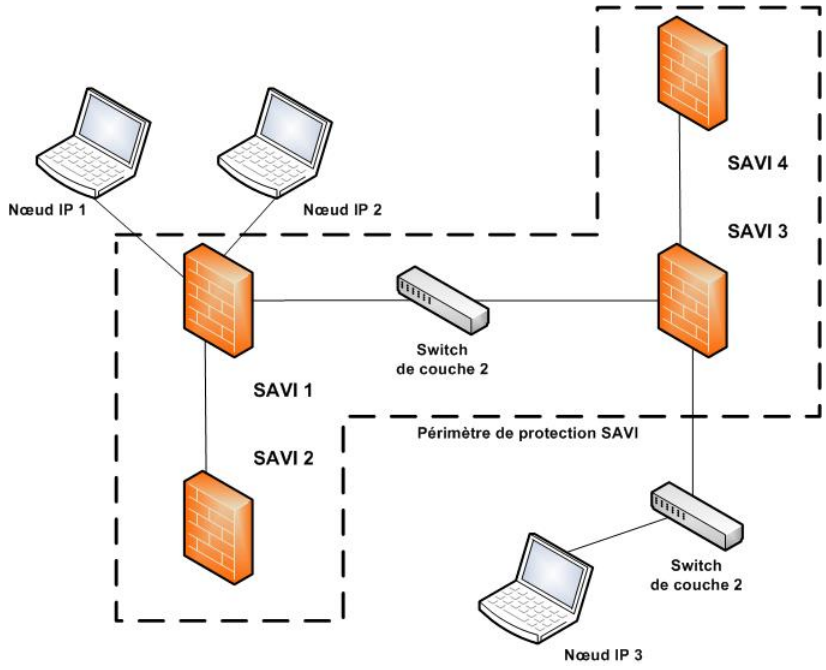

FIGURE 3.6 – Exemple de Périmètre de protection SAVI

Enfin, les solutions SAVI peuvent nécessiter de générer ou reconfigurer des SAVI-B pour certaines raisons. En effet, tout d'abord, la signalisation SLAAC, ainsi que SEND, n'est pas fiable et certains des échanges peuvent être "perdus", ce qui empêchera la génération correcte d'un SAVI-B. Autre situation, c'est lorsque le nœud IP est mobile et il se peut qu'il change de BAnchor lors d'un mouvement. Enfin, tout simplement, suite à un soucis technique (e.g., coupure de courant), l'entité SAVI peut perdre toute sa base de données stockant les SAVI-B. Aussi, chaque solution SAVI a un mécanisme de récupération de

SAVI-B.

3.4.2 FCFS SAVI

La solution *First-Come First-Serve SAVI* (**FCFS SAVI**) [NBLA12] utilise la signalisation SLAAC pour générer les SAVI-B. Avec SLAAC, c'est le premier nœud générant une adresse IPv6 spécifique qui peut se l'assigner et ainsi, avec cette solution, aura un SAVI-B correspondant à cette adresse IPv6. C'est de là que provient le nom de cette solution SAVI.

La solution FCFS SAVI définit 2 types de port : le *Trusted Port* (**TP**) et le *Validating Port* (**VP**). Les flux IP arrivant sur un TP proviennent de l'intérieur du périmètre de protection SAVI alors que ceux arrivant sur un VP proviennent de l'extérieur de ce périmètre. Il est à noter que la solution FCFS SAVI suppose que tout routeur IP est inclus au sein du périmètre de protection, comme illustré dans la Figure 3.7.

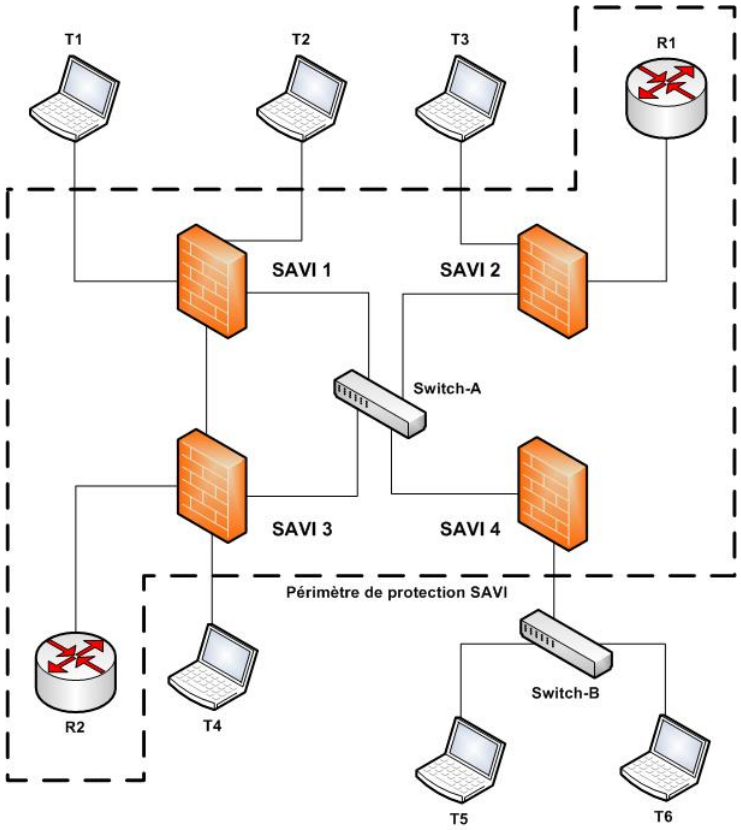

Figure 3.7 – Architecture FCFS SAVI

La solution FCFS SAVI utilise 2 bases de données. La première, nommée *FCFS SAVI Data Base* (**FCFS SAVI DB**), permet de stocker les SAVI-B. Lors de l'initialisation de l'entité SAVI, la FCFS SAVI DB est vide. Chaque entrée de cette base de données est constituée des informations suivantes :

- `IP ADDRESS`
  Indique une adresse IP source.
- `BINDING ANCHOR`
  Indique le BAnchor lié à cette adresse IP source.
- `LIFETIME`
  Indique la durée de vie du SAVI-B.

– `STATUS`
  Indique l'état du SAVI-B (i.e., `TENTATIVE`, `VALID`, `TESTING_VP` ou `TESTING_TP-LT`).
– `CREATION TIME`
  Indique quand l'entrée a été la première fois créée.

La deuxième base de données est la *FCFS SAVI Prefix List* (**FCFS SAVI PL**). Elle permet de stocker les préfixes IP employés sur les liens réseau et observés par l'entité SAVI. La FCFS SAVI PL est complétée, soit manuellement par l'administrateur de l'entité SAVI, soit dynamiquement grâce aux messages RA reçus sur un TP. Chaque entrée de la FCFS SAVI PL contient les informations suivantes :

– `PREFIX`
  Indique un préfixe IP.
– `PORT`
  Indique le port sur lequel est observé ce préfixe IP.

Lorsqu'un paquet IP est reçu sur l'un des ports de l'entité SAVI, le port `P` par exemple, celui-ci est traité en considérant les différents cas suivants :

1. Si le port est un TP,
   Alors le paquet est considéré comme de confiance (i.e., il provient de l'intérieur du périmètre de protection SAVI) et donc légitime. De plus, si c'est un message NS impliqué dans une procédure DAD (**DAD_NS**), cela active le processus FCFS SAVI.
2. Si le port est un VP et dans le cas où,
   (a) le préfixe lié à l'adresse IP source n'est pas dans la FCFS SAVI PL,
   celui-ci doit provenir d'un nœud IP dans un lien réseau non directement connecté à l'entité SAVI, donc devrait être normalement acheminé par un routeur et arriver sur un TP. Du coup, le paquet IP est considéré comme illégitime et rejeté.
   (b) le préfixe lié à l'adresse IP source est dans la FCFS SAVI PL,
   alors le processus FCFS SAVI, décrit par la suite, est activé.
   (c) l'adresse IP source est l'adresse indéfinie (*unspecified address*, `::0/128`),
      i. et c'est un message DAD_NS,
      alors le processus FCFS SAVI est activé.
      ii. sinon,
      le paquet IP est considéré comme légitime et est acheminé.

Par défaut, une adresse IP source, `@IP`, n'ayant pas d'entrée dans la FCFS SAVI DB, se trouve dans l'état `NO_BIND`. Suite à une des conditions d'activation de la procédure FCFS SAVI mentionnées auparavant, l'entité SAVI vérifie si un SAVI-B existe pour l'adresse `@IP`.

Si ce n'est pas le cas, un SAVI-B est créé et (1) l'état devient `TENTATIVE`, comme illustré dans la Figure 3.8. Dans le cas 2.c.i, le processus FCFS SAVI attend la fin de la procédure DAD : si elle échoue, (2) l'état repasse en `NO_BIND` sinon (3) il passe en `VALID`. Pour le cas 2.b, l'entité SAVI initie une procédure DAD en se faisant passer pour le nœud d'adresse IP `@IP` (i.e., l'entité SAVI usurpe l'adresse IP `@IP`) : si elle échoue, (2) l'état repasse en `NO_BIND` sinon (3) il passe en `VALID`. En arrivant à l'état `VALID`, tout paquet ayant comme adresse IP source `@IP` passant par le port `P` est considéré comme légitime.

Si un SAVI-B existe pour l'adresse `@IP`, avec un état `VALID` pour le port `P`, et qu'un DAD_NS est reçu sur un TP (i.e., cas 1), cela peut indiquer que le nœud IP s'est connecté ailleurs et (4) l'état passe à `TESTING_TP-LT`. Si la procédure DAD réussit, alors (7) l'état

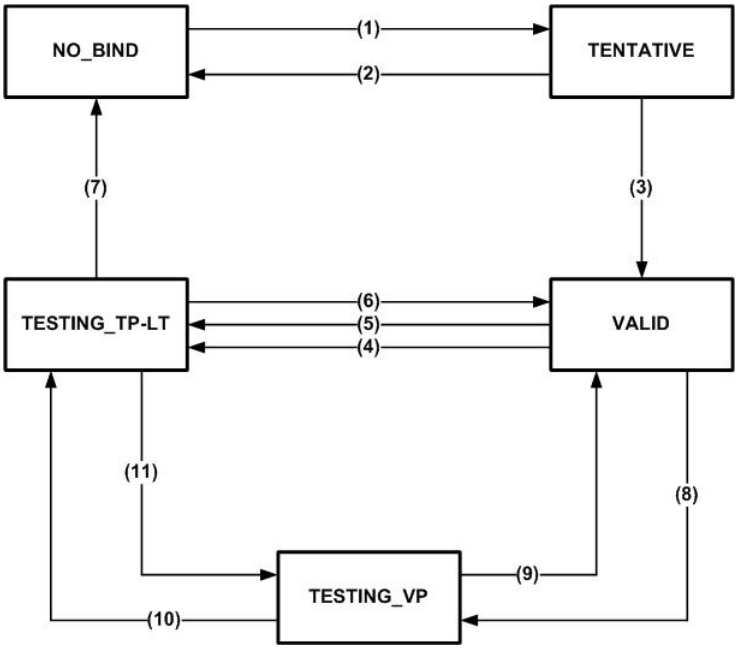

Figure 3.8 – Machine à états de FCFS SAVI

passe à `NO_BIND` sinon si elle échoue, soit (6) l'état passe à `VALID` si la réponse dans le DAD était un NA reçu sur le port `P`, soit (11) l'état passe à `TESTING_VP` si la réponse a été reçue sur un port différent `P'`.

Si un SAVI-B existe pour l'adresse `@IP`, avec un état `VALID` pour le port `P`, et que le paquet a été reçu sur un port différent `P'`, alors (8) l'état passe à `TESTING_VP`. Dans le cas où le paquet était un DAD_NS et que la procédure DAD réussit, alors (9) l'état repasse à `VALID` avec maintenant le port `P'` comme BAnchor dans le SAVI-B. Si la procédure DAD échoue, à cause d'un message NA reçu sur le port `P` alors (9) l'état repasse à `VALID`. Si la procédure DAD échoue, à cause d'un DAD_NS reçu sur le port `P"`, différent de `P` et `P'`, alors l'état reste à `TESTING_VP`. Si la procédure DAD échoue, à cause d'un DAD_NS reçu via un TP, alors (10) l'état passe à `TESTING_TP-LT`. Dans le cas où le paquet était autre chose qu'un DAD_NS, alors l'entité SAVI initie une procédure DAD en se faisant passer pour le nœud d'adresse IP `@IP` (i.e., l'entité SAVI usurpe l'adresse IP `@IP`) et le même processus qu'avec un DAD_NS est appliqué.

Finalement, comme décrit auparavant, chaque SAVI-B a une durée de vie (i.e., champ `LIFETIME`). Cette dernière est renouvelée régulièrement en se basant sur l'activité du nœud IP (e.g., flux de données de ce nœud). Un SAVI-B ayant l'état `VALID` et dont la durée de vie expire, (5) passe à l'état `TESTING_TP-LT` et l'entité SAVI initie une procédure DAD en se faisant passer pour le nœud d'adresse IP `@IP` (i.e., l'entité SAVI usurpe l'adresse IP `@IP`) : si elle échoue, (2) l'état repasse en `NO_BIND` sinon (3) il passe en `VALID`.

3.4.3 SEND SAVI

Le mécanisme SEND SAVI [BGM12] utilise la signalisation SEND pour générer les SAVI-B. En effet, lorsque SEND est déployé, la solution FCFS SAVI ne peut être utilisée : l'entité SAVI peut être obligée d'usurper l'adresse IP d'un des nœuds connectés, ce qui n'est pas possible avec SEND.

Comme FCFS SAVI, la solution SEND SAVI définit 2 types de port : le *Trusted*

Port (**TP**) et le *Validating Port* (**VP**). Les flux IP arrivant sur un TP proviennent de l'intérieur du périmètre de protection SAVI alors que ceux arrivant sur un VP proviennent de l'extérieur de ce périmètre. La solution SEND SAVI suppose qu'un seul nœud IP est connecté sur chaque port et que, contrairement à FCFS SAVI, un routeur IP n'est pas obligatoirement inclus au sein du périmètre de protection, comme illustré dans la Figure 3.9.

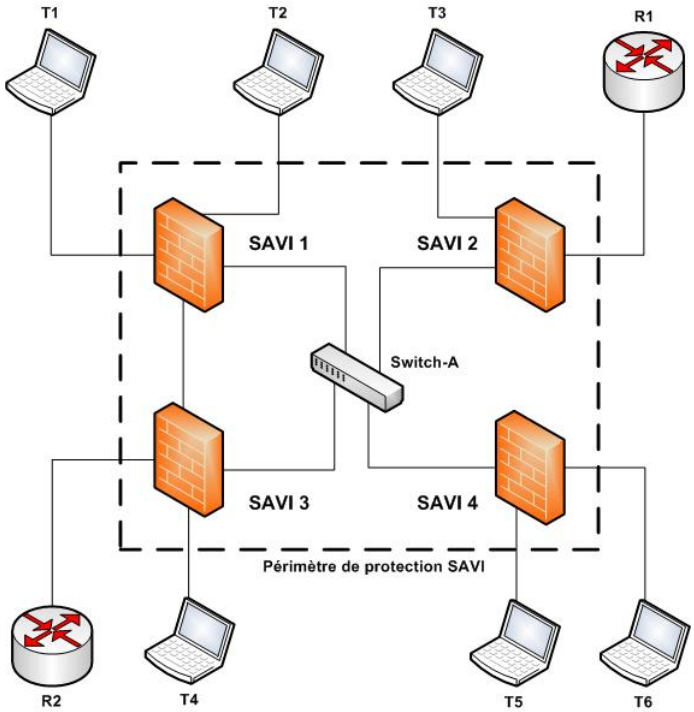

FIGURE 3.9 – Architecture SEND SAVI

La solution SEND SAVI utilise 3 bases de données. La première, nommée *SEND SAVI Data Base* (**SEND SAVI DB**), permet de stocker les SAVI-B. Lors de l'initialisation de l'entité SAVI, la SEND SAVI DB est vide. Chaque entrée de cette base de données est constituée des informations suivantes :

- `IP ADDRESS`
  Indique une adresse IPv6 source.
- `BINDING ANCHOR`
  Indique le BAnchor lié à cette adresse IPv6 source.
- `LIFETIME`
  Indique la durée de vie du SAVI-B.
- `STATUS`
  Indique l'état du SAVI-B (i.e., `TENTATIVE_DAD`, `TENTATIVE_NUD`, `VALID`, `TESTING_VP` ou `TESTING_VP'`).
- `ALTERNATIVE BINDING ANCHOR`
  Indique un BAnchor, différent de celui dans `BINDING ANCHOR`, par lequel un DAD_NS ou un paquet de données est arrivé.
- `CREATION TIME`
  Indique quand l'entrée a été la première fois créée.

La deuxième base de données est la *SEND SAVI Prefix List* (**SEND SAVI PL**).

Elle permet de stocker les préfixes IP employés sur les liens réseau et observés par l'entité SAVI. La SEND SAVI PL est complétée, soit manuellement par l'administrateur de l'entité SAVI, soit dynamiquement grâce aux messages RA reçus suite à une requête RS envoyée régulièrement par l'entité SAVI. Chaque entrée de la SEND SAVI PL contient les informations suivantes :

- `PREFIX`
  Indique un préfixe IPv6.
- `PREFIX LIFETIME`
  Indique le temps de vie du préfixe IPv6.

Enfin, la dernière base de donnée est la *SEND SAVI Router List* (**SEND SAVI RL**). Elle permet de répertorier les routeurs attachés à un VP. La SEND SAVI RL est complétée, soit manuellement par l'administrateur de l'entité SAVI, soit dynamiquement grâce aux messages RA reçus suite à une requête RS envoyée régulièrement par l'entité SAVI. Chaque entrée de la SEND SAVI RL contient les informations suivantes :

- `IPV6 ADDRESS`
  Indique l'adresse IPv6 d'un routeur.
- `ROUTER LIFETIME`
  Indique le temps de vie de ce routeur.

La solution SEND SAVI ne peut initier une procédure DAD en se faisant passer pour le nœud d'adresse IP `@IP` (i.e., l'entité SAVI usurpe l'adresse IP `@IP`), comme c'est le cas avec FCFS SAVI. En effet, l'entité SAVI ne possède pas les secrets nécessaires pour signer les messages avec SEND. Aussi, la solution SEND SAVI va utiliser à la place la procédure NUD, précédemment décrite dans la Section 1.3.1. Du coup, il est nécessaire de configurer une adresse CGA sur l'entité SAVI.

Lorsqu'un paquet IP est reçu sur l'un des ports de l'entité SAVI, le port `P` par exemple, celui-ci est traité en considérant les différents cas suivants :

1. Si le port est un TP,
   Alors le paquet est considéré comme de confiance (i.e., il provient de l'intérieur du périmètre de protection SAVI) et donc légitime.
2. Si le port est un VP et dans le cas où,
   (a) le préfixe lié à l'adresse IP source n'est pas dans la SEND SAVI PL,
      i. s'il existe un SAVI-B pour le port `P` et que l'adresse IP `@IP` associée est dans la SEND SAVI RL,
         alors c'est du trafic acheminé par un routeur et le paquet IP est considéré comme légitime.
      ii. sinon,
         le paquet est illégitime et rejeté.
   (b) le préfixe lié à l'adresse IP source est dans la SEND SAVI PL,
      alors le processus SEND SAVI, décrite par la suite, est activé.
   (c) l'adresse IP source est l'adresse indéfinie (*unspecified address*, `::0/128`),
      alors le processus SEND SAVI est activé.

Par défaut, une adresse IP source, `@IP`, n'ayant pas d'entrée dans la SEND SAVI DB, se trouve dans l'état `NO_BIND`. Suite à une des conditions d'activation de la procédure SEND SAVI mentionnées auparavant, l'entité SAVI vérifie si un SAVI-B existe pour l'adresse `@IP`.

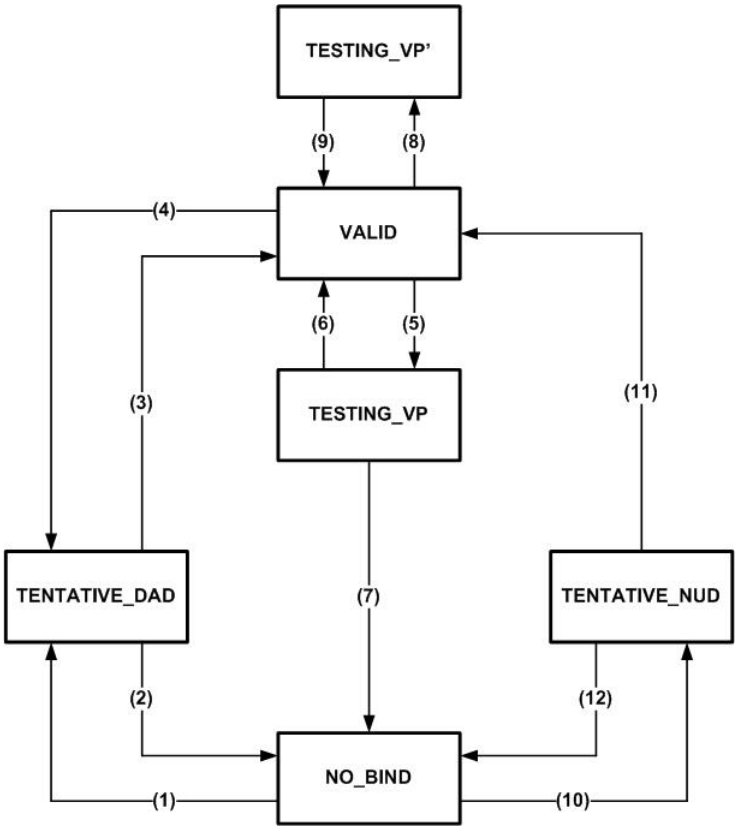

FIGURE 3.10 – Machine à états de SEND SAVI

Si ce n'est pas le cas, un SAVI-B est créé. Dans le cas 2.c, (1) l'état devient `TENTATIVE_DAD`, comme illustré dans la Figure 3.10. Le processus SEND SAVI attend la fin de la procédure DAD : si elle échoue, (2) l'état repasse en `NO_BIND` sinon (3) il passe en `VALID`. Pour le cas 2.b, (10) l'état devient `TENTATIVE_NUD` et l'entité SAVI initie une procédure NUD : si elle échoue, (12) l'état repasse en `NO_BIND` sinon (11) il passe en `VALID`.

Si un SAVI-B existe pour l'adresse `@IP`, avec un état `VALID` pour le port `P`, cela peut indiquer que le nœud IP s'est connecté ailleurs. L'entité SAVI initie une procédure NUD et (5) l'état passe à `TESTING_VP`. Si la procédure NUD échoue, alors (7) l'état passe à `NO_BIND` sinon (6) l'état repasse à `VALID`.

Si un SAVI-B existe pour l'adresse `@IP`, avec un état `VALID` pour le port `P`, et qu'un DAD_NS est reçu sur un port différent `P'`, cela peut indiquer que le nœud IP s'est connecté ailleurs. L'entité SAVI initie une procédure NUD et (8) l'état passe à `TESTING_VP'`. Si la procédure NUD réussit, alors (7) l'état repasse à `VALID` sinon (9) l'état repasse à `VALID` avec maintenant le port `P'` comme BAnchor dans le SAVI-B.

Si un SAVI-B existe pour l'adresse `@IP`, avec un état `VALID` pour le port `P`, et qu'un DAD_NS est reçu sur `P`, (4) l'état devient `TENTATIVE_DAD`. Le processus SEND SAVI attend la fin de la procédure DAD : si elle échoue, (2) l'état passe en `NO_BIND` sinon (3) il repasse en `VALID`.

Finalement, comme décrit auparavant, chaque SAVI-B a une durée de vie (i.e., champ `LIFETIME`). Cette dernière est renouvelée régulièrement en se basant sur l'activité du nœud IP (e.g., flux data). Un SAVI-B ayant l'état `VALID` et dont la durée de vie expire, (5) passe à l'état `TESTING_VP` et l'entité SAVI initie une procédure NUD : si elle échoue, alors (7)

l'état passe à `NO_BIND` sinon (6) l'état repasse à `VALID`.

Avec la solution FCFS SAVI, un paquet, arrivant sur le port `P` avec une adresse IP source `@IP`, est considéré comme légitime s'il existe un SAVI-B associé à `@IP` avec `P` comme BAnchor et `VALID` comme état. Avec la solution SEND SAVI, le même paquet est considéré légitime aussi avec les états `TENTATIVE_DAD`, `TESTING_VP` et `TESTING_VP'`.

3.4.4 Mix SAVI

Différentes solutions SAVI (i.e., DHCP SAVI [BWYB12], FCFS SAVI, SEND SAVI) peuvent être déployées dans un même réseau si ce dernier utilise plusieurs mécanismes d'allocation/assignation d'adresse IP (e.g., Autoconfiguration IPv6 et SEND) [BYHLA11a]. Cela peut entraîner des problèmes de collisions lors de la génération de SAVI-B.

Afin de limiter de potentielles collisions, il est recommandé d'utiliser des préfixes réseau différents pour chacun des mécanismes d'allocation/assignation d'adresse IP.

Il existe deux cas de collision : une même adresse se retrouve liée à 2 BAnchors (e.g., un nœud A, connecté au BAnchor `X`, récupère l'adresse `@IP` via SLAAC et plus tard un nœud B, connecté au BAnchor `Y`, se voit allouer la même adresse `@IP` via DHCPv6) et une même adresse se retrouve liée plusieurs fois à un même BAnchor (e.g., un nœud A, connecté au BAnchor `X`, récupère l'adresse `@IP` via SLAAC et plus tard, se voit allouer la même adresse `@IP` via DHCPv6). Il est recommandé dans le premier cas de garder le SAVI-B qui est créé en premier, sauf quand SAVI-SEND est utilisé et alors c'est le SAVI-B généré par cette solution qui doit être gardé. Pour le second cas, le principal impact concerne le temps de vie du SAVI-B, aussi, il est recommandé de garder le SAVI-B jusqu'à ce que toutes les solutions SAVI déployées recommandent sa suppression.

3.5 Intégration concrète : réseau d'accès IPv6 ADSL/Fibre

Afin d'optimiser le potentiel des solutions SAVI, il est important de décider où seront déployées les entités SAVI, ainsi que le type de BAnchor qui sera utilisé.

La solution FCFS SAVI, et SEND SAVI si SEND est déployé aussi, est nécessaire dans une architecture IPv6 d'accès ADSL/Fibre de type *split-horizon forwarding*, architecture standardisée par le *Broadband Forum* (**BBF**) [9], afin de protéger contre l'usurpation d'adresse IP source. Ce type d'architecture, illustré dans la Figure 3.11, sera déployé chez France Telecom - Orange par exemple.

Dans ce type d'architecture, le premier routeur IP du réseau du fournisseur d'accès Internet est la emphBroadband Network Gateway (**BNG**). Un client du fournisseur d'accès Internet est connecté au réseau de ce dernier grâce à un *Customer Premises Equipment* (**CPE**), qui est soit un modem-routeur, soit un terminal IP. L'entité, appelé *Access Node* (**AN**), est un *bridge* de couche 2. Son rôle est d'empêcher que des CPEs communiquent entre eux directement. Cela oblige ainsi les flux de données provenant d'un CPE à passer par la BNG. Du point de vue du CPE, au niveau de la couche IP, chaque CPE est "connecté" point à point à la BNG (i.e., le CPE "pense" qu'il n'y a que le routeur sur le même lien réseau). Du point de vue de la BNG, ce n'est pas le cas par contre : par exemple, il peut communiquer avec plusieurs CPEs en même temps (e.g., multicast en direction de CPE1, CPE2 et CPE3 sur la Figure 3.11).

Ce type d'architecture empêche du coup l'emploi du mécanisme SLAAC (cf. Section 1.4) par un CPE. En particulier, une fois une adresse générée, il ne peut pas détecter, grâce à la procédure DAD (cf. Section 1.4.2), si un autre CPE se l'est assignée déjà ou

9. http://www.broadband-forum.org/technical/download/TR-177.pdf

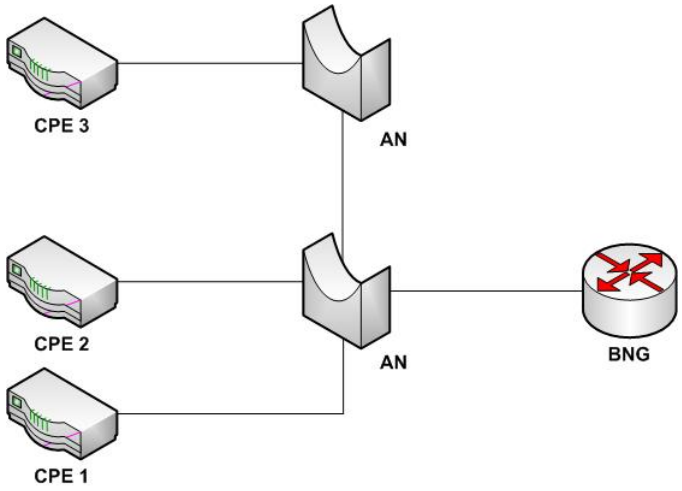

Figure 3.11 – Architecture ADSL/Fibre de type *split-horizon forwarding*

pas : les messages NS sont acheminés obligatoirement à la BNG et celle-ci ne les routent pas ensuite vers les autres CPEs car elles utilisent une LUA comme adresse IPv6 source.

Le BBF a réclamé, auprès de l'IETF, la spécification d'une solution qui réponde à cette problématique et protège aussi contre l'usurpation d'adresses IPv6. Dans ce cadre là, nous avons soumis une telle solution [CPLC12] et elle a été accepté comme document de travail officiel à l'IETF : notre contribution deviendra le standard (i.e., RFC) concernant cette problématique.

Notre solution est composée de deux parties : une partie spécifiant le mécanisme *DAD Proxy*, permettant l'utilisation du mécanisme SLAAC par un CPE, et une partie détaillant l'intégration des mécanismes SAVI, protégeant contre l'usurpation d'adresses IPv6.

Concernant la première partie, la BNG intègre une base de données, la *Binding Table* (**BT**), où chaque entrée contient une adresse IPv6 et le CPE qui se l'est assignée. Celle-ci est mise à jour à chaque fois que la BNG observe un messages NS émis par un CPE lors d'une procédure DAD. Si l'adresse impliquée dans le message est contenue dans la BT, alors la BNG vérifie grâce à la procédure NUD (cf. Section 1.3.1) si elle est toujours assignée au CPE indiqué dans la BT. Dans le cas d'une réponse positive, la BNG forge un message NA, semblant provenir de ce dernier CPE, et l'envoie au CPE effectuant la procédure DAD. Dans le cas d'une réponse négative ou si l'adresse IPv6 n'était pas contenue dans la BT, la BNG met à jour la BT en indiquant que l'adresse IPv6 en question correspond maintenant au CPE effectuant la procédure DAD.

Concernant la deuxième partie, nous avons spécifié comment utiliser les solutions FCFS SAVI et SEND SAVI dans ce type d'architecture. Ainsi, l'entité SAVI doit être la BNG. Le BAnchor doit correspondre à l'adresse MAC du CPE. Grâce au mécanisme *Virtual MAC* (**VMAC**), intégré dans l'AN connecté à ce CPE, l'unicité du point d'ancrage du nœud IP, et donc le niveau de sécurité du BAnchor, est assuré.

3.6 Implémentations et déploiement existants de SAVI

Il est intéressant de savoir quelles sont les implémentations disponibles actuellement et si elles sont actuellement déployées.

3.6.1 Implémentations existantes

A ce jour, les principales implémentations officiellement [10] connues sont :
- draft-ietf-savi-dhcp-07 [BWYB10],
- draft-bi-savi-stateless-01 [BYWB10],
- draft-bi-savi-mix-04 (partiellement) [BYHLA11b],
- draft-an-savi-mib-00 [AJWB10].

Et elles sont produites par :
- ZTE, Huawei, H3C (3Com),
- Ruijie, Digital China ("spin off" de Lenovo),
- Bitway, Centac.

3.6.2 Déploiement existant

Le réseau de recherche universitaire de Chine *CERNET2* a déployé les solutions SAVI implémentées qui sont mentionnées auparavant. Les entités SAVI sont des switchs de couche 2. Ces solutions sont déployées sur une centaine de sites, illustré dans la Figure 3.12. Cela représente environ un million d'utilisateurs impactés par ces mécanismes.

Figure 3.12 – Déploiement SAVI en Chine

3.7 Limites des solutions SAVI

Les solutions SAVI ont aussi des limites. En effet, comme ces solutions reposent principalement sur l'observation de protocoles utilisés dans les réseaux, elles héritent des inconvénients liés à ceux-ci.

3.7.1 Fragmentation

La signalisation observée devient complexe lorsque les solutions SAVI sont intégrées dans du matériel ayant peu de ressources de calcul (e.g., switch de couche 2). Dans le cas

10. http://www.ietf.org/proceedings/80/slides/savi-2.pdf

où la signalisation IP est fragmentée [DH98], l'interprétation de celle-ci devient coûteuse pour l'entité SAVI, voire impossible. Aussi, l'entité SAVI ne pourra être capable de générer les SAVI-B. Du coup, du trafic légitime sera considéré comme illégitime.

3.7.2 Optimistic DAD

Dans le cas où la procédure oDAD [Moo06], présentée dans la Section 1.4.2, serait utilisée dans une architecture où des solutions SAVI seraient déployées, les flux IPv6 utilisant ce type d'adresse seront considérés comme illégitimes (car aucun SAVI-B n'existera), et donc rejetés.

3.7.3 Privacy

Les solutions SAVI permettent d'associer un nœud IP à une adresse IP. La surveillance d'un réseau peut s'appuyer sur ces solutions, particulièrement quand une méthode d'allocation d'adresse générant un IID aléatoire (i.e., non basé sur l'adresse MAC du nœud IP) est utilisée (cf. Section 1.2). Cette technique pourrait aider les administrateurs à administrer leurs réseaux [Cho11] : quel nœud IP, identifié par son BAnchor, envoie et reçoit quel trafic. Maintenant, une telle surveillance va à l'encontre du principe de la vie privée des utilisateurs définie par l'IETF [II00] : suite à un consensus, il a été décidé dans le passé que cet organisme ne spécifiera pas de mécanismes de surveillance. Aussi, des discussions sur le sujet au sein de l'IETF ont ralenti la standardisation des solutions SAVI. Finalement, il a été décidé qu'il est prohibé de logger les SAVI-B : ainsi, seules les tentatives d'usurpation d'adresses IP seront loggées.

3.8 Potentiels futurs travaux au sein de l'IETF

A court terme, afin de simplifier le déploiement et la gestion des solutions SAVI, l'utilisation du protocole *Simple Network Management Protocol* (**SNMP**) [HPW02] peut être utile. Aussi, il est nécessaire de spécifier une *Management Information Base* (**MIB**) pour les solutions SAVI. Les travaux sur ce sujet ont actuellement débuté [AJWB11].

A moyen terme, comme toute nouvelle solution spécifiée, il est nécessaire d'avoir un retour concret des implémentations et déploiements des solutions SAVI. Il y aura certainement des modifications à effectuer du coup dans les spécifications de ces solutions.

A long terme, comme les solutions SAVI actuellement spécifiées ne couvrent pas toutes les techniques d'allocation/assignement d'adresse IP, il pourrait être intéressant, selon le besoin de combler ce manque. Tout particulièrement, dans un contexte de *Virtual Private Network* (**VPN**) IPsec [KS05] où un terminal IP se connecte à son réseau d'entreprise, le protocole *Internet Key Exchange Protocol Version 2* (**IKEv2**) [KHNE10] [ELM10] permet de lui allouer une adresse IP interne à ce réseau. Il est nécessaire alors de se protéger d'un nœud hostile localisé dans ce même réseau désirant usurper l'adresse allouée.

Les solutions SAVI, complémentaires à la technique de *Network Ingress Filtering*, ne règlent qu'une des limites de cette technique : la granularité. Aussi, il pourrait être nécessaire de réviser cette technique afin de résoudre, ou limiter, les 2 dernières limites.

Enfin, certaines personnes dans la communauté IETF ont soulevé la possibilité d'étendre le périmètre de protection SAVI, dépendant d'un seul opérateur, à un périmètre plus global intégrant plusieurs opérateurs et plusieurs technologies (i.e., SAVI et *Network Ingress Filtering*). Cela permettrait de former une zone de confiance où les flux à surveiller et analyser ne seraient plus qu'en bordure de celle-ci.

3.9 Synthèse du chapitre

Dans ce chapitre, nous avons présenté la nouvelle technique SAVI, complémentaire à la technique *Network Ingress Filtering* permettant de lutter contre l'usurpation de l'adresse IP source. Les différentes solutions SAVI sont spécifiées, sous notre responsabilité, à l'IETF.

Dans un premier temps, nous avons rappelé les principales attaques reposant sur l'usurpation de l'adresse IP source (cf. Section 3.1) puis la réponse apportée par l'IETF, la technique de *Network Ingress Filtering* (cf. Section 3.2). Nous avons ensuite expliqué alors les limites de cette technique qui ont conduit l'IETF a standardisé les solutions SAVI (cf. Section 3.3). Puis, nous avons décrit ces différentes solutions, FCFS SAVI, SEND SAVI et MIX SAVI (cf. Section 3.4). Il est important de rappeler que, comme les solutions SAVI sont encore en cours de standardisation, leurs spécifications peuvent encore évoluer dans un proche futur.

Ensuite, nous avons présenté notre proposition à l'IETF concernant l'intégration de SAVI dans le cas concret d'une architecture IPv6 d'accès IPv6 ADSL/Fibre (cf. Section 3.5). Elle repose sur les solutions FCFS SAVI, et SEND SAVI si le mécanisme SEND est déployé dans une telle architecture. Etant standardisée au BBF, elle est le standard international lors d'un déploiement IPv6 pour les technologies ADSL et Fibre. Aussi, comme notre proposition doit prochainement devenir une RFC (cela devrait être finalisé d'ici Octobre 2012), notre solution sera mise en oœuvre chez tout fournisseur d'accès Internet employant une telle architecture.

Finalement, nous avons également mentionné les implémentations/déploiements existants à ce jour (cf. Section 3.6). Et, nous avons décrit les limites propres à SAVI (cf. Section 3.7) et les potentiels futurs travaux à l'IETF concernant le sujet (cf. Section 3.8).

Chapitre 4

Authentification dans IKEv2 à base de CGA

Aujourd'hui, la famille de protocoles *Internet Protocol security* (**IPsec**) est largement utilisée pour sécuriser les flux au sein de l'Internet. En effet, IPsec permet de fournir authentification et/ou chiffrement à n'importe quelle communication IP. Afin d'établir dynamiquement des connexions IPsec entre des nœuds IP, l'utilisation du protocole *Internet Key Exchange version 2* (**IKEv2**) est recommendée. IKEv2 peut utiliser 3 types de matériel de sécurité lors de l'authentification d'un nœud IP : clés secrètes partagées, certificats X.509 ou *Extensible Authentication Protocol* (**EAP**).

Chacun de ces types de matériel de sécurité possède des inconvénients liés au déploiement. La provision des clés secrètes est difficile à effectuer à grande échelle. Les certificats X.509 nécessitent de mettre en place et administrer une infrastructure de confiance, comme une *Public Key Infrastructure* (**PKI**). Enfin, l'intégration d'EAP n'est pas obligatoire dans les implémentations de IKEv2, ce qui peut entraîner des problèmes d'interopérabilité.

D'un autre côté, comme nous l'avons vu (cf. Section 1.2.4), les propriétés cryptographiques des adresses CGA permettent de prouver la possession d'une telle adresse, et ainsi de fournir une authentification de son possesseur. L'utilisation de ce type d'adresses est normalement restreinte à un niveau local lors de son emploi avec le mécanisme SEND (cf. Section 2.3). Etendre l'usage des adresses CGA et ses propriétés en sécurité à un niveau plus large pourrait être intéressant. En particulier, ces adresses pourraient être utilisées comme matériel de sécurité alternatif pour IKEv2.

Dans ce chapitre, tout d'abord, nous rappelons ce qu'est IPsec ainsi que IKEv2. Ensuite, nous présentons les avantages et limitations de combiner les adresses CGA avec IKEv2, en illustrant nos propos avec un exemple d'utilisation, la sécurisation de la RO pour MIPv6 (cf. Section 1.6). Finalement, notre solution d'intégration des adresses CGA dans IKEv2 et une implémentation permettant de la valider sont décrites.

4.1 Suite de protocoles IPsec

Internet Protocol security (**IPsec**) [KS05] est une suite de protocoles permettant de sécuriser les communications IP en fournissant authentification et/ou chiffrement. IPsec, fortement recommandé dans toute implémentation de la pile IPv6 [JLN11], est largement déployé aujourd'hui pour IPv4. Même si IPsec peut être manuellement configuré sur les nœuds IP, il est fortement recommandé d'utiliser *Internet Key Exchange version 2* (**IKEv2**) [KHNE10] afin de faciliter son déploiement et utilisation.

4.1.1 Sécurité des paquets IP avec AH et ESP

Les 2 principaux protocoles utilisés par IPsec sont *IP Authentication Header* (**AH**) [Ken05a] et *IP Encapsulating Security Payload* (**ESP**) [Ken05b]. AH et ESP fournissent authentification et intégrité des données. ESP permet en plus de chiffrer ces données. Deux modes d'utilisation sont spécifiés pour ces protocoles : le mode *transport* et le mode *tunnel*. Dans le mode *transport*, la politique IPsec est appliquée directement sur le paquet IP. Dans le mode *tunnel*, le paquet IP originel est encapsulé dans un autre paquet IP sur lequel la politique IPsec est appliquée.

La politique IPsec, configurée dans la base de donnée nommée *Security Policy Database* (**SPD**), est basée sur des règles spécifiant la démarche à effectuer pour un paquet IP spécifique. Cela peut consister à laisser passer le paquet, rejeter le paquet ou appliquer IPsec. Dans ce dernier cas, par exemple, une règle IPsec pourrait être que pour tout paquet IP provenant d'un nœud `A` à destination d'un nœud `B`, utilisant le port 80, il faudrait appliquer IPsec, ESP en mode *transport*. Les nœuds IP sont principalement identifiés par leur adresse IP ou par leur nom de domaine complètement qualifié (e.g., `foo.example.org`), *Fully Qualified Domain Name* (**FQDN**). Il est à noter que d'autres types d'identifiant existent. L'utilisation du FQDN d'un nœud IP est généralement préférée car une adresse IP n'est pas toujours la plus adaptée. Tout d'abord, parce qu'une adresse IP peut avoir seulement été allouée temporairement à un nœud IP, ce qui entraînerait une mise à jour de la SPD à chaque fois que le nœud IP se voit attribuer une nouvelle adresse IP. D'autre part, les adresses IPv6 ne sont pas faciles à manipuler car elles sont plus longues que les adresses IPv4 et sont représentées en code hexadécimal.

Une association de sécurité IPsec, *Security Association* (**SA**) ou *Child SA* [KHNE10], décrit comment une règle IPsec de la SPD est appliquée : par exemple, du nœud `A` vers le nœud `B`, appliquer ESP en mode *transport*, avec l'algorithme 3DES-CBC [PA98], en utilisant la clé `Secret` pour le chiffrement, etc.. Une SA est unidirectionnelle et donc il est nécessaire d'en avoir 2 pour sécuriser toutes les communications entre 2 nœuds IP (i.e., une SA pour une direction et une autre SA pour la direction inverse). La base de données *Security Association Database* (**SAD**) contient toutes les SA. Une SA peut être configurée manuellement ou dynamiquement (e.g., avec IKEv2). Cette dernière méthode permet à AH et ESP de fournir une protection supplémentaire d'anti-rejeu aux paquets sécurisés avec IPsec.

La dernière base de données IPsec est la *Peer Authorization Database* (**PAD**). Elle est utilisée par les mécanismes établissant dynamiquement les SA comme IKEv2. La PAD contient les informations concernant les méthodes d'authentification autorisées ainsi que le type et la valeur de l'identifiant du nœud IP désirant utiliser IPsec. Par exemple, pour le nœud de FQDN `foo.example.org`, une clé secrète partagée doit être utilisée.

4.1.2 Gestion dynamique de la sécurité avec IKEv2

Internet Key Exchange version 2 (**IKEv2**) [KHNE10] est un protocole qui permet de négocier dynamiquement les SA pour IPsec. Cette version du protocole rend caduc IKE [HC98], sa première version.

Dans IKEv2, la communication entre 2 nœuds IP, nommés respectivement *Initiator* et *Responder*, est composée par un ensemble de *IKEv2 Exchanges*. Un *IKEv2 Exchange* est la combinaison d'une requête et d'une réponse entre *Initiator* et *Responder*. Comme illustré dans la Figure 4.1, le premier *IKEv2 Exchange* est *IKE_SA_INIT* et le second est *IKE_AUTH*. Les suivants, optionnels, sont *CREATE_CHILD_SA* et *INFORMATIONAL*.

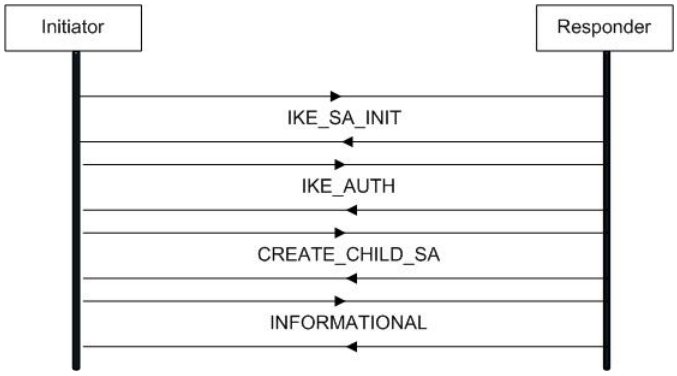

Figure 4.1 – Echanges IKEv2

Les différents *IKEv2 Exchanges* sont :
- IKE_SA_INIT, où les paramètres de sécurité, *nonces*, valeurs *Diffie-Hellman* sont échangés et la SA IKE, fournissant l'intégrité et le chiffrement des données pour les prochains échanges, est négociée puis mise en place ;
- IKE_AUTH, où les identifiants de chaque nœud sont échangés, chaque nœud est authentifié et la première SA IPsec est négociée et mise en place ;
- CREATE_CHILD, quand une nouvelle SA IPsec est créée ;
- INFORMATIONAL, quand une SA est supprimée, une erreur notifiée ou un lien réseau testé.

IKE_SA_INIT

Cet échange initialise les paramètres de confiance, négocie et met en place une SA IKE entre *Initiator* et *Responder*.

Figure 4.2 – Echange IKE_SA_INIT

Différentes informations sont échangées durant *IKE_SA_INIT*, comme illustré dans la Figure 4.2. Le *Security Parameter Index* (**SPI**), dans le *payload IKEv2 Header* `HDR`, permet d'identifier la SA IKE. Le *payload* `SAi1` contient les propositions d'*Initiator* pour la SA IKE, qui va sécuriser les *IKEv2 Exchanges* suivants, et le *payload* `SAr1` la SA IKE sélectionnée par *Responder*. Les *payloads* `KEi` et `KEr` contiennent respectivement la valeur *Diffie-Hellman* d'*Initiator* et de *Responder*. De manière identique, le *nonce* d'*Initiator* et de *Responder* sont respectivement dans les *payloads* `Ni` et `Nr`.

Facultativement, *Responder* peut demander un certificat à *Initiator* grâce au *payload* `CERTREQ`.

A la fin de cet *IKEv2 Exchange*, à partir des *nonces* échangés et du résultat de *Diffie-Hellman*, les 2 nœuds peuvent générer une clé commune, nommée *SKEYSEED*. Par la suite, toutes les futures clés utilisées par IKEv2 seront dérivées de *SKEYSEED* : la clé de

chiffrement *SK_e*, la clé d'authentification (i.e., protection de l'intégrité) *SK_a* et la clé *SK_d* qui sert à générer de nouvelles clés pour les SA IPsec.

IKE_AUTH

Durant cet échange, protégé par la SA IKE, chaque nœud est authentifié et la première IPsec SA est négociée puis mise en place.

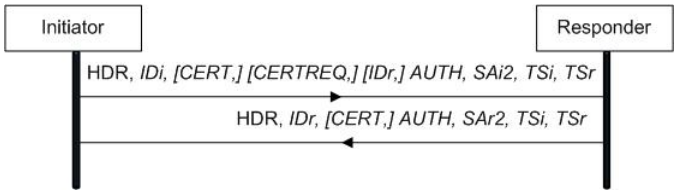

Figure 4.3 – Echange IKE_AUTH

Les différentes informations échangées durant *IKE_AUTH* sont illustrées dans la Figure 4.3. Le SPI dans le *payload IKEv2 Header* `HDR` permet d'identifier la SA IPsec. Le *payload* `SAi2` contient les propositions d'*Initiator* pour la SA IPsec et le *payload* `SAr2` la SA IKE sélectionnée par *Responder*. Le *payload* `AUTH` contient les données nécessaires pour vérifier l'intégrité du message ainsi que l'authentification de son émetteur. Les identifiants d'*Initiator* et *Responder* sont respectivement dans les payloads `IDi` et `IDr`.

Facultativement, *Initiator* peut demander à *Responder* de lui fournir un certificat grâce au *payload* `CERTREQ`. Les certificats échangés entre les 2 nœuds sont inclus dans des *payloads* `CERT`. Le champ *Certificate Encoding* des *payloads* `CERT` et `CERTREQ` indique le type du certificat (e.g., certificat X.509).

Les *payloads* en italique, dans la Figure 4.3, sont chiffrés avec *SK_e*. L'intégrité est fournie grâce à *SK_a*. Un nœud, dont son identifiant est dans `IDi` pour *Initiator* et `IDr` pour *Responder*, prouve son identité avec les données localisées dans le *payload* `AUTH`.

4.2 Inconvénients des méthodes d'authentification actuelles de IKEv2

L'authentification d'un nœud dans IKEv2 est basée classiquement sur une des méthodes de sécurité suivantes : clé secrète partagée, certificats X.509 [CSF+08] et *Extensible Authentication Protocol* (**EAP**) [ABV+04]. Il est important de mentioner que le support d'EAP n'est pas obligatoire pour les implémentations de IKEv2.

Malheureusement, chacune de ces méthodes possède des inconvénients. Tout d'abord, la fourniture de clés secrètes partagées est complexe à mettre en place et inadaptée à grande échelle. Les certificats X.509 peuvent obliger à déployer une architecture dédiée, comme une *Public Key Infrastructure* (**PKI**), entraînant un surcoût et complexe à administrer. Enfin, l'intégration d'EAP n'est pas obligatoire dans les implémentations de IKEv2, ce qui peut conduire à des problèmes d'interopérabilité entre des nœuds désirant utiliser EAP avec IKEv2.

Au passage, il existe d'autres méthodes d'authentification pour IKEv2. Mais, comme EAP, elles ne sont pas obligatoires dans les implémentations de IKEv2. La première est de stocker la clé publique, utilisée pour l'authentification en combinaison avec la clé privée associée, dans un *Domain Name Server* (**DNS**) [Ric05]. La clé publique se retrouve ainsi liée à l'identité du nœud, l'identifiant étant ici le FQDN, grâce au *Resource Record* (**RR**)

IPSECKEY (pour plus d'informations concernant l'architecture DNS et les RR, cf. Section 5.1). Mais afin de sécuriser ce lien et de pouvoir garantir l'intégrité de la clé publique ainsi récupérée, il est nécessaire de déployer l'extension DNS *Domain Name Server security* (**DNSSEC**) [AAL+05]. La deuxième méthode permettant "une authentification" pour IKEv2 est connue sous le nom de *Better-Than-Nothing Security* (**BTNS**) [WR08]. Avec cette méthode, il est fait la supposition qu'il n'y a pas de nœud malicieux essayant de faire une attaque du type *Man-in-the-Middle* (**MitM**) durant *IKE_SA_INIT* et ainsi aucune authentification n'est nécessaire durant *IKE_AUTH* ...

4.3 Intérêt d'utiliser les adresses CGA dans l'authentification IKEv2

Comme nous l'avons vu dans la Section 1.2.4, les propriétés cryptographiques des adresses CGA permettent de prouver la possession d'une telle adresse, et ainsi de fournir une authentification de son possesseur. Celles-ci peuvent être réutilisées afin de pallier les inconvénients des méthodes d'authentification actuelles de IKEv2, identifiés dans la Section 4.2.

Un de ces inconvénients, l'existence nécessaire d'une infrastructure de confiance, a entraîné par le passé la non-adoption de IPsec pour sécuriser des services, comme la RO pour MIPv6 (cf. Section 1.6.2). L'utilisation des adresses CGA avec IKEv2 pourrait remédier à cela.

4.3.1 Avantages et limitations

L'utilisation des adresses CGA avec IKEv2 apporte des avantages mais aussi des limitations. Pour chacune de ces limitations, nous proposons des solutions potentielles permettant de les pallier.

Approche sans infrastructure

L'utilisation des adresses CGA n'oblige pas à déployer une infrastructure dédiée. En effet, une adresse CGA est générée par son propriétaire et toutes les informations nécessaires pour vérifier la véracité de cette adresse, et le message associé à celle-ci, sont envoyées directement au destinataire du message.

Il est intéressant de comparer cette propriété avec les certificats auto-signés. En effet, cette famille de certificats nécessitent seulement une paire de clés publique/privée et aucune infrastructure, comme pour les adresses CGA. Maintenant, il est impossible de vérifier si les informations contenues dans ce genre de certificats, en particulier l'adresse IP si elle est incluse, sont correctes ou pas. En effet, il est possible de mettre n'importe quelle information dans ce type de certificat. Par comparaison, une adresse CGA est le résultat d'un algorithme fixé (cf. Section 1.2.4). Il n'est pas possible de générer l'adresse CGA désirée à moins de réussir une attaque par préimage (i.e., pour une valeur h donnée, trouver un message m tel que h soit le résultat de la fonction de hachage appliquée à m). Ou alors, avec de la chance, un attaquant devra trouver une collision afin de pouvoir modifier des informations liées à cette adresse. Ceci est plus complexe que falsifier un certificat auto-signé et donc on peut considérer les adresses CGA comme plus sûres que cette famille de certificats.

De plus, le destinataire d'un message sécurisé par une adresse CGA a obtenu tout le matériel nécessaire à sa vérification contrairement à l'utilisation d'un certificat X.509. En

effet, pour vérifier un certificat X.509, il est nécessaire d'avoir un chemin de certification jusqu'à un certificat de confiance. Or, ceci nécessite généralement le déploiement d'une *Public Key Infrastructure* (**PKI**). Comme illustré dans la Figure 4.4, une PKI comprend plusieurs entités comme la CA ou la *Registration Authority* - dans certaines architectures de déploiement, ces deux entités sont co-localisées. Les entités constituant la PKI et les messages échangés entre elles peuvent introduire de potentielles failles de sécurité. Si, l'une d'entre elles était correctement exploitée, comme cela a été déjà le cas il n'y a pas si longtemps [1] [2], toute la chaîne de confiance serait brisée et le niveau de sécurité serait identique à celle avec une utilisation de certificats auto-signés.

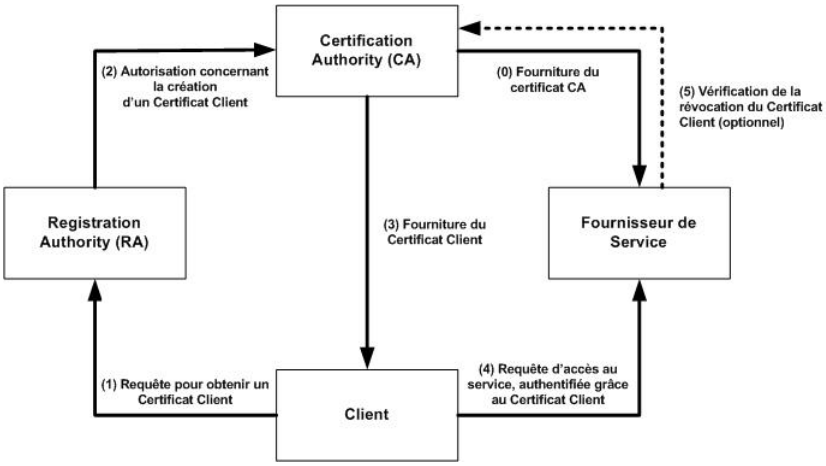

Figure 4.4 – Architecture d'une PKI

Niveau de sécurité

Les adresses CGA ont le même niveau de sécurité que des certificats X.509 lors d'un usage avec IKEv2. En effet, IKEv2 requiert 2 vérifications avec des certificats utilisés pour l'authentification :

1. le certificat doit être valide (i.e., lien sécurisé entre l'identité et la clé publique dans le certificat)
2. le certificat doit être utilisé seulement par son propriétaire

Pour le premier point, IKEv2 vérifie qu'il existe un chemin de certification jusqu'à un certificat de confiance (i.e., généralement un certificat émis par une autorité de certification, *Certification Authority*) (**CA**). Si cela est possible, IKEv2 peut aussi vérifier que le certificat utilisé n'a pas été révoqué. Concernant le deuxième point, IKEv2 vérifie que la signature, contenue dans le *payload* `AUTH`, a bien été effectuée avec la clé privée associée à la clé publique du certificat utilisé.

En comparant avec l'utilisation des adresses CGA, il y a les mêmes 2 vérifications. Pour le premier point, IKEv2 vérifie que l'adresse CGA est valide, à savoir qu'il y a un lien sécurisé entre l'identité, l'adresse CGA, et la clé publique, incluse dans les *CGA Parameters*. Pour cela, IKEv2 vérifie qu'il peut regénérer l'adresse CGA à partir de ces éléments (cf. Section 1.2.4). Pour le deuxième point, IKEv2 vérifie que la signature dans

1. http://isc.sans.org/diary/DigiNotar+breach+-+the+story+so+far/11500
2. http://isc.sans.edu/diary.html?storyid=10603

le *payload* `AUTH` a été générée avec la clé privée associée à la clé publique incluse aux *CGA Parameters* liés à l'adresse CGA (cf. Section 4.5).

Identification

Les adresses CGA sont avant tout des adresses IPv6. Celles-ci sont difficiles à mémoriser pour un humain car elles sont plus longues que les adresses IPv4 (i.e., passage de 32 bits à 128 bits) mais surtout sont représentées en code hexadecimal. De plus, le destinataire d'un message, émis à partir d'une adresse CGA, ne connaît généralement pas qui est l'expéditeur derrière cette adresse. En effet, les adresses CGA peuvent être utilisées pour rendre anonymes les connexions IPv6 en s'en attribuant une nouvelle fréquemment : il n'y a aucun lien entre l'adresse MAC d'une interface et l'IID d'une adresse CGA, rendant aussi les communications d'un même nœud intraçables.

Afin de pallier ce problème d'identification, logiquement, la première idée serait d'associer une adresse CGA à un FQDN et de l'enregistrer dans un serveur DNS. Cependant, cela introduirait du coup les failles de sécurité inhérentes à l'architecture DNS, en particulier au niveau des échanges DNS, comme l'attaque *DNS Cache poisoning* [AA04]. Le niveau de sécurité apporté par l'utilisation d'adresses CGA serait alors perdu.

Pour solutionner cette perte de sécurité, nous avons besoin que les mises à jour DNS ainsi que les résolutions DNS soient sécurisées. Les mises à jour DNS permettent à un nœud IPv6, propriétaire d'une adresse CGA, d'enregistrer dans un serveur DNS l'association de celle-ci avec son FQDN. Les résolutions DNS sont utilisées par un nœud IPv6 afin de récupérer l'adresse IPv6, et donc ici l'adresse CGA, associée à un FQDN spécifique. Afin de sécuriser les mises à jour DNS, il est possible d'utiliser TSIG [VGrW00] (cf. Section 5.2.3) et SIG(0) [3rd00] (cf. Section 5.2.4). Cependant, ces protocoles souffrent d'inconvénients comme nous le verrons plus tard (cf. Chapitre 5). Les résolutions DNS peuvent être sécurisées grâce au déploiement de *Domain Name Server security* (**DNSSEC**) [AAL+05]. Avec de tels outils de protection mis en place, il ne reste comme option à un attaquant potentiel que de trouver une collision au niveau de l'adresse CGA.

Il est à noter que l'utilisation de CGA combinée avec DNSSEC sera meilleure que la solution se basant sur le RR *IPSECKEY* combinée avec DNSSEC pour un serveur DNS. En effet, cette dernière solution augmente la taille des informations à signer dans le DNS et par conséquence le temps nécessaire à cette signature. Ce qui est en particulier critique quand des informations sont mises à jour fréquemment dans le serveur DNS (i.e., nécessitant une nouvelle signature).

Algorithmes cryptographiques

Les spécifications des adresses CGA [Aur05] n'autorisent que l'utilisation de deux algorithmes cryptographiques : l'algorithme à clé publique RSA [RSA78] et la fonction de hachage SHA-1 [iost02].

Concernant l'algorithme à clé publique RSA, il est connu pour être coûteux d'un point de vue ressources et temps de calcul (i.e., contraintes au niveau du processeur et de la batterie). Aussi, l'algorithme RSA n'est pas réellement adapté pour des nœuds IPv6 à faibles capacités, comme les terminaux mobiles ou les capteurs par exemple. Malheureusement, les spécifications actuelles des adresses CGA n'autorisent pas l'usage d'algorithmes alternatifs comme ceux basés sur les courbes elliptiques, *Elliptic Curve Cryptography* (**ECC**), malgré la publication d'articles [CBL10] ou de propositions de standards à l'IETF [SXZ11].

Concernant la fonction de hachage SHA-1, en se basant sur les dernières cryptanalyses rendues publiques, la communauté en sécurité s'attend à ce que cette fonction de hachage

soit "cassée" d'ici un futur proche et que, du coup, il soit facile de générer des collisions. Aussi, la communauté en sécurité est actuellement en train de travailler sur la prochaine génération de fonction de hachage : SHA-3. Mais, encore une fois, les spécifications actuelles des adresses CGA n'autorisent pas l'usage d'algorithmes alternatifs pour le moment.

Enfin, il est important de mentionner que l'usage d'algorithmes cryptographiques peut être réglementé dans certains pays ou entreprises. C'est par exemple le cas en Russie, où l'algorithme GOST R 34.11-94 [Dol10] doit être obligatoirement utilisé en lieu et place de toute autre fonction de hachage.

Révocation

Une adresse CGA fournit une relation forte entre une adresse IPv6 et son propriétaire mais rien n'empêche un nœud malicieux d'être capable d'utiliser cette même adresse dans le futur dans le cas où elle aurait été compromise grâce à une collision. Les certificats X.509 peuvent être révoqués lorsqu'ils sont combinés avec une PKI et que le mécanisme *Certificate Revocation List* (**CRL**) [CSF+08] ou le mécanisme *Online Certificate Status Protocol* (**OCSP**) [MT07] a été déployé. A l'inverse, une adresse CGA ne repose pas sur une infrastructure et donc ne peut être révoquée.

Une solution potentielle pour pallier ce problème serait d'utiliser à nouveau DNS, combiné à DNSSEC afin de garder le même niveau de sécurité. En effet, si la valeur du champ *Time To Live* (**TTL**), dans le RR *AAAA* (cf. Section 5.1) pour le FQDN associé à l'adresse CGA, était fixée de telle sorte qu'elle soit plus courte que le temps nécessaire à la découverte d'une collision pour cette même adresse CGA, alors cela pourrait limiter l'attaque. Maintenant, cette solution deviendrait inutile dans le cas où l'attaquant essaierait de calculer à l'avance les collisions pour un grand nombre d'adresses CGA.

4.4 Intérêt des adresses CGA avec IKEv2 pour sécuriser la RO dans MIPv6

Comme cela a été présenté précédemment (cf. Section 1.6.2), la sécurisation de la RO pour MIPv6 ne repose pas entièrement sur IPsec. Ceci est principalement dû à une condition imposée : cette sécurisation ne devait pas être dépendante d'une infrastructure. Or, il était nécessaire pour pouvoir utiliser IPsec, et IKEv2, qu'il existe une telle infrastructure qui soit commune au MN et à tout CN potentiel. Pour cette raison, le mécanisme *Return Routability* a été spécifié et adopté comme standard.

Malheureusement, le mécanisme *Return Routability* comporte une vulnérabilité : il est ainsi plus faible d'un point de vue niveau de sécurité que l'utilisation de IPsec et IKEv2. En effet, comme illustré dans la Figure 4.5, si un attaquant arrive à se mettre en interception des messages échangés lors du mécanisme *Return Routability*, messages qui ne sont pas chiffrés, il sera capable alors d'avoir en sa possession toutes les informations nécessaires afin de générer convenablement la clé *Kbm*. Pour rappel, cette clé permet d'authentifier la signalisation envoyée par le MN. Du coup, l'attaquant pourra l'utiliser par la suite afin de se faire passer pour le MN et envoyer des BU falsifiés.

Nous avons proposé, à l'IETF, les spécifications permettant de sécuriser, de manière forte, la signalisation entre un MN et un CN grâce à IPsec et IKEv2 [DC08]. Un des points importants dans celles-ci est qu'il est obligatoire que le CN puisse vérifier que le MN est bien le possesseur de la HoA. Un MN peut le prouver grâce à un certificat X.509, incluant sa HoA, qui sera alors vérifié lors de l'échange *IKE_AUTH* (cf. Section 4.1.2), mais cela nécessiterait alors une infrastructure de confiance commune entre le MN et le CN.

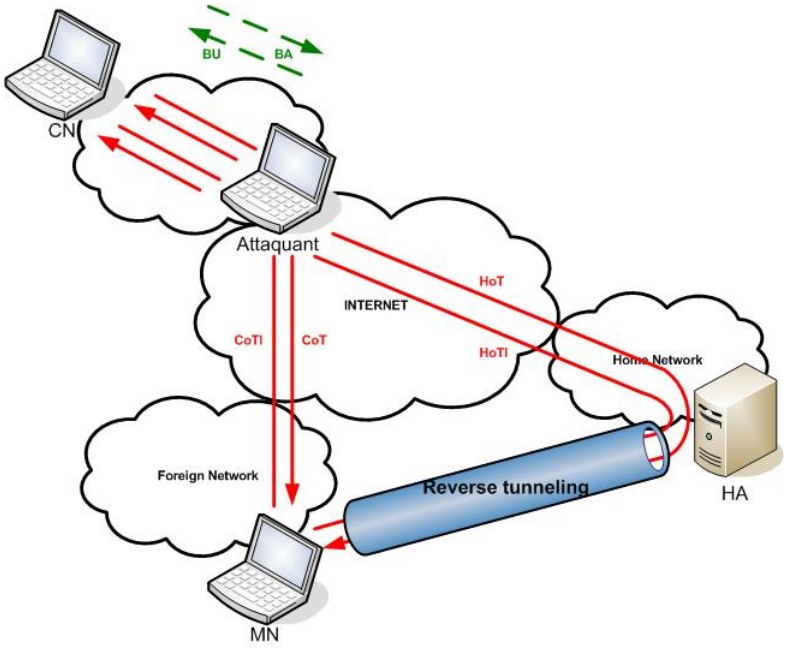

Figure 4.5 – Attaque du mécanisme Return Routability

Une solution, plus simple et remplissant la condition que la sécurisation de la signalisation entre MN et CN ne doit pas reposer sur une infrastructure, est l'utilisation de notre méthode d'authentification avec les adresses CGA dans IKEv2. Dans ce cas là, la HoA du MN doit être une adresse CGA. Ensuite, lors de l'utilisation de IKEv2 avec le CN, configurant les SA IPsec nécessaires à la sécurisation des BU et BA, l'authentification du MN se fait grâce à notre solution (i.e., la méthode d'authentification est basée sur la HoA, qui est une adresse CGA) et le CN peut ainsi vérifier que le MN possède bien la HoA en question.

4.5 Intégration des adresses CGA dans IKEv2

Une approche combinant les adresses CGA avec IKEv2 a initialement été proposée en 2004 dans un article académique [CMLN04] dont l'objet était de décrire une solution de chiffrement opportuniste entre deux passerelles de sécurité.

A la suite de ce travail, une proposition [LMK07], spécifiant comment utiliser les adresses CGA pour n'importe quel scénario IPsec, a été soumise à l'IETF en 2007. Pour notre travail, nous nous sommes inspirés de ce dernier document pour décider de nos choix de modifications dans IKEv2 nécessaires pour l'intégration des adresses CGA [CWL11] [CWL12] mais avec des différences que nous décrivons par la suite (cf. Section 4.5.3).

Pour nos choix, que cela soit pour les *payloads* ou pour les échanges IKEv2, nous avons évité d'en créer de nouveaux et nous avons conservé la nature de ceux que nous avons utilisés pour transporter les informations nécessaires à notre solution. Ceci nous permet de pouvoir intégrer le plus facilement possible notre solution dans toute implémentation existante de IKEv2 conforme aux spécifications [KHNE10].

4.5.1 Modifications au niveau des payloads IKEv2

Tout d'abord, les *payloads* `ID` doivent contenir l'identifiant du nœud qui sera authentifié grâce au *payload* `AUTH` : c'est l'adresse CGA de ce nœud, au format *ID_IPV6_ADDR* [KHNE10].

Les spécifications de IKEv2 décrivent que le *payload* `CERT` peut contenir n'importe quel type de matériel permettant l'authentification. Du coup, nous avons décidé que les *CGA Parameters* seraient inclus et codés en réutilisant le format d'un certificat auto-signé [CSF+08], comme illustré dans la Figure 4.6. De plus, nous avons attribué la valeur 222 à ce nouveau type de matériel d'authentification dans le champ *Certificate Encoding* du *payload* `CERT`.

Le *payload* `CERTREQ` doit permettre d'indiquer que son émetteur souhaite utiliser la méthode d'authentification basée sur les adresses CGA et, du coup, recevoir les *CGA Parameters* du récepteur de la requête. Aussi, nous avons attribué la même valeur (i.e., 222) pour le champ *Certificate Encoding* du *payload* `CERTREQ`.

Finalement, le *payload* `AUTH` contient les données nécessaires pour l'authentification du propriétaire de l'adresse CGA : une signature électronique basée sur la clé privée associée à la clé publique des *CGA Parameters* inclus dans le *payload* `CERT`.

4.5.2 Modifications au niveau des échanges IKEv2

Durant l'échange *IKE_SA_INIT* (cf. Figure 4.2), la seule modification concerne la réponse provenant du *Responder*. Cette dernière doit inclure un *payload* `CERTREQ` avec la valeur 222 afin de notifier qu'il désire authentifier l'*Initiator* en utilisant les adresses CGA.

L'échange IKE_AUTH (cf. Figure 4.3), quand à lui, subit plusieurs modifications. Tout d'abord, les payloads `IDi` et `IDr` doivent contenir respectivement l'adresse CGA de l'*Initiator* et l'adresse CGA du *Responder*. Le *payload* `CERTREQ`, comme précédemment, doit avoir la valeur 222 pour notifier que la méthode d'authentification est basée sur les adresses CGA. Le *payload* `CERT` doit contenir les *CGA Parameters* de l'émetteur du message et avoir 222 comme valeur de type de méthode d'authentification. Et enfin, le *payload* `AUTH` doit contenir la signature basée sur la clé privée associée à la clé publique des *CGA Parameters* contenus dans le *payload* `CERT`.

4.5.3 Comparaison avec les autres solutions existantes

Nos choix concernant les modifications apportées à IKEv2 pour supporter les adresses CGA sont très proches de la proposition à l'IETF [LMK07]. Cependant, il existe plusieurs différences. Tout d'abord, contrairement à la proposition faite à l'IETF, nous avons défini précisément comment l'authentification basée sur les adresses CGA était initiée par IKEv2 (cf. Section 4.6) : cela est configuré dans la PAD (i.e., fichier de configuration IPsec de StrongSwan). Enfin, contrairement à la proposition faite à l'IETF, nous avons spécifié précisément comment les *CGA Parameters* étaient inclus dans le *payload* `CERT` (cf. Section 4.6) : les *CGA Parameters* sont encodés dans un format de type certificat X.509.

Une autre solution de IKEv2 intégrant les adresses CGA a été spécifiée et implémentée[3]. Cependant, les choix concernant les modifications sont vraiment différents des nôtres. Tout d'abord, la solution s'appuie principalement sur IKEv1 [HC98] et semble avoir été récemment adaptée pour IKEv2. Ensuite, autre différence, un nœud annonce qu'il est capable d'utiliser la méthode d'authentification basée sur les adresses CGA en utilisant

3. http://www.msdn.microsoft.com/en-us/library/cc233219(v=prot.10).aspx

un *payload* IKE spécifique, nommé *Vendor ID*. La différence suivante concerne la transmission des *CGA Parameters* : ceux-ci sont encodés dans le *payload* `ID` et un certificat auto-signé, généré à partir de la paire de clés publique/privée associée à l'adresse CGA, est transmis dans le *payload* `CERT`. Ce certificat est seulement employé pour authentifier les échanges IKE. Enfin, dernière différence, la signature des échanges IKE est vérifiée avant la regénération de l'adresse CGA, contrairement à notre solution. Ce point pourrait faire l'objet d'une attaque de type DoS : il suffit d'initier un grand nombre de connexions IKEv2 avec des signatures erronées dans le *payload* `AUTH`. En effet, il est plus coûteux en temps et puissance de calculs de vérifier une signature que de regénérer une adresse CGA. C'est pour cela que IKEv2 vérifie d'abord la validité du certificat (cf. Section 4.3.1).

4.6 Implémentation de notre solution

Afin de vérifier la validité des choix pour notre solution et que celle-ci permettait bien de mettre en place des connexions IPsec, nous l'avons implémentée. Cette implémentation [4] est basée sur 2 composantes déjà existantes :

1. Implémentation CGA
   La librairie CGA provient de l'implémentation de SEND développée par DoCoMo USA Labs en code libre. Nous avons choisi cette librairie car elle était l'une des rares disponible et fonctionnant. Aujourd'hui, cette implémentation n'est plus maintenue.
2. Implémentation IKEv2
   StrongSwan [5] fut sélectionnnée car cette implémentation en code libre de IKEv2 est bien codée, documentée et mise à jour régulièrement.

Pour intégrer directement nos choix dans StrongSwan, plusieurs modifications furent nécessaires. Tout d'abord, le fichier de configuration IPsec (i.e., intégrant la PAD dans StrongSwan) et l'analyseur syntaxique associé (i.e., *parser*) ont été modifiés afin d'accepter des règles basées sur les adresses CGA. Ces adresses CGA doivent maintenant être dans le *payload* `ID` durant les échanges IKEv2. Ensuite, pour demander et fournir des *CGA Parameters*, inclus respectivement dans le *payload* `CERTREQ` et le *payload* `CERT`, la nouvelle valeur attribuée 222 pour l'encodage du type de matériel d'authentification a été spécifiée. Ces *CGA Parameters* sont encodés dans un format de type certificat X.509 [CSF+08], comme illustré dans la Figure 4.6. Plus précisément, dans ce certificat, le champ *Subject* indique que c'est une adresse CGA et le champ *Subject Public Key Info* inclus l'algorithme utilisé, à savoir RSA, et les *CGA Parameters*. Le *parser* par défaut de StrongSwan peut dynamiquement inclure des fonctions via des modules d'extension (i.e., fonction système *dlopen*). Aussi, pour récupérer les *CGA Parameters* à l'intérieur d'un certificat et les transformer dans un format valide pour StrongSwan, un nouveau module d'extension (i.e., *plugin*) a été implémenté. Concernant la gestion du *payload* `AUTH`, maintenant, avant de vérifier la validité de la signature, le *daemon* IKEv2 doit regénérer l'adresse CGA en utilisant les *CGA Parameters* fournis.

Finalement, afin de vérifier visuellement que les informations étaient correctement fournies durant les échanges IKEv2, un module d'extension pour Wireshark [6] a été implémenté, comme illustré dans la Figure 4.7. En effet, à cause de nos choix concernant la solution pour intégrer les adresses CGA dans IKEv2, aucune autre implémentation n'existe pour pouvoir effectuer des tests d'interopérabilité.

4. http://tinyurl.com/d37m36o
5. http://www.strongswan.org/
6. http://www.wireshark.org/

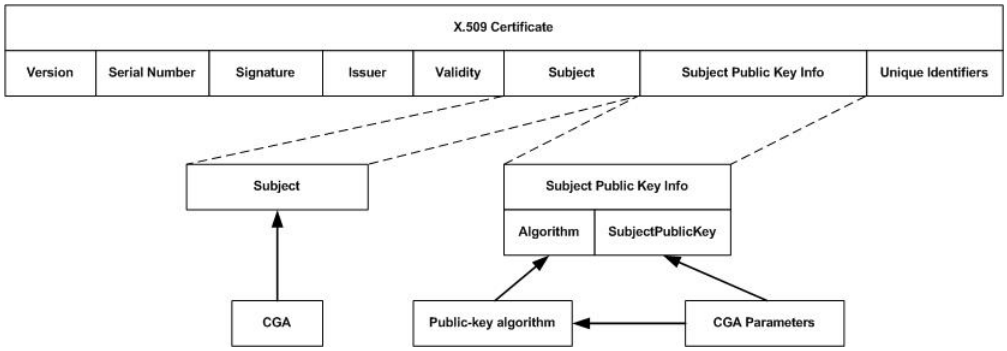

Figure 4.6 – *CGA Parameters* inclus dans un modèle de certificat X.509

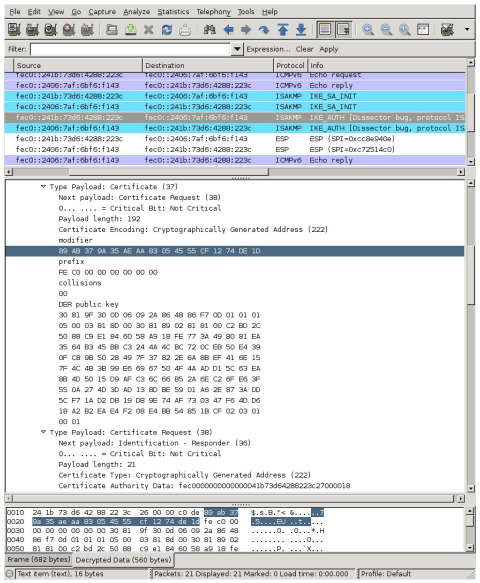

Figure 4.7 – Module d'extension Wireshark pour IKEv2 intégrant les adresses CGA

4.7 Synthèse du chapitre

Dans ce chapitre, nous avons tout d'abord détaillé la famille de protocoles IPsec et IKEv2 qui permettent de sécuriser la couche IPv6 (cf. Section 4.1). Ensuite nous avons décrit les différentes méthodes d'authentification utilisables par IKEv2 et leurs inconvénients respectifs, pouvant entraîner la non-adoption d'IPsec pour sécuriser des services IPv6, comme la RO pour MIPv6 (cf. Section 4.2). Du coup, nous avons présenté comment l'utilisation des adresses CGA comme méthode d'authentification alternative dans IKEv2 pouvait pallier à certains des précédents inconvénients (cf. Section 4.3).

Pour montrer l'intérêt d'une telle utilisation, nous l'avons illustrée en proposant une solution de sécurisation de la signalisation pour la RO dans MIPv6 (cf. Section 4.4). Celle-ci apporte un niveau de sécurité plus élevé que celle actuellement standardisée, qui est vulnérable à une attaque MitM comme nous l'avons montré. De plus, notre solution repose sur IPsec et IKEv2 qui doivent être présents dans chaque nœud IPv6, et donc dans chaque potentiel CN d'un MN. Nous l'avons soumis à l'IETF et elle est toujours en cours de standardisation. Maintenant, son adoption pourrait s'accélérer suite à une demande

croissante de mécanismes sécurisés, de manière forte, permettant le *off-loading* (i.e., flux de données mobiles ne passant plus par le cœur de réseau du fournisseur d'accès Internet).

Ensuite, nous avons détaillé nos choix afin intégrer des adresses CGA dans IKEv2 (cf. Section 4.5). Ceux-ci ne réutilisent que des *payloads* et échanges de IKEv2 existants, ce qui facilite l'intégration de notre solution dans toute implémentation conforme aux spécifications de IKEv2.

Finalement, nous avons implémenté et testé notre solution (cf. Section 4.6). Ceci nous a permis de valider nos choix et surtout de confirmer que les identifiants cryptographiques, les adresses CGA dans ce cas présent, pouvaient constituer un matériel de sécurité alternatif, facilement utilisable, lors de la sécurisation d'un service IPv6.

Chapitre 5

Mise à jour DNS sécurisée à base de CGA et d'IBC

L'architecture *Domain Name System* (**DNS**), déployée largement dans l'Internet, est utilisée pour stocker le lien entre un *Fully Qualified Domain Name* (**FQDN**) d'un nœud IP (e.g., `hostname.example.com`) et des informations relatives à cet FQDN (e.g., l'adresse IP du nœud possédant le FQDN en question). Quand l'adresse IP du nœud change, cette donnée doit, en conséquence, être mise à jour dans le serveur DNS. Cela peut être fait dynamiquement grâce au mécanisme *dynamic DNS update* (**DNS UPDATE**). Afin d'éviter des attaques basées sur l'usurpation, des mécanismes de protection ont été standardisés, comme *Secret Key Transaction Authentication for DNS* (**TSIG**) ou *DNS Request and Transaction Signatures* (**SIG(0)**), mais ils possèdent des inconvénients.

Nous proposons un nouveau mécanisme, permettant de protéger la signalisation DNS UPDATE et palliant aux précédents inconvénients, en utilisant les propriétés cryptographiques des adresses CGA et l'*Identity-Based Cryptography* (**IBC**). La combinaison de ces deux dernières technologies permet de sécuriser de manière forte le lien entre l'identifiant d'un nœud IP (i.e., son FQDN) et son adresse IPv6 : notre solution apporte la preuve de propriété de ces deux informations.

Dans ce chapitre, tout d'abord, nous rappelons l'architecture DNS, les failles de sécurité inhérentes à celle-ci, les mécanismes actuellement existants pour les combler et leurs limites. Ensuite, nous présentons notre solution, en la décomposant en 3 parties. Dans la première, nous décrivons l'utilisation des adresses CGA permettant de prouver la possession de l'adresse IPv6 dans le DNS UPDATE. Dans la deuxième partie, nous rappelons la méthode cryptographique IBC puis nous décrivons son utilisation pour prouver la possession du FQDN. Enfin, nous décrivons, dans la dernière partie, notre solution complète, combinaison des deux précédentes. Ensuite, nous présentons les détails de l'implémentation de notre solution. Finalement, nous donnons une analyse de ses avantages et limitations, tout en proposant des moyens de pallier à ces dernières.

5.1 Architecture DNS et mise à jour dynamique

L'architecture *Domain Name System* (**DNS**) [Moc87b] permet de stocker, pour un nom de domaine spécifique (e.g., `example.com`), le lien entre l'identifiant, *Fully Qualified Domain Name* (**FQDN**), d'un nœud IP et des informations le concernant (e.g., l'adresse IP du nœud ayant cet FQDN). Aussi, par exemple, le serveur DNS, qui est une base de données, nécessite d'être mis à jour quand l'adresse IP, associée à un FQDN, change. Cela est effectué dynamiquement grâce au mécanisme DNS UPDATE. Celui-ci introduit de

potentielles failles de sécurité pour l'architecture DNS. Du coup, des protocoles ont été spécialement standardisés pour contrer de telles failles. Maintenant, ces derniers possédent des limitations, principalement dûes à l'architecture où ils sont déployés.

5.1.1 Mise à jour dynamique du DNS

Le mécanisme *dynamic DNS update* (**DNS UPDATE**) [VTRB97] permet à un nœud IP de modifier ou supprimer des informations d'un serveur DNS. Ce mécanisme utilise le format des messages DNS [Moc87b], basés sur la couche UDP et illustré dans la Figure 5.1. Il emploie de plus la structure permettant de stocker les informations dans un serveur DNS, connue sous le nom de *Resource Record* (**RR**) et illustré dans la Figure 5.2.

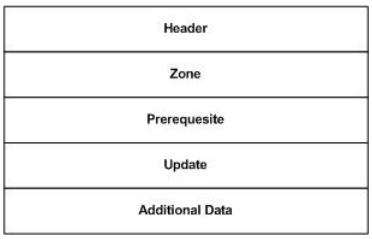

Figure 5.1 – Format d'un message DNS

Un message DNS est identifié comme de type DNS UPDATE grâce au champ *Header*. Le champ *Zone* indique le nom de domaine incluant le FQDN à mettre à jour (e.g., `example.com`). Le champ *Prerequisite* contient les conditions nécessaires à cette mise à jour. Le champ *Update* indique les actions à mettre en œuvre : soit supprimer et/ou ajouter des informations, en utilisant la structure RR pour encoder ces actions et les informations qui sont associées. Finalement, le champ *Additional Data* inclut des informations complémentaires liées au champ *Update*.

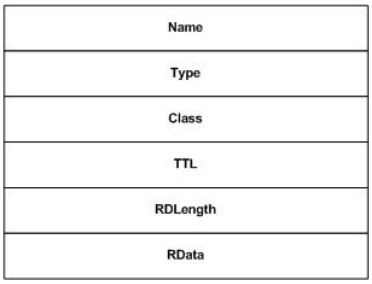

Figure 5.2 – Format d'un RR

Concernant la structure RR, le champ *Name* inclut le FQDN (e.g., `hostname.example.com`). Le champ *Type* indique le type de l'information liée à ce FQDN (e.g., `AAAA` pour une adresse IPv6). Le champ *TTL* inclut le temps de vie de cette information. Le champ *RData* contient l'information (e.g., l'adresse IPv6 `2001:db8::2012/64`) et le champ *RDLength* indique la taille du champ *RData*.

5.2 Vulnérabilités, méthodes de protection et leurs limites

Permettre un nœud IP de mettre à jour, ou supprimer, des données dans un serveur DNS introduit des failles de sécurité. Aussi, des solutions de sécurité ont été standardisées afin de pallier à celles-ci : TSIG, SIG(0) et HTTPS. L'utilisation de ces solutions est fortement dépendante de l'architecture DNS déployée. Du coup, cette dernière peut entrainer des limitations aux solutions de sécurisation, en particulier dans un environnement où le mécanisme SLAAC est mis en œuvre.

5.2.1 Vulnérabilités de la mise à jour DNS dynamique

Quand un nœud IP met à jour des données dans un serveur DNS, deux informations sont critiques : le FQDN et l'adresse IP. En effet, un nœud malicieux pourrait falsifier, soit le FQDN, soit l'adresse IP, soit les deux, lors de la mise à jour de ces données dans le serveur DNS.

Dans le premier cas (i.e., FQDN usurpé), toute les communications à destination du FQDN seront redirigées vers l'adresse IP du nœud malicieux, comme illustré dans la Figure 5.3. Plus précisément, (1) le nœud IP malicieux envoie un message DNS UPDATE au serveur DNS incluant un FQDN usurpé (e.g., `www.mabanque.fr`) associé à son adresse IP. (2) Quand une victime veut contacter le nœud IP associée au FQDN qui a été usurpé par le nœud malicieux, il envoie une requête DNS au serveur DNS afin d'obtenir l'adresse IP associé à ce FQDN et (3) le serveur DNS lui répond en incluant l'adresse IP du nœud malicieux dans sa réponse. (4) Au lieu de contacter le correspondant correct, (5) la victime établit une communication avec le nœud malicieux. Cette attaque ressemble à l'attaque connue sous le nom de *DNS Cache Poisoning* [AA04] mais est plus critique parce que toutes les bases de données DNS (i.e., serveurs DNS maitres/esclaves, caches DNS) sont impactées. De plus, le mécanisme *DNS Security* (**DNSSEC**) [AAL+05] ne peut protéger contre ce type d'attaque. En effet, DNSSEC permet de garantir l'intégrité et l'authenticité des informations fournies par un serveur DNS mais, ici, les données dans ce serveur DNS sont déjà corrompues.

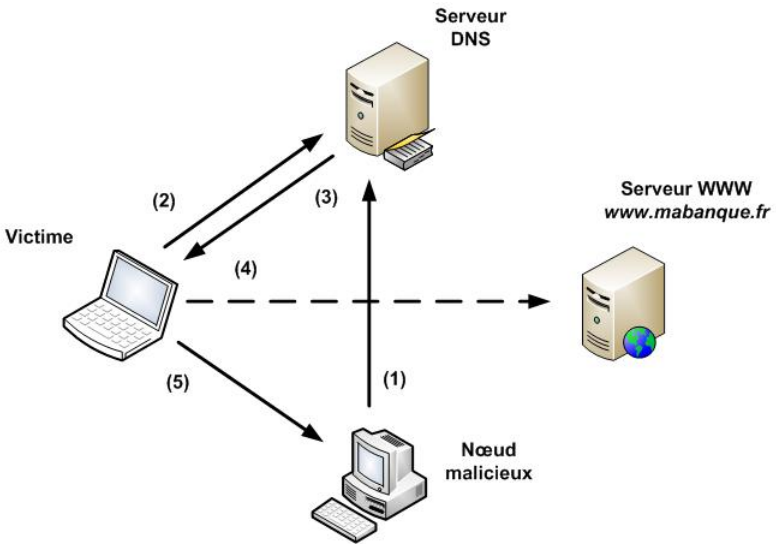

Figure 5.3 – Attaque basée sur un FQDN usurpé

Dans le deuxième cas (i.e., adresse IP usurpée), toutes les communications en direction

d'un FQDN spécifique seront redirigées vers une localisation erronée. L'adresse IP usurpée peut être une adresse non assignée à un terminal IP et du coup la victime ne peut initier une communication avec le nœud IP associé au FQDN en question. Ou alors, ce peut être une adresse IP appartenant à un autre nœud et donc dans ce cas, ce dernier peut potentiellement devenir la cible d'un *Denial of Service* (**DoS**) s'appuyant sur des attaques de type *flooding* (i.e., envoi massif de requêtes vers une même cible), comme illustré dans la Figure 5.4 : (1) le nœud malicieux envoie un message DNS UPDATE, au serveur DNS, comprenant son FQDN associé à une adresse IP usurpée, possédée en réalité par la cible de l'attaque DoS. (2) Quand un nœud IP veut contacter le nœud malicieux, il interroge le serveur DNS pour obtenir l'adresse IP associée au FQDN du nœud malicieux et (3) le serveur DNS lui répond en lui fournissant en fait l'adresse IP de la cible. (4) Au lieu de contacter le nœud malicieux, (5) ce nœud IP essaie d'établir une communication avec la cible. (6) Il en sera de même pour n'importe quel nœud IP désirant contacter le FQDN du nœud malicieux et, s'ils sont nombreux, cela peut entraîner une attaque de type *flooding*. Encore une fois, DNSSEC ne peut protéger contre cette menace : comme expliqué auparavant, les informations enregistrées dans le serveur DNS sont déjà corrompues.

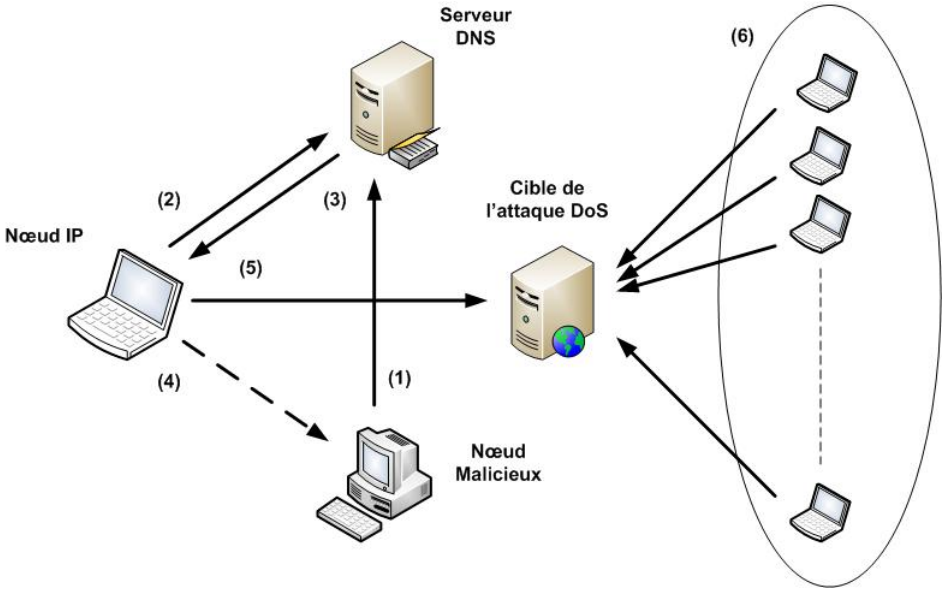

Figure 5.4 – Attaque basée sur une adresse IP usurpée

Naturellement, un attaquant potentiel peut combiner les deux attaques décrites (i.e., adresse IP et FQDN usurpés).

5.2.2 HTTPS

Suite à l'achat d'un nom de domaine, le moyen le plus commun, fourni par un hébergeur de noms de domaine, d'associer une adresse IP à un FQDN, inclus dans ce nom de domaine, s'appuie sur l'utilisation d'une page *web* que le client a à compléter manuellement. La communication des informations est basée, du coup, sur le mécanisme *Hypertext Transfer Protocol* (**HTTP**) [FGM+99]. Elle est généralement protégée grâce au protocole *Transport Layer Security* (**TLS**) [DR08] [Res00]. Le client s'authentifie sur la page *web* grâce à un couple "identifiant/mot de passe", fourni par l'hébergeur. Cela permet à ce dernier de lier le nom de domaine acheté, donc le FQDN mis à jour, au client. Il est à noter que cette

solution de sécurisation n'empêche pas un client malintentionné de fournir une adresse IP usurpée à l'hébergeur lors de la mise à jour. Enfin, le serveur DNS peut être mis à jour manuellement ou dynamiquement, en utilisant les informations fournies par le client. Dans le dernier cas, les mécanismes de sécurité décrits par la suite peuvent être mis en œuvre.

5.2.3 TSIG

Secret Key Transaction Authentication for DNS (**TSIG**) [VGrW00] est un mécanisme d'authentification pour les messages DNS UPDATE basé sur la cryptographie symétrique : le client DNS, envoyant des messages DNS UPDATE, et le serveur DNS partagent une même clé secrète. TSIG utilise les algorithmes *Hash-based Message Authentication Code* (**HMAC**) [KBC97] comme méthodes d'authentification. Les informations de sécurité TSIG sont incluses dans une structure RR (cf. Figure 5.2) nommée TSIG RR et où, le champ *Name* contient le nom de la clé employée avec la méthode d'authentification, le champ *Type* est fixé à la valeur 250, le champ *TTL* est fixé à la valeur 0, le champ *RData* contient l'algorithme HMAC utilisé ainsi que son résultat. Une fois la méthode d'authentification appliquée sur le message DNS UPDATE, le RR TSIG est inclus dans le champ *Additionnal Data* de ce message (cf. Figure 5.1). Le message DNS UPDATE et la réponse du serveur DNS sont tous les deux sécurisés avec TSIG.

5.2.4 SIG(0)

DNS Request and Transaction Signatures (**SIG(0)**) [3rd00] est un mécanisme d'authentification pour les messages DNS UPDATE aussi mais, contrairement à TSIG, il repose sur la cryptographie asymétrique : le serveur DNS connaît la clé publique du client DNS, enregistrée dans un KEY RR du fichier de configuration du nom de domaine, et le client DNS utilise la clé privée associée pour sécuriser les messages DNS UPDATE qu'il envoie. SIG(0) peut employer différentes méthodes d'authentification (e.g., Diffie-Hellman [3rd99c], RSA/SHA-1 [3rd01], DSA [3rd99b]). Comme TSIG, les informations de sécurité de SIG(0) sont incluses dans une structure RR (cf. Figure 5.2) nommée SIG RR et où, le champ *Type* est fixé à la valeur 24, le champ *TTL* est fixé à la valeur 0, le champ *RData* contient le FQDN de l'émetteur, la méthode d'authentification employée pour la génération de la signature et son résultat. Une fois la méthode d'authentification appliquée sur le message DNS UPDATE, le RR SIG est inclus dans le champ *Additionnal Data* de ce message (cf. Figure 5.1).

5.2.5 Limitations

La solution basée sur HTTPS (cf. Section SEC :HTTPS) a plusieurs inconvénients. Tout d'abord, elle nécessite des actions manuelles. En effet, le propriétaire du FQDN doit remplir les informations de mise à jour sur la page *web* de l'hébergeur de noms de domaine, et cela, à chaque fois que son adresse IP change. Aussi, cette solution n'est pas vraiment adaptée dans les cas où, soit l'adresse IP change trop fréquemment, soit un seul administrateur gère un grand nombre de nœuds IP. Enfin, comme mentionné précédemment, cette solution ne permet pas d'éviter qu'un propriétaire malintentionné de FQDN effectue une mise à jour avec une adresse IP falsifiée.

Aujourd'hui, dans la plupart des réseaux IPv4 appartenant à des entreprises ou à des fournisseurs d'accès Internet, le scénario classique, comme illustré dans le Figure 5.5, consiste à ce que le serveur DHCP [Dro97], (1) qui alloue une adresse IP à un nœud IP, (2) mette à jour aussi les informations dans le serveur DNS (i.e., lien entre le FQDN

du nœud IP et l'adresse IP allouée à ce nœud). Le serveur DHCP est aussi client DNS. Concernant la sécurité entre le serveur DHCP et le serveur DNS, soit aucune sécurité n'est mise en place car les serveurs sont dans un même domaine administratif et la sécurité repose essentiellement sur une confiance de l'infrastructure, soit TSIG est employé car TSIG est plus simple à déployer (i.e., il est nécessaire de configurer seulement une clé secrète partagée entre le serveur DHCP et le serveur DNS) et moins coûteux en terme de puissance de calcul que SIG(0) (car mise en œuvre avec la cryptographie symétrique). Du coup, d'après nos connaissances actuelles, SIG(0) n'est pas vraiment déployé et TSIG est généralement employé.

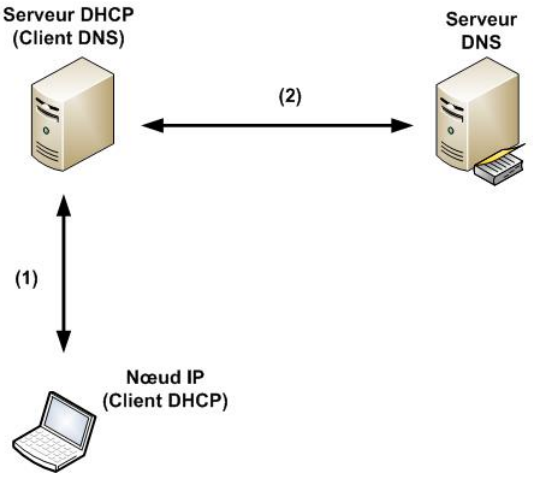

Figure 5.5 – Utilisation de DHCP dans des scénarios IPv4

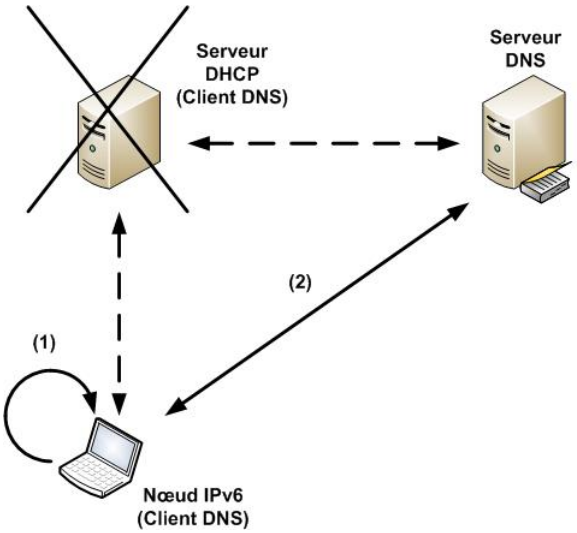

Figure 5.6 – Scénario d'autoconfiguration IPv6

Dans un environnement IPv6, un serveur DHCPv6 [DBV+03] peut devenir inutile quand, comme illustré dans la Figure 5.6, (1) les nœuds IPv6 peuvent autoconfigurer leurs propres adresses IPv6, comme présenté dans la Section 1.4. Du coup, (2) ces nœuds devront mettre à jour eux-mêmes les informations dans le serveur DNS. Par conséquent, le serveur

DNS a besoin de vérifier que les informations (i.e., FQDN et adresse IPv6), fournies par un nœud IPv6, sont correctes et sont bien possédées par celui-ci. L'utilisation de TSIG n'est pas adaptée ici car il serait nécessaire de configurer une clé secrète partagée sur le serveur DNS pour chaque nœud IPv6 et ceci n'est pas vraiment réalisable à grande échelle (e.g., nombre important de nœuds IPv6 chez un fournisseur d'accès Internet). Aussi, SIG(0) reste la seule option mais nécessite de configurer, sur le serveur DNS, une clé publique pour chaque nœud IPv6. De plus, SIG(0) ne solutionne pas le problème de possession d'adresse IPv6.

5.3 Nouvelle méthode de sécurisation des messages de mise à jour DNS à base d'identifiants cryptographiques

Nous proposons une nouvelle solution de sécurisation des messages de mise à jour DNS [CAL12], alternative à celles présentées précédemment. Celle-ci permet de solutionner les deux problèmes suivants : la possession d'une adresse IPv6 et la possession d'un FQDN. Naturellement, ces informations ne doivent pas être modifiées durant les échanges entre le client DNS et le serveur DNS, aussi notre solution fournit les mêmes propriétés de sécurité que TSIG et SIG(0) (i.e., authentification et intégrité). Pour faciliter un déploiement à grande échelle de notre solution, nous avons basé notre solution sur la cryptographie asymétrique et avons décidé de réutiliser SIG(0). Ceci nous permet, d'une part, de garder les propriétés de sécurité fournissant l'authentification et l'intégrité des messages DNS UPDATE. D'autre part, l'intégration de notre solution n'aura nécessité que quelques modifications mineures dans les implémentations actuelles de SIG(0).

Notre solution comprend deux parties : la première partie permet de prouver la possession d'une adresse IPv6 et la deuxième, la possession d'un FQDN.

5.3.1 Preuve de possession de l'adresse IPv6 grâce à CGA

Pour cette problématique, nous avons décidé de réutiliser les propriétés cryptographiques des adresses CGA, dont nous avons fait la présentation dans la Section 1.2.4. Nous avons spécifié un nouveau type de RR afin de transporter les paramètres CGA associés à ces adresses et apporté une modification mineure à SIG RR.

Utilisation de CGA

La preuve de possession d'une adresse CGA, combinée à un message DNS UPDATE, permet de certifier que l'émetteur de ce message est bien le propriétaire de l'adresse (i.e., l'adresse CGA) qu'il va mettre à jour dans le serveur DNS. Comme illustré dans la Figure 5.7, (1) le nœud IPv6 génère son adresse CGA (et donc une paire de clés publique/privée, $K_{CGApub}/K_{CGApriv}$). Ensuite, (2) il envoie un message DNS UPDATE, en y intégrant les *CGA Parameters* et en le signant avec $K_{CGApriv}$. Enfin, (3) le serveur DNS vérifie la possession de l'adresse CGA (i.e., regénération de l'adresse CGA et contrôle de la signature) et met à jour les informations concernant le nœud IPv6 si la vérification est correcte.

CGAparam RR

Afin de transporter les *CGA Parameters*, nécessaires au serveur DNS pour vérifier la possession de l'adresse CGA, nous avons spécifié un nouveau type de RR et qui est inclus

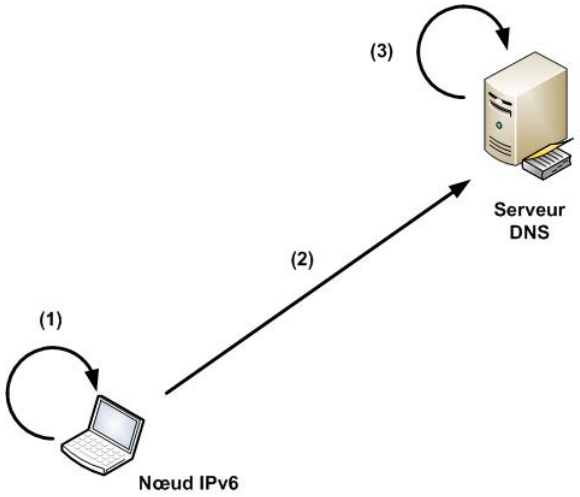

Figure 5.7 – Scénario d'utilisation de l'adresse CGA dans DNS

dans le message DNS UPDATE. Ce nouveau type de RR est nommé *CGAparams* et sa structure ressemble à celle des *CGA Parameters*, comme illustré dans la Figure 5.8.

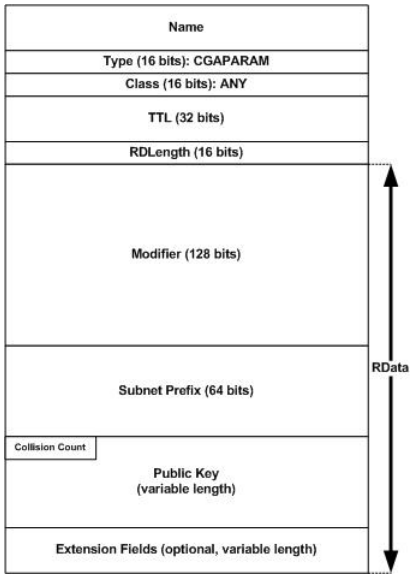

Figure 5.8 – Format de *CGAparams* RR

Modification de SIG(0)

Afin d'indiquer notre nouvelle méthode d'authentification basée sur les adresses CGA, nous avons modifié SIG(0) en définissant une nouvelle valeur pour le champ *Algorithm* inclus dans le champ *RData* de SIG RR. Quand cette nouvelle valeur est indiquée, le processus SIG(0) doit procéder à une vérification CGA (i.e., regénération de l'adresse CGA suivie du contrôle de la signature).

5.3.2 Preuve de possession du FQDN grâce à l'IBC

Pour cette problématique, nous avons décidé d'utiliser les propriétés cryptographiques de l'Identity-Based Cryptography. Nous avons aussi apporté une modification mineure à SIG RR. Enfin, nous avons spécifié un mécanisme, simple car celui-ci n'était pas l'objectif principal de nos travaux mais il était nécessaire pour nos tests, permettant d'échanger le matériel de sécurité entre les différentes entités de notre solution.

Identity Based Cryptogaphy

L'*Identity-Based Cryptography* (**IBC**) [Sha85] utilise des schémas cryptographiques asymétriques (i.e., utilisation d'une paire de clés publique/privée) où les clés publiques sont dérivées via une procédure spécifique. En effet, alors qu'habituellement la clé publique est dérivée d'une clé privée générée "aléatoirement", avec les schémas IBC, la clé publique est dérivée de manière déterministe à partir d'un identifiant choisi par son utilisateur (e.g., l'adresse email `example@foo.com`). La clé privée associée est générée par une entité nommée *Private Key Generator* (**PKG**). Le PKG est configuré avec un secret "Maître" servant à la génération d'une clé privée associée à un identifiant. Le PKG doit fournir, à l'émetteur et/ou le récepteur, les paramètres PKG publiques, *public PKG parameters*, utilisés lors des différents calculs cryptographiques. L'article académique [BF01] fournit les paramètres PKG publiques à utiliser avec un exemple d'algorithme ECC [Kob87] [Mil]. L'IBC peut être employé autant pour le chiffrement, cas le plus répandu, que pour la signature de données.

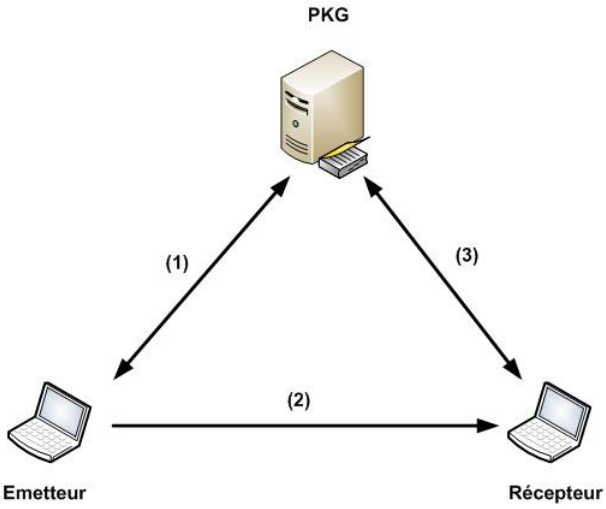

Figure 5.9 – Architecture et échanges IBC

Dans le cas d'une opération de chiffrement, appelée aussi *Identity-Based Encryption* (**IBE**), comme illustré dans la Figure 5.9, (1) l'émetteur obtient les *public PKG parameters* du PKG, qu'il peut enregistrer en cas de nouveau chiffrement de données. (2) En les utilisant, il génère l'identifiant du récepteur (i.e., la clé publique), chiffre grâce à cette dernière les données et les envoie. (3) Le récepteur récupère auprès du PKG les *public PKG parameters* ainsi que sa propre clé privée, celle associée à son identifiant. Il peut enregistrer cette clé pour tout déchiffrement futur lié à cet identifiant, et déchiffre les données reçues.

Dans le cas de génération d'une signature, comme illustré dans la Figure 5.9, (1)

l'émetteur obtient du PKG les *public PKG parameters* ainsi que la clé privée associée à l'identifiant, qu'il peut enregistrer en cas de nouvelle signature liée à cet identifiant. (2) A partir de ces éléments, il signe les données et les envoie. (3) Le récepteur obtient du PKG les *public PKG parameters*, qu'il peut enregistrer pour de futures vérifications de signature ; il génère l'identifiant de l'émetteur et contrôle la signature des données. Par la suite, nous ne ferons que usage de la signature.

Utilisation de l'IBC

Lors de la sécurisation du message DNS UPDATE, nous avons décidé d'utiliser l'IBC, à la place du KEY RR [3rd00], pour contrôler que l'émetteur du message est bien le propriétaire du FQDN à mettre à jour dans le serveur DNS. Ainsi, en reprenant la procédure de signature IBC présentée auparavant, comme illustré dans la Figure 5.10, (1) le nœud IPv6 obtient du PKG les *public PKG parameters*. (2) Le nœud IPv6 génère alors son identifiant (i.e., sa clé publique $K_{FQDNpub}$), qui est en fait son FQDN, et obtient du PKG la clé privée associée $K_{FQDNpriv}$. (3) Grâce à cette dernière, le nœud IPv6 signe le message DNS UPDATE et l'envoie au serveur DNS. (4) Le serveur DNS obtient du PKG les *public PKG parameters*, vérifie la validité de la signature et, si la vérification est correcte, met à jour les informations relatives à ce nœud IPv6.

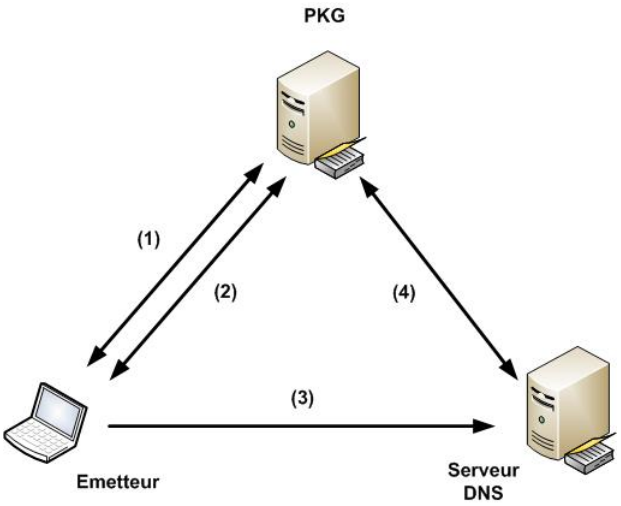

Figure 5.10 – Scénario d'utilisation d'IBC dans DNS

Modification de SIG(0)

La modification dans SIG(0) est relativement mineure et consiste à définir une nouvelle valeur dans le SIG RR, pour le champ *Algorithm*, indiquant que la méthode d'authentification repose sur l'IBC. Lorsque que cette valeur est indiquée, le processus SIG(0) doit obtenir les *public PKG parameters*, si jamais le serveur DNS ne les connaissait pas déjà, et vérifier la signature avec l'identifiant (i.e., le FQDN) comme clé publique.

Transport du matériel de sécurité

Pour nos tests, nous avons dû spécifier comment le PKG communique avec les différentes entités afin de s'échanger le matériel de sécurité nécessaire à l'IBC. Aussi, nous

avons défini 3 types de message sur UDP [Pos80] : un message *Public PKG parameters Request*, permettant de réclamer les *public PKG parameters*, un message *Private Key Request*, permettant de réclamer une clé privée associée à un identifiant spécifique, et enfin un message *PKG Reply* permettant au PKG de fournir les informations demandées. Nous supposons que l'échange de ces messages est correctement sécurisé et, en particulier, que les entités recevant les *public PKG parameters* font confiance au PKG.

5.3.3 Solution globale

Finalement, nous avons intégré les deux parties décrites précédemment (i.e., utilisation des adresses CGA et utilisation de l'IBC) pour achever notre objectif et obtenir notre solution globale. Ainsi, comme illustré dans la Figure 5.11, (1) un nœud IPv6 génère son adresse CGA, basée sur une paire de clés publique/privée K_{CGApub} et $K_{CGApriv}$. (2) Le nœud IPv6 obtient les *public PKG parameters*. (3) Le nœud IPv6 génère alors son identifiant (i.e., sa clé publique $K_{FQDNpub}$), qui est en fait son FQDN, et obtient du PKG la clé privée associée $K_{FQDNpriv}$. (4) Le nœud IPv6 construit ensuite le message DNS UPDATE en y ajoutant le *CGAparams* RR associé à l'adresse CGA, il génère deux signatures distinctes de ce message, respectivement avec $K_{FQDNpriv}$ et $K_{CGApriv}$, et les inclut dans ce même message (i.e., un SIG RR par signature) avant de l'envoyer. (5) A la réception de ce message, le serveur DNS obtient les *public PKG parameters* du PKG et vérifie la validité de la signature basée sur le FQDN. (6) Finalement, le serveur DNS vérifie la possession de l'adresse CGA et, si les deux vérifications sont correctes, met à jour les informations concernant le nœud IPv6.

Grâce à la vérification de la signature IBC à l'étape (5), le serveur DNS peut controler que l'émetteur du message est le possesseur du FQDN et qu'il est autorisé à mettre à jour des informations liées à celui-ci. Grâce à la vérification CGA à l'étape (6), le serveur DNS peut contrôler que l'émetteur du message est le propriétaire de l'adresse CGA à mettre à jour avec le FQDN déjà vérifié.

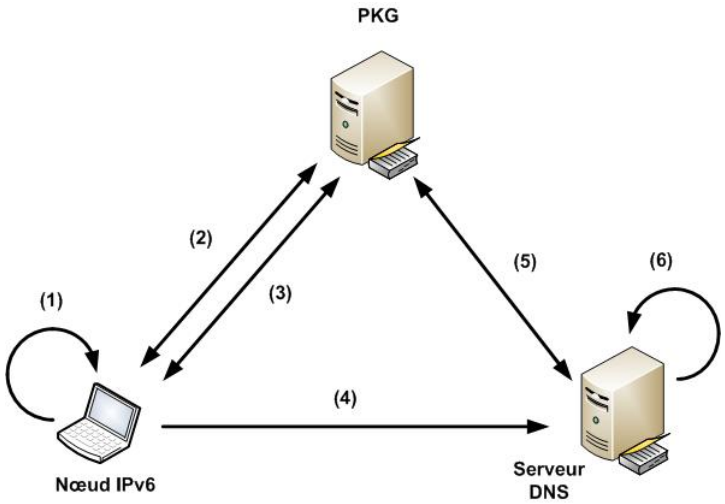

Figure 5.11 – Architecture et échanges de la solution globale

5.4 Implémentation de la méthode de sécurisation à base d'identifiants cryptographiques

Nous avons implémenté et déployé notre solution sur une plate-forme de tests afin de la valider et prouver sa faisabilité.

5.4.1 Plate-forme de tests

La plate-forme, comme illustré dans la Figure 5.12, est composée de 3 entités : un nœud IPv6 jouant le rôle de client DNS, un routeur jouant le rôle de PKG et enfin un serveur DNS. Le système d'exploitation sur ces entités est *Debian 2.6.32*. Cette plate-forme fonctionne entièrement sous IPv6. Le serveur DNS utilise *BIND 9.7.3*[1] produit par ISC. L'implémentation du PKG est basée sur la librairie *Pairing-Based Cryptography* (**PBC**)[2] de l'université de Stanford.

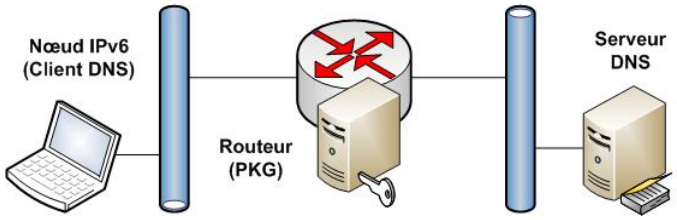

Figure 5.12 – Plateforme expérimentale intégrant notre solution

5.4.2 Architecture logicielle

Nous avons décidé d'implémenter notre solution en 3 composantes : une composante "Client", une composante "Serveur" et une composante "PKG". Comme illustré dans la Figure 5.13, la composante "Serveur", localisée dans le serveur DNS, (4) n'a qu'à envoyer localement une commande `nsupdate` au *daemon* BIND quand (3) la sécurité du message DNS UPDATE est valide. Ceci permet d'éviter des modifications de BIND. (1) La composante "Client" sert à générer l'adresse CGA, sécuriser le message DNS UPDATE en appliquant notre solution et (2) à l'envoyer. La composante "PKG" fournit le matériel de sécurité nécessaire aux composantes "Client" et "Serveur".

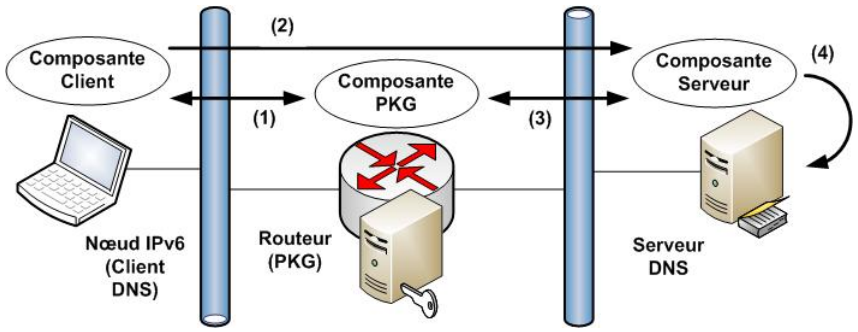

Figure 5.13 – Architecture de l'implémentation

1. http ://www.isc.org/software/bind
2. http ://crypto.stanford.edu/pbc/

Concernant la partie utilisant les adresses CGA de notre solution, nous avons re-utilisé le code de NDProtector [3], une implémentation en Python de SEND, et en particulier le module de génération et validation des adresses CGA. Ce dernier est lui-même basé sur scapy6 [4] qui est un dérivé pour IPv6 du bien connu logiciel scapy [5].

Concernant la partie utilisant l'IBC, nous avons re-utilisé la librairie PBC avec comme modèle de signature, le schéma de Hess [Hes02].

Toutes les fonctions nécessaires à notre solution, illustrées dans la Figure 5.14, sont décrites par composante ci-dessous.

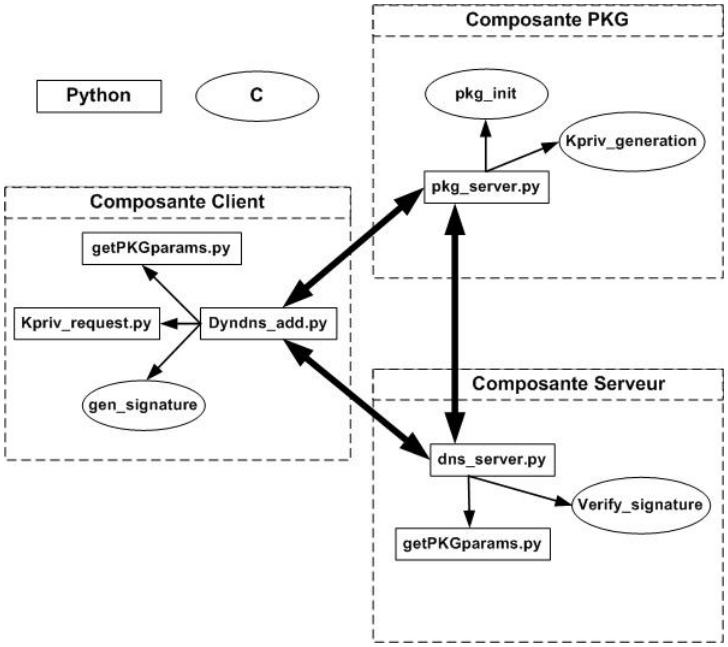

Figure 5.14 – Fonctions de l'implémentation

Composante "PKG"

La composante "PKG" doit fournir le matériel de sécurité nécessaire aux composantes "Client" et "Serveur" : la clé privée associée à un FQDN et les *public PKG parameters* du PKG. Cette composante repose sur 3 fonctions.

La fonction C `pkg_init` initialise le PKG : génération des *public PKG parameters* et du secret "Maître" du PKG. La fonction C `Kpriv_generation` permet la génération de la clé privée (i.e., $K_{FQDNpriv}$) associée à un identifiant (i.e., le FQDN dans notre contexte). Enfin, la fonction Python `pkg_server.py` est la partie partie principale de la composante "PKG" : tout d'abord, elle lance la fonction `pkg_init`, puis, attend des requêtes (i.e., demande des *public PKG parameters* ou d'une clé privée associée à un identifiant, dans ce dernier cas, ce qui lance la fonction `Kpriv_generation`) et y répond.

3. http ://amnesiak.org/NDprotector/
4. http ://natisbad.org/scapy/
5. https ://secdev.org/projects/scapy/

Composante "Client"

La composante "Client" doit générer l'adresse CGA, obtenir du PKG les *public PKG parameters* et la clé privée associée au FQDN, construire et signer le message DNS UPDATE, et finalement l'envoyer. Cette composante repose sur 4 fonctions.

La fonction Python `getPKGparams.py` permet à la composante "Client" de demander et obtenir les *public PKG parameters*. La fonction Python `Kpriv_request.py` est utilisée par la composante "Client" pour réclamer et obtenir la clé privée (i.e., $K_{FQDNpriv}$) associée à un identifiant (i.e., le FQDN dans notre contexte). La fonction C `gen_signature` permet la signature du message DNS UPDATE en utilisant la clé privée (i.e., $K_{FQDNpriv}$) associée au FQDN. Enfin, la fonction Python `Dyndns_add.py` est la partie principale de la composante "Client" : elle génère l'adresse CGA, demande et obtient les *public PKG parameters* en lançant la fonction `getPKGparams.py`, demande et obtient la clé privée associée au FQDN en lançant la fonction `Kpriv_request.py`, construit le message DNS UPDATE en incluant le *CGAparams* RR associé à l'adresse CGA, génère deux signatures distinctes de ce message, respectivement avec $K_{FQDNpriv}$ et $K_{CGApriv}$, en utilisant la fonction `gen_signature`, puis les inclus dans ce même message (i.e., un SIG RR par signature) avant de l'envoyer au serveur DNS.

Composante "Serveur"

Quand il reçoit un message DNS UPDATE, la composante "Server" doit obtenir les *public PKG parameters* du PKG si elle ne les a pas déjà, contrôler la validité de la sécurité des messages DNS UPDATE, et finalement, si la vérification est correcte, notifier la mise à jour à effectuer auprès du *daemon* du serveur DNS. Cette composante repose sur 3 fonctions.

La fonction Python `getPKGparams.py` permet à la composante "Serveur" de réclamer et obtenir les *public PKG parameters*. La fonction C `Verify_signature` permet la vérification de la signature basée sur la clé privée (i.e., $K_{FQDNpriv}$) associée au FQDN. La fonction Python `dns_server.py` est la partie principale de la composante "Serveur" : à la réception d'un message DNS, cette fonction réclame et obtient les *public PKG parameters* grâce à la fonction `getPKGparams.py`, vérifie la possession du FQDN avec la fonction `Verify_signature` et la possession de l'adresse CGA (i.e., regénération de l'adresse CGA et contrôle de la signature avec K_{CGApub}). Finalement, si les vérifications sont correctes, la fonction génère et envoie localement un message DNS UPDATE (i.e., lien entre le FQDN et l'adresse CGA) au *daemon* du serveur DNS grâce à la commande `nsupdate`.

5.5 Analyse de notre solution

Notre solution possède des avantages mais aussi quelques limitations. Pour chacune de ces dernière, nous proposons de potentielles solutions pour les pallier.

5.5.1 Avantages

Concernant les avantages de notre solution, tout d'abord, lors d'une mise à jour, le serveur DNS est toujours capable de vérifier que la possession du FQDN, comme le fait TSIG ou SIG(0). Ainsi, notre nouvelle méthode d'authentification ne dégrade en rien le niveau de sécurité comparativement à ces deux dernières solutions. Mieux, elle permet de vérifier la possession de l'adresse IPv6 contenue dans le message DNS UPDATE qui doit

être associée au FQDN dans le serveur DNS. A notre connaissance, aucune autre solution apportant une telle fonctionnalité de sécurité n'existe à ce jour.

Ensuite, contrairement à SIG(0), notre solution ne nécessite plus le stockage d'une clé publique pour chaque nœud IPv6, grâce au KEY RR [3rd99a], dans le serveur DNS. Ceci apporte deux avantages. Tout d'abord, il n'est plus nécessaire de reconfigurer le serveur DNS quand le nœud IPv6 change de matériel de sécurité pour authentifier ses mises à jour. En effet, il fournit ce matériel en même temps que les messages DNS UPDATE avec notre solution. De plus, dans le cas où DNSSEC est déployé, le fichier de configuration d'un nom de domaine à signer, incluant tous les FQDNs s'y rattachant avec les données associées (e.g., adresse IPv6, clé publique), doit être signé. Or, le temps de signature de ce fichier prend moins de temps avec notre solution car celui-ci est plus petit. Ceci est appréciable quand ce fichier comporte un grand nombre de FQDNs ou que des informations s'y trouvant sont mises à jour très fréquemment.

Avec notre solution, il est simple de connaître le matériel de sécurité relatif à la possession d'un FQDN. En effet, la clé publique est tout simplement ce FQDN, ou alors ce FQDN concaténé à une autre donnée (e.g., une date) qui peut être connue à l'avance par le serveur DNS. Ainsi, les ressources du serveur DNS se focalisent à fournir le service DNS seulement car le serveur DNS n'a pas à gérer d'échanges additionnels (e.g., pour obtenir et configurer la clé publique associée au FQDN pour chaque nœud IPv6) exception faite de l'obtention des *public PKG parameters*, pouvant être effectuée une unique fois si le serveur DNS est capable de les stocker.

De manière identique, le serveur DNS obtient facilement le matériel de sécurité relatif à la possession d'une adresse IPv6. En effet, celui-ci est fourni en même temps que l'envoi du message DNS UPDATE.

Notre solution est plus facilement déployable à grande échelle que TSIG car cette dernière méthode nécessite la provision d'une clé secrète partagée pour chaque nœud désirant mettre à jour ses informations sur le serveur DNS. En effet, avec notre solution, le serveur ne nécessite pas une pré-configuration d'une telle envergure dans le cas d'un grand nombre de nœuds IPv6. Il ne doit connaître que le PKG et avoir une communication sécurisée avec lui.

Enfin, notre solution est basée sur l'utilisation de deux paires de clés publique/privée, ce qui accroît la robustesse de la sécurité des messages DNS UPDATE. En effet, un attaquant potentiel doit compromettre chacune des deux paires afin de falsifier à la fois le FQDN et l'adresse IPv6 inclus dans un message DNS UPDATE.

5.5.2 Limitations et solutions potentielles

Concernant les limites à notre solution, le dernier avantage mentionné précédemment pourrait aussi être pris comme une limitation. En effet, les spécifications CGA [Aur05] n'autorisent que l'utilisation de l'algorithme RSA [RSA78] et l'IBC est quant à lui principalement utilisé avec ECC : du coup, ces deux types d'algorithme nécessitent d'être implémentés/intégrés dans le serveur DNS et ainsi que dans chaque nœud IPv6 désirant employer notre solution. Cette limitation pourrait être résolue en utilisant le même algorithme à la fois pour les adresses CGA et pour l'IBC : soit RSA, soit ECC. Bien que des études aient été faites concernant l'utilisation de RSA avec IBC [DT], il est connu que RSA est coûteux d'un point de vue ressources/temps de calculs et qu'il n'est pas adapté pour des terminaux de faibles capacités (i.e., processeur et batterie électrique) comme les terminaux mobiles ou les capteurs. Malheureusement, les spécifications actuelles CGA interdisent l'utilisation d'algorithmes alternatifs comme ECC, même si des travaux sur ce sujet ont été publiés [CBL10] et des propositions de standard [SXZ11] soumis à l'IETF.

Une autre limitation est liée à la génération des adresses CGA et plus spécifiquement l'obligation d'utiliser SHA-1. Cette limite a été déjà présentée dans le chapitre précédent (cf. Section. 4.3.1). Pour rappel, tout d'abord, selon les dernières cryptanalyses, il serait probable que cet algorithme soit compromis dans un futur proche, ce qui entraînerait du coup une compromission des adresses CGA. La communauté scientifique travaille actuellement sur le remplaçant de cet algorithme : SHA-3. Par ailleurs, il est important de mentionner que l'usage d'algorithmes cryptographiques peut être réglementé dans certains pays ou sociétés : par exemple, en Russie, l'algorithme GOST R 34.11-94 [Dol10] doit être obligatoirement utilisé en lieu et place de toute autre fonction de hachage.

La limitation suivante, qui a aussi été présentée pour les adresses CGA dans le chapitre précédent (cf Section 4.3.1), concerne la révocation. Pour rappel, une adresse CGA possède des propriétés cryptographiques prouvant sa possession mais rien n'empêche un attaquant de l'utiliser si celui-ci parvient à compromettre ces propriétés en trouvant une collision. Or il est impossible de révoquer une adresse CGA comme on pourrait le faire avec un certificat de type X.509. Concernant l'IBC, révoquer une clé publique serait similaire à révoquer un identifiant et donc le FQDN dans notre contexte. Une solution potentielle pour pallier ce problème serait de concaténer une donnée spécifique (e.g., une date) à l'identifiant, comme cela a été déjà proposé dans des travaux antérieurs [BF01].

Enfin, la dernière limitation concerne le PKG, qui devient avec notre solution un point critique : si ce dernier est compromis, alors, toutes les paires de clés publique/privée utilisées par les nœuds IPv6 sont à leur tour compromises. Afin de limiter ce problème, de précédentes études [ArPH03] recommandent que le PKG ne génère seulement qu'une partie de la clé privée associée à un identifiant et que cela soit l'utilisateur final qui achève la génération de cette clé privée.

5.6 Synthèse du chapitre

Dans ce chapitre, nous avons tout d'abord rappelé l'architecture DNS ainsi que son mécanisme de mise à jour dynamique (cf. Section 5.1). Ensuite, nous avons présenté les principales méthodes de sécurisation de ce mécanisme ainsi que leurs limites, en particulier dans un environnement IPv6 où l'autoconfiguration est utilisée (cf. Section 5.2). Dans ce cas, comme nous l'avons montré, la sécurisation ne peux plus reposer sur l'utilisation d'un serveur DHCP, comme en IPv4, qui sert d'intermédiaire pour mettre à jour les données d'un nœud IP dans le serveur DNS.

Nous avons alors présenté notre nouvelle solution qui s'appuie sur des identifiants cryptographiques (cf. Section 5.3). Comme dans le Chapitre 4, nous réutilisons les propriétés cryptographiques des adresses CGA dans notre solution afin de vérifier la possession d'une adresse IPv6 à mettre à jour dans le serveur DNS. Mais, de plus, grâce aux identifiants cryptographiques basés sur l'IBC, dont nous avons rappelé les principes, nous pouvons vérifier la possession du FQDN associé à cette adresse.

Afin de valider nos choix et vérifier le bon fonctionnement de notre solution, nous l'avons implémenté et testé sur une plate-forme expérimentale (cf. Section 5.4). Grâce à nos spécifications qui ne nécessitent que de légères modifications dans le format standardisé des messages DNS, nous avons pu intégrer facilement notre solution dans une implémentation DNS existante et la tester, prouvant ainsi que tout déploiement chez un fournisseur d'accès Internet serait de la même façon faisable.

Finalement, nous avons analysé les avantages et inconvénients de notre solution par rapport aux principales méthodes de sécurisation existantes (cf. Section 5.5). Tout d'abord, comme il n'est plus nécessaire de stocker le matériel de sécurité nécessaire à l'authenti-

fication des données lors des mises à jour, la taille des fichiers de configuration de nom de domaine est réduite, ce qui facilite leurs signatures quand DNSSEC est déployé. Autre avantage, notre solution permet de vérifier la possession de l'adresse IPv6 mise à jour, en plus de celle du FQDN. Or, nous ne connaissons pas d'autres méthodes de sécurisation à ce jour ayant une telle caractéristique. Maintenant, comme notre solution repose sur des identifiants cryptographiques, il est nécessaire que tout nœud IPv6 employant celle-ci supporte les mécanismes cryptographiques inhérents (i.e., adresse CGA et IBC).

Conclusions et perspectives de recherche

A mi-chemin entre le domaine du réseau et le domaine de la cryptographie, les identifiants cryptographiques permettent de sécuriser de manière à la fois simple et forte le mécanisme NDP, cœur du protocole IPv6, via le mécanisme SEND. Mais ces identifiants pourraient aussi s'employer pour la sécurisation d'autres services. Dans cette thèse, nous avons essayé d'analyser le mécanisme SEND employant ce type d'identifiant, les adresses CGA, puis de travailler sur l'utilisation de tels identifiants pour la sécurisation en IPv6.

Le premier problème que nous avons souhaité étudier concerne la fiabilité du mécanisme SEND et des adresses CGA qu'il utilise. En effet, les spécifications de ces technologies à l'IETF sont récentes et la communauté scientifique a peu analysé celles-ci. Du coup, pouvons-nous considérer la sécurité fournie par ces technologie comme fiable ? N'ont-elles pas des limites d'utilisation ?

Le deuxième problème que nous avons souhaité traiter concerne la réutilisation des adresses CGA pour la sécurisation de services en IPv6. Tout d'abord, une fois le mécanisme SEND fiable, ne pourrait-il pas être la base de solutions de sécurisation ? Dans le cadre de ce mécanisme, les propriétés cryptographiques des adresses CGA ne sont utilisées que localement (i.e., sur le même lien réseau). Ne serait-il pas possible de les employer sur une échelle plus grande ? N'est-il pas possible d'utiliser d'autres types d'identifiants cryptographiques avec ces adresses ?

Notre première contribution est la découverte d'une attaque par rejeu de messages dans le mécanisme SEND [CC08] (cf. Section 2.4.2). Cette attaque empêche un nœud IPv6 de pouvoir s'assigner une adresse IPv6, l'empêchant du coup de pouvoir communiquer avec un autre nœud IPv6. Nous avons proposé les modifications à effectuer dans le mécanisme SEND pour contrer cette attaque. Nous espérons qu'elles seront prises en compte lors de la révision des standards SEND et CGA.

Notre deuxième contribution est une modification du mécanisme SEND pour sécuriser les messages NDP entre un routeur et un nœud IPv6 (cf. Section 2.4.1). Elle a été proposée dans le groupe de travail CSI [XKH+08]. Elle consiste à remplacer l'emploi de la cryptographie asymétrique par de la cryptographie symétrique, moins coûteuse en temps et puissance de calculs pour le routeur. Comme le mécanisme SEND n'est pas encore déployé, l'adoption de cette proposition a peu de chance d'aboutir dans un futur proche.

Notre troisième contribution est l'identification de scénarios dans lesquels le mécanisme SEND, et en particulier les adresses CGA, ne peut être utilisé tel qu'il est spécifié et qui a l'objet d'une RFC [CKD10] (cf. Section 2.4.2). Ces scénarios ne sont pas rares car cela se produit quand au moins deux nœuds utilisent une même adresse IPv6. L'usage de protocoles de mobilité IPv6, d'IPsec, du mécanisme ND Proxy et d'adresses de type *anycast* s'en retrouvent impacté. Aussi, suite à notre découverte, l'IETF a créé le groupe de travail CSI afin d'y spécifier une solution remédiant à cette problématique.

Notre quatrième contribution est l'identification de la limite de la sécurisation des messages NDP apportée par le mécanisme SEND dans un contexte de mobilité IPv6 [DCLM08] (cf. Section 2.4.2). Le mécanisme SEND permet d'apporter la preuve qu'un routeur est bien habilité à l'être et qu'il est autorisé à avertir des préfixes IPv6 spécifiques. La sécurité apportée par ce mécanisme ne couvre que ces informations. Aussi, toute autre information dans un message RA, comme les caractéristiques d'un HA pour les protocoles de mobilité IPv6, ne peut être considérée comme fiable même si le mécanisme SEND est employé. Comme le mécanisme DHAAD a peu de chance d'être déployé, en partie grâce à notre contribution et à notre participation à la spécification de mécanismes alternatifs, cette proposition a peu de chance d'être adoptée à l'IETF.

Notre cinquième contribution est la participation, dans le groupe de travail SAVI à l'IETF, à la standardisation de solutions permettant de contrer l'usurpation d'une adresse IP source [CL12] (cf. Section 3.4). Ces solutions reposent sur l'observation de la signalisation des mécanismes d'assignation d'adresses IP, comme NDP ou SEND. En tant que président de ce groupe de travail, nous avons participé à la spécification de ces solutions, en apportant les modifications nécessaires pour qu'elles soient conformes aux standards déjà existants, et nous les avons portés, en les défendant au niveau technique auprès de la communauté IETF, afin qu'elles deviennent les standards de demain.

Notre sixième contribution est une proposition de standard à l'IETF concernant l'intégration des solutions FCFS SAVI et SEND SAVI dans une architecture IPv6 de type ADSL/Fibre [CPLC12] (cf. Section 3.5). Etant standardisée au BBF, cette architecture est le standard international lors d'un déploiement IPv6 pour les technologies ADSL et Fibre. Notre proposition décrit où doit se situer l'entité SAVI et quel type de Binding Anchor utiliser dans cette architecture. Cette proposition devrait prochainement devenir une RFC (cela devrait être finalisé d'ici Octobre 2012). Ainsi, notre contribution sera mise en œuvre chez tout fournisseur d'accès Internet employant une telle architecture.

Notre septième contribution est une nouvelle méthode d'authentification pour le mécanisme IKEv2 [CWL11] [CWL12] (cf. Section 4.5). Elle réutilise les propriétés cryptographiques des adresses CGA tout en gardant le même niveau de sécurité et en palliant certaines limitations, en particulier la nécessité de mettre en place une infrastructure de confiance, des méthodes déjà existantes. Cette nouvelle méthode a été implémentée et testée (cf. Section 4.6). Ceci nous a permis de confirmer que les identifiants cryptographiques, les adresses CGA dans ce cas présent, pouvaient constituer un matériel de sécurité alternatif, facilement utilisable, lors de la sécurisation d'un service IPv6. Suite aux bons retours lors de la présentation des résultats de cette contribution à l'IETF, nous pensons retravailler les spécifications de la proposition initiale [LMK07], avec l'accord des auteurs, pour qu'elle soit adoptée comme RFC.

Notre huitième contribution est une proposition de standard à l'IETF permettant l'utilisation d'IPsec/IKEv2 pour sécuriser de manière forte la signalisation de la RO pour MIPv6 [DC08] (cf. Section 4.4). Elle est un exemple concret d'usage de notre méthode alternative d'authentification pour IKEv2 basée sur les adresses CGA. Bien que la standardisation de cette proposition soit gelée actuellement, elle pourrait reprendre suite à l'intérêt récent de la communauté scientifique pour des mécanismes sécurisés d'*Offloading* pour les communications mobiles.

Notre neuvième et dernière contribution est une nouvelle solution de sécurisation pour la mise à jour DNS dynamique [CAL12] (cf. Section 5.3). Elle repose, encore une fois, sur l'utilisation des adresses CGA mais aussi sur l'IBC. Cette solution permet de pallier aux limites des solutions déjà existantes dans un contexte d'autoconfiguration IPv6 et apporte la preuve de possession d'un FQDN. A ce jour, nous ne connaissons pas d'autres solutions

ayant cette dernière caractéristique. Nous avons implémenté et testé notre solution sur une plate-forme expérimentale (cf. Section 5.4).

Perspectives de recherche

En premier lieu, nous avons vu que le mécanisme SEND, et les adresses CGA, comporte des limitations. Nous avons donc proposé des solutions pour y pallier. Il est nécessaire maintenant de les pousser à l'IETF afin de rendre le déploiement du mécanisme SEND plus facile et plus sûr. Maintenant, plusieurs points demandent encore des travaux de recherche. Tout d'abord, concernant la problématique d'utilisation de préfixes de type ULA avec SEND (cf. Section 2.4.2), nous devons finaliser les spécifications de notre solution et l'implémenter afin de vérifier sa validité. Ensuite, comme le mécanisme SEND emploie des certificats X.509, pouvant être basés sur une RPKI, il serait intéressant de spécifier une méthode sécurisée permettant de les fournir dynamiquement. Cela faciliterait grandement le déploiement du mécanisme SEND. Enfin, la solution standardisée aujourd'hui à l'IETF palliant le problème d'utilisation d'une même adresse IPv6 par plusieurs nœuds (cf. Section 2.4.2) repose sur l'usage de certificats. Nous trouvons qu'elle est lourde à mettre en place et nous pensons qu'une solution plutôt basée sur la signature de groupe aurait été plus adaptée. Nous avons commencé à réfléchir sur une telle solution pendant notre thèse mais nous n'avons pas pu la finaliser.

Dans notre thèse, nous avons proposé plusieurs mécanismes réutilisant les propriétés des adresses CGA. Par manque de temps, nous n'avons pas pu évaluer les performances de celles-ci afin de les comparer aux mécanismes déjà existants. Il serait intéressant d'effectuer de telles évaluations. De même, ces mécanismes pourraient être poussés en standardisation à l'IETF.

Finalement, au cours de cette thèse, nous nous sommes concentrés sur la réutilisation des adresses CGA pour la sécurisation de services IPv6. Nous pensons qu'il est possible aussi de réutiliser les certificats X.509. Nous avons en tête la sécurisation de la RO pour le protocole NEMO. En effet, celle-ci nécessite d'apporter la preuve de la possession de préfixes IPv6. Or les certificats X.509, employés par le mécanisme SEND, nous la fournissent.

Liste des contributions

Articles

[BCLML07] J. Bournelle, J-M. Combes, M. Laurent-Maknavicius, and S. Larafa. Using pana for mobile ipv6 bootstrapping. In *NETWORKING 2007. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, volume 4479 of *Lecture Notes in Computer Science*, pages 345–355. 2007.

[BVB+06] J. Bournelle, G. Valadon, D. Binet, S. Zrelli, M. Laurent-Maknavicius, and J-M. Combes. AAA considerations within several NEMO deployments scenarios. In *The first International Workshop on Network Mobility (NEMO)*, 2006.

[CAL12] J-M. Combes, G. Arfaoui, and M. Laurent. Dynamic DNS Update security, based on Cryptographically Generated Addresses and ID-Based Cryptography, in an IPv6 autoconfiguration context. In *ARES '12 : 7th International Conference on Availability, Reliability and Security*, 2012.

[CC08] T. Cheneau and J-M. Combes. Une attaque par rejeu sur le protocole SEND. In *SAR-SSI '08 : 3e Conférence sur la Sécurité des Architectures Réseaux et des Systèmes d'Information*, pages 289 – 300, 2008.

[CL12] J-M. Combes and M. Laurent. Source Address Validation Improvements (SAVI), mécanismes de prévention contre l'usurpation d'adresses IP source. In *SSTIC '12 : Symposium sur la sécurité des technologies de l'information et des communications*, pages 264 – 295, 2012.

[CWL11] J-M. Combes, A. Wailly, and M. Laurent. Internet Key Exchange Version 2 with IPV6 Cryptographically Generated Addresses. In *ICSNA '12 : International Conference on Secure networking and Applications*, 2011.

[CWL12] J-M. Combes, A. Wailly, and M. Laurent. CGA as alternative security credentials with IKEv2 : implementation and analysis. In *SAR-SSI '12 : 7e Conférence sur la Sécurité des Architectures Réseaux et des Systèmes d'Information*, pages 53 – 59, 2012.

[LBC+06] R. Marin Lopez, J. Bournelle, J-M. Combes, M. Laurent-Maknavicius, and A. F. Gomez Skarmeta. Improved EAP keying framework for a secure mobility access service. In *International Wireless Communications and Mobile Computing Conference IWCMC 2006*, 2006.

IETF

[CKD10] J-M. Combes, S. Krishnan, and G. Daley. Securing Neighbor Discovery Proxy : Problem Statement. RFC 5909, Internet Engineering Task Force, July 2010.

[CPLC12] F. Costa, X. Pougnard, L. LiHongyu, and J-M. Combes. Duplicate Address Detection Proxy. Internet-Draft draft-ietf-6man-dad-proxy-02, Internet Engineering Task Force, March 2012. Work in progress.

[DC08] F. Dupont and J-M. Combes. Using IPsec between Mobile and Correspondent IPv6 Nodes. Internet-Draft draft-ietf-mip6-cn-ipsec-08, Internet Engineering Task Force, August 2008. Expired.

[DCLM08] F. Dupont, J-M. Combes, and M. Laurent-Maknavicius. Dynamic Home Agent Address Discovery (DHAAD) Considered Harmful. Internet-Draft draft-dupont-mext-dhaadharmful-00, Internet Engineering Task Force, June 2008. Expired.

[XKH+08] F. Xia, S. Krishnan, W. Haddad, J-M. Combes, and C. Li. Distributing a Symmetric Neighbor Discovery Key Using SEND. Internet-Draft draft-xia-csi-symmetric-key-00, Internet Engineering Task Force, June 2008. Expired.

Brevets

[CM12] J-M. Combes and D. Migault. Method for the secure allocation, to a private network node, of an IPv6 address, January 2012. WO/2012/001273.

[eDM11] J-M. Combes et D. Migault. Procédé d'allocation sécurisée d'une adresse IPv6 à un noeud d'un réseau privé, Décembre 2011. FR 2961994 (A1).

Bibliographie

[3rd99a] D. Eastlake 3rd. Domain Name System Security Extensions. RFC 2535, Internet Engineering Task Force, March 1999.

[3rd99b] D. Eastlake 3rd. DSA KEYs and SIGs in the Domain Name System (DNS). RFC 2536, Internet Engineering Task Force, March 1999.

[3rd99c] D. Eastlake 3rd. Storage of Diffie-Hellman Keys in the Domain Name System (DNS). RFC 2539, Internet Engineering Task Force, March 1999.

[3rd00] D. Eastlake 3rd. DNS Request and Transaction Signatures (SIG(0)s). RFC 2931, Internet Engineering Task Force, September 2000.

[3rd01] D. Eastlake 3rd. RSA/SHA-1 SIGs and RSA KEYs in the Domain Name System (DNS). RFC 3110, Internet Engineering Task Force, May 2001.

[AA04] D. Atkins and R. Austein. Threat Analysis of the Domain Name System (DNS). RFC 3833, Internet Engineering Task Force, August 2004.

[AAL+05] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. RFC 4033, Internet Engineering Task Force, March 2005.

[ABV+04] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz. Extensible Authentication Protocol (EAP). RFC 3748, Internet Engineering Task Force, June 2004.

[AJWB10] C. An, J. JiahaiYang, J. Wu, and J. Bi. Definition of Managed Objects for SAVI Protocol. Internet-Draft draft-an-savi-mib-00, Internet Engineering Task Force, December 2010. Work in progress.

[AJWB11] C. An, J. JiahaiYang, J. Wu, and J. Bi. Definition of Managed Objects for SAVI Protocol. Internet-Draft draft-an-savi-mib-02, Internet Engineering Task Force, December 2011. Work in progress.

[AKZN05] J. Arkko, J. Kempf, B. Zill, and P. Nikander. SEcure Neighbor Discovery (SEND). RFC 3971, Internet Engineering Task Force, March 2005.

[ArPH03] S. Al-riyami, K. Paterson, and R. Holloway. Certificateless public key cryptography. pages 452–473. Springer-Verlag, 2003.

[Aur] T. Aura. Cryptographically generated addresses (cga). In *Information Security*, volume 2851 of *Lecture Notes in Computer Science*, pages 29–43. Springer Berlin / Heidelberg.

[Aur05] T. Aura. Cryptographically Generated Addresses (CGA). RFC 3972, Internet Engineering Task Force, March 2005.

[BF01] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. pages 213–229. Springer-Verlag, 2001.

[BGM12] M. Bagnulo and A. Garcia-Martinez. SEND-based Source-Address Validation Implementation. Internet-Draft draft-ietf-savi-send-07, Internet Engineering Task Force, March 2012. Work in progress.

[BS04] F. Baker and P. Savola. Ingress Filtering for Multihomed Networks. RFC 3704, Internet Engineering Task Force, March 2004.

[BWYB10] J. Bi, J. Wu, G. Yao, and F. Baker. SAVI Solution for DHCP. Internet-Draft draft-ietf-savi-dhcp-07, Internet Engineering Task Force, November 2010. Work in progress.

[BWYB12] J. Bi, J. Wu, G. Yao, and F. Baker. SAVI Solution for DHCP. Internet-Draft draft-ietf-savi-dhcp-12, Internet Engineering Task Force, February 2012. Work in progress.

[BYHLA11a] J. Bi, G. Yao, J. Halpern, and E. Levy-Abegnoli. SAVI for Mixed Address Assignment Methods Scenario. Internet-Draft draft-ietf-savi-mix-01, Internet Engineering Task Force, October 2011. Work in progress.

[BYHLA11b] J. Bi, G. Yao, J. Halpern, and E. Levy-Abegnoli. SAVI for Mixed Address Assignment Methods Scenario. Internet-Draft draft-bi-savi-mix-04, Internet Engineering Task Force, September 2011. Work in progress.

[BYWB10] J. Bi, G. Yao, J. Wu, and F. Baker. SAVI Solution for Stateless Address. Internet-Draft draft-bi-savi-stateless-00, Internet Engineering Task Force, April 2010. Expired.

[CAL12] J-M. Combes, G. Arfaoui, and M. Laurent. Dynamic DNS Update security, based on Cryptographically Generated Addresses and ID-Based Cryptography, in an IPv6 autoconfiguration context. In *ARES '12 : 7th International Conference on Availability, Reliability and Security*, 2012.

[CBL10] T. Cheneau, A. Boudguiga, and M. Laurent. Significantly improved performances of the cryptographically generated addresses thanks to ecc and gpgpu. volume 29, pages 419 – 431, 2010.

[CC08] T. Cheneau and J-M. Combes. Une attaque par rejeu sur le protocole SEND. In *SAR-SSI '08 : 3e Conférence sur la Sécurité des Architectures Réseaux et des Systèmes d'Information*, pages 289 – 300, 2008.

[CDG06] A. Conta, S. Deering, and M. Gupta. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC 4443, Internet Engineering Task Force, March 2006.

[Cho11] T. Chown. IPv6 Address Accountability Considerations. Internet-Draft draft-chown-v6ops-address-accountability-01, Internet Engineering Task Force, July 2011. Work in progress.

[CKD10] J-M. Combes, S. Krishnan, and G. Daley. Securing Neighbor Discovery Proxy : Problem Statement. RFC 5909, Internet Engineering Task Force, July 2010.

[CL12] J-M. Combes and M. Laurent. Source Address Validation Improvements (SAVI), mécanismes de prévention contre l'usurpation d'adresses IP source. In *SSTIC '12 : Symposium sur la sécurité des technologies de l'information et des communications*, pages 264 – 295, 2012.

[CM12] J-M. Combes and D. Migault. Method for the secure allocation, to a private network node, of an IPv6 address, January 2012. WO/2012/001273.

[CMLN04] C. Castelluccia, G. Montenegro, J. Laganier, and C. Neumann. Hindering eavesdropping via ipv6 opportunistic encryption. In *in Proceedings of the European Symposium on Research in Computer Security, Lecture Notes in Computer Science*, pages 309–321. Springer-Verlag, 2004.

[CPLC12] F. Costa, X. Pougnard, L. LiHongyu, and J. Combes. Duplicate Address Detection Proxy. Internet-Draft draft-ietf-6man-dad-proxy-02, Internet Engineering Task Force, March 2012. Work in progress.

[CSF+08] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, Internet Engineering Task Force, May 2008.

[CWL11] J-M. Combes, A. Wailly, and M. Laurent. Internet Key Exchange Version 2 with IPV6 Cryptographically Generated Addresses. In *ICSNA '12 : International Conference on Secure networking and Applications*, 2011.

[CWL12] J-M. Combes, A. Wailly, and M. Laurent. CGA as alternative security credentials with IKEv2 : implementation and analysis. In *SAR-SSI '12 : 7e Conférence sur la Sécurité des Architectures Réseaux et des Systèmes d'Information*, pages 53 – 59, 2012.

[DBV+03] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). RFC 3315, Internet Engineering Task Force, July 2003.

[DC08] F. Dupont and J-M. Combes. Using IPsec between Mobile and Correspondent IPv6 Nodes. Internet-Draft draft-ietf-mip6-cn-ipsec-08, Internet Engineering Task Force, August 2008. Expired.

[DCLM08] F. Dupont, J-M. Combes, and M. Laurent-Maknavicius. Dynamic Home Agent Address Discovery (DHAAD) Considered Harmful. Internet-Draft draft-dupont-mext-dhaadharmful-00, Internet Engineering Task Force, June 2008. Expired.

[DCU10] V. Dolmatov, A. Chuprina, and I. Ustinov. Use of GOST Signature Algorithms in DNSKEY and RRSIG Resource Records for DNSSEC. RFC 5933, Internet Engineering Task Force, July 2010.

[DH98] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, Internet Engineering Task Force, December 1998.

[Dol10] V. Dolmatov. GOST R 34.11-94 : Hash Function Algorithm. RFC 5831, Internet Engineering Task Force, March 2010.

[DR08] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, Internet Engineering Task Force, August 2008.

[Dro97] R. Droms. Dynamic Host Configuration Protocol. RFC 2131, Internet Engineering Task Force, March 1997.

[DT] X. Ding and G. Tsudik. Simple identity-based cryptography with mediated rsa. In *RSA, Proceedings CT-RSA 2003, LNCS 2612, Springer-Verlag 2003*, pages 193–210. Springer.

[DWPT05] V. Devarapalli, R. Wakikawa, A. Petrescu, and P. Thubert. Network Mobility (NEMO) Basic Support Protocol. RFC 3963, Internet Engineering Task Force, January 2005.

[ECBM08] A. Ebalard, J-M. Combes, A. Boudguiga, and M. Maknavicius. IPv6 autoconfiguration mechanisms security (node part). Technical report, ANR, 2008.

[ECC+08] A. Ebalard, J-M. Combes, M. Charfi, M. Maknavicius, and F. Fainelli. IPv6 autoconfiguration mechanisms security (router part). Technical report, ANR, 2008.

[Edd07] W. Eddy. TCP SYN Flooding Attacks and Common Mitigations. RFC 4987, Internet Engineering Task Force, August 2007.

[eDM11] J-M. Combes et D. Migault. Procédé d'allocation sécurisée d'une adresse IPv6 à un noeud d'un réseau privé, Décembre 2011. FR 2961994 (A1).

[EF94] K. Egevang and P. Francis. The IP Network Address Translator (NAT). RFC 1631, Internet Engineering Task Force, May 1994.

[ELM10] P. Eronen, J. Laganier, and C. Madson. IPv6 Configuration in Internet Key Exchange Protocol Version 2 (IKEv2). RFC 5739, Internet Engineering Task Force, February 2010.

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, Internet Engineering Task Force, June 1999.

[FS00] P. Ferguson and D. Senie. Network Ingress Filtering : Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2827, Internet Engineering Task Force, May 2000.

[GJK12] I. Gashinsky, J. Jaeggli, and W. Kumari. Operational Neighbor Discovery Problems. Internet-Draft draft-ietf-v6ops-v6nd-problems-05, Internet Engineering Task Force, March 2012. Work in progress.

[GKK12] R. Gagliano, S. Krishnan, and A. Kukec. Certificate Profile and Certificate Management for SEcure Neighbor Discovery (SEND). RFC 6494, Internet Engineering Task Force, February 2012.

[GLD+08] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, and B. Patil. Proxy Mobile IPv6. RFC 5213, Internet Engineering Task Force, August 2008.

[Gon12a] F. Gont. A method for Generating Stable Privacy-Enhanced Addresses with IPv6Stateless Address Autoconfiguration (SLAAC). Internet-Draft draft-gont-6man-stable-privacy-addresses-01, Internet Engineering Task Force, March 2012. Work in progress.

[Gon12b] F. Gont. Neighbor Discovery Shield (ND-Shield) : Protecting against Neighbor Discovery Attacks. Internet-Draft draft-gont-opsec-ipv6-nd-shield-00, Internet Engineering Task Force, June 2012. Work in progress.

[HC98] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). RFC 2409, Internet Engineering Task Force, November 1998.

[HD06] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 4291, Internet Engineering Task Force, February 2006.

[Hes02] F. Hess. Efficient identity based signature schemes based on pairings. In *SAC 2002, LNCS 2595*, pages 310–324. Springer-Verlag, 2002.

[HH05] R. Hinden and B. Haberman. Unique Local IPv6 Unicast Addresses. RFC 4193, Internet Engineering Task Force, October 2005.

[HPW02] D. Harrington, R. Presuhn, and B. Wijnen. An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. RFC 3411, Internet Engineering Task Force, December 2002.

[II00] IAB and IESG. IETF Policy on Wiretapping. RFC 2804, Internet Engineering Task Force, May 2000.

[iost02] National institute of standards and technology. FIPS 180-2, Secure Hash Standard, Federal Information Processing Standard (FIPS), Publication 180-2. Technical report, DEPARTMENT OF COMMERCE, August 2002.

[JLN11] E. Jankiewicz, J. Loughney, and T. Narten. IPv6 Node Requirements. RFC 6434, Internet Engineering Task Force, December 2011.

[KBC97] H. Krawczyk, M. Bellare, and R. Canetti. HMAC : Keyed-Hashing for Message Authentication. RFC 2104, Internet Engineering Task Force, February 1997.

[Ken05a] S. Kent. IP Authentication Header. RFC 4302, Internet Engineering Task Force, December 2005.

[Ken05b] S. Kent. IP Encapsulating Security Payload (ESP). RFC 4303, Internet Engineering Task Force, December 2005.

[KHNE10] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen. Internet Key Exchange Protocol Version 2 (IKEv2). RFC 5996, Internet Engineering Task Force, September 2010.

[KNB+11] M. Kohno, B. Nitzan, R. Bush, Y. Matsuzaki, L. Colitti, and T. Narten. Using 127-Bit IPv6 Prefixes on Inter-Router Links. RFC 6164, Internet Engineering Task Force, April 2011.

[Kob87] N. Koblitz. Elliptic Curve Cryptosystems. volume 48, pages 203–209, 1987.

[KS05] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301, Internet Engineering Task Force, December 2005.

[Kum12] W. Kumari. Neighbor Discovery Enhancements for DOS mititgation. Internet-Draft draft-gashinsky-6man-v6nd-enhance-00, Internet Engineering Task Force, January 2012. Work in progress.

[LAdVPM11] E. Levy-Abegnoli, G. Van de Velde, C. Popoviciu, and J. Mohacsi. IPv6 Router Advertisement Guard. RFC 6105, Internet Engineering Task Force, February 2011.

[LK12] M. Lepinski and S. Kent. An Infrastructure to Support Secure Internet Routing. RFC 6480, Internet Engineering Task Force, February 2012.

[LMK07] J. Laganier, G. Montenegro, and A. Kukec. Using IKE with IPv6 Cryptographically Generated Addresses. Internet-Draft draft-laganier-ike-ipv6-cga-02, Internet Engineering Task Force, July 2007. Obsolete.

[MBH11] D. McPherson, F. Baker, and J. Halpern. SAVI Threat Scope. Internet-Draft draft-ietf-savi-threat-scope-05, Internet Engineering Task Force, April 2011. Work in progress.

[MD99] P. Marques and F. Dupont. Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing. RFC 2545, Internet Engineering Task Force, March 1999.

[Mil] V. Miller. Use of elliptic curves in cryptography. In Hugh Williams, editor, *Advances in Cryptology - CRYPTO '85 Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer Berlin / Heidelberg.

[MNJH08] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Host Identity Protocol. RFC 5201, Internet Engineering Task Force, April 2008.

[Moc87a] P.V. Mockapetris. Domain names - concepts and facilities. RFC 1034, Internet Engineering Task Force, November 1987.

[Moc87b] P.V. Mockapetris. Domain names - implementation and specification. RFC 1035, Internet Engineering Task Force, November 1987.

[Moo06] N. Moore. Optimistic Duplicate Address Detection (DAD) for IPv6. RFC 4429, Internet Engineering Task Force, April 2006.

[MT07] M. Myers and H. Tschofenig. Online Certificate Status Protocol (OCSP) Extensions to IKEv2. RFC 4806, Internet Engineering Task Force, February 2007.

[MTJH12] R. Moskowitz, T. TobiasHeer, P. Jokela, and T. Henderson. Host Identity Protocol Version 2 (HIPv2). Internet-Draft draft-ietf-hip-rfc5201-bis-08, Internet Engineering Task Force, March 2012. Work in progress.

[NBLA12] E. Nordmark, M. Bagnulo, and E. Levy-Abegnoli. FCFS SAVI : First-Come First-Serve Source-Address Validation for Locally Assigned IPv6 Addresses. RFC 6620, Internet Engineering Task Force, May 2012.

[NDK07] T. Narten, R. Draves, and S. Krishnan. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. RFC 4941, Internet Engineering Task Force, September 2007.

[NKN04] P. Nikander, J. Kempf, and E. Nordmark. IPv6 Neighbor Discovery (ND) Trust Models and Threats. RFC 3756, Internet Engineering Task Force, May 2004.

[NLD07] P. Nikander, J. Laganier, and F. Dupont. An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers (ORCHID). RFC 4843, Internet Engineering Task Force, April 2007.

[NLD11] P. Nikander, J. Laganier, and F. Dupont. An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers (ORCHID). Internet-Draft draft-ietf-hip-rfc4843-bis-01, Internet Engineering Task Force, March 2011. Expired.

[NNSS07] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. Neighbor Discovery for IP version 6 (IPv6). RFC 4861, Internet Engineering Task Force, September 2007.

[PA98] R. Pereira and R. Adams. The ESP CBC-Mode Cipher Algorithms. RFC 2451, Internet Engineering Task Force, November 1998.

[PJA11] C. Perkins, D. Johnson, and J. Arkko. Mobility Support in IPv6. RFC 6275, Internet Engineering Task Force, July 2011.

[Plu82] D. Plummer. Ethernet Address Resolution Protocol : Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. RFC 0826, Internet Engineering Task Force, November 1982.

[Pos80] J. Postel. User Datagram Protocol. RFC 0768, Internet Engineering Task Force, August 1980.

[Pos81] J. Postel. Internet Protocol. RFC 0791, Internet Engineering Task Force, September 1981.

[Res00] E. Rescorla. HTTP Over TLS. RFC 2818, Internet Engineering Task Force, May 2000.

[Ric05] M. Richardson. A Method for Storing IPsec Keying Material in DNS. RFC 4025, Internet Engineering Task Force, March 2005.

[Riv92] R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321, Internet Engineering Task Force, April 1992.

[RLH06] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271, Internet Engineering Task Force, January 2006.

[RMK+96] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address Allocation for Private Internets. RFC 1918, Internet Engineering Task Force, February 1996.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. volume 21, pages 120–126, New York, NY, USA, February 1978. ACM.

[SE01] P. Srisuresh and K. Egevang. Traditional IP Network Address Translator (Traditional NAT). RFC 3022, Internet Engineering Task Force, January 2001.

[Sha85] A. Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[SXZ11] B. Sarikaya, F. Xia, and G. Zaverucha. Lightweight Secure Neighbor Discovery for Low-power and LossyNetworks. Internet-Draft draft-sarikaya-6lowpan-cgand-01, Internet Engineering Task Force, May 2011. Work in progress.

[TNJ07] S. Thomson, T. Narten, and T. Jinmei. IPv6 Stateless Address Autoconfiguration. RFC 4862, Internet Engineering Task Force, September 2007.

[TTP06] D. Thaler, M. Talwar, and C. Patel. Neighbor Discovery Proxies (ND Proxy). RFC 4389, Internet Engineering Task Force, April 2006.

[VGrW00] P. Vixie, O. Gudmundsson, D. Eastlake 3rd, and B. Wellington. Secret Key Transaction Authentication for DNS (TSIG). RFC 2845, Internet Engineering Task Force, May 2000.

[VTRB97] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound. Dynamic Updates in the Domain Name System (DNS UPDATE). RFC 2136, Internet Engineering Task Force, April 1997.

[WBB+12] J. Wu, J. Bi, M. Bagnulo, F. Baker, and C. Vogt. Source Address Validation Improvement Framework. Internet-Draft draft-ietf-savi-framework-06, Internet Engineering Task Force, January 2012. Work in progress.

[WR08] N. Williams and M. Richardson. Better-Than-Nothing Security : An Unauthenticated Mode of IPsec. RFC 5386, Internet Engineering Task Force, November 2008.

[XKH+08] F. Xia, S. Krishnan, W. Haddad, J-M. Combes, and C. Li. Distributing a Symmetric Neighbor Discovery Key Using SEND. Internet-Draft draft-xia-csi-symmetric-key-00, Internet Engineering Task Force, June 2008. Expired.

Acronymes

ADD : *Authorization Delegation Discovery*
AH : *Authentication Header*
AN : *Access Node*
ARP : *Address Resolution Protocol*
BA : *Binding Acknowledgment*
BAnchor : *Binding Anchor*
BBF : *Broadband Forum*
BNG : *Broadband Network Gateway*
BT : *Binding Table*
BTNS : *Better-Than-Nothing Security*
BU : *Binding Update*
CA : *Certification Authority*
CGA : *Cryptographically Generated Addresses*
CN : *Correspondent Node*
CoA : *Care-of Address*
CoT : *Care-of Test*
CoTI : *Care-of Test Init*
CPA : *Certification Path Advertisement*
CPE : *Customer Premises Equipment*
CPS : *Certification Path Solicitation*
CSI : *CGA & SEND maIntenance*
DAD : *Duplicate Address Detection*
DAD_NS : *DAD NS*
DHAAD : *Dynamic Home Agent Address Discovery*
DHCP : *Dynamic Host Configuration Protocol*
DHCPv6 : *Dynamic Host Configuration Protocol for IPv6*
DNS : *Domain Name System*
DNS UPDATE : *DNS update*
DNSSEC : *Domain Name Server security*
DoS : *Denial of Service*
EAP : *Extensible Authentication Protocol*
ECC : *Elliptic Curve Cryptography*
ESP : *Encapsulating Security Payload*
FCFS SAVI : *First-Come First-Serve SAVI*
FCFS SAVI DB : *FCFS SAVI Data Base*
FCFS SAVI PL : *FCFS SAVI Prefix List*
FQDN : *Fully Qualified Domain Name*
HA : *Home Agent*
HoA : *Home Address*

HoT : *Home Test*
HoTI : *Home Test Init*
HIP : *Host Identity Protocol*
HTTP : *Hypertext Transfer Protocol*
IBC : *Identity-Based Cryptography*
IBE : *Identity-Based Encryption*
ICMPv6 : *Internet Control Message Protocol for IPv6*
IETF : *Internet Engineering Task Force*
IID : *Interface Identifier*
IKEv2 : *Internet Key Exchange version 2*
IPsec : *Internet Protocol security*
IPv4 : *Internet Protocol version 4*
IPv6 : *Internet Protocol version 6*
LAND : *Local Area Network Denial*
LLA : *Link-Layer Address*
LUA : *Link-local Unicast Address*
MAC : *Media Access Control*
MitM : *Man-in-the-Middle*
MN : *Mobile Node*
MIB : *Management Information Base*
MIPv6 : *Mobility Support in IPv6*
NA : *Neighbor Advertisement*
ND : *Neighbor Discovery*
ND Proxy : *Neighbor Discovery Proxy*
NDP : *Neighbor Discovery Protocol*
NEMO : *Network Mobility Basic Support Protocol*
NS : *Neighbor Solicitation*
NUD : *Neighbor Unreachability Detection*
oDAD : *Optimistic Duplicate Address Detection*
ORCHID : *Overlay Routable Cryptographic Hash Identifiers*
PAD : *Peer Authorization Database*
PBC : *Pairing-Based Cryptography*
PKG : *Private Key Generator*
PKI : *Public Key Infrastructure*
PMIPv6 : *Proxy Mobile IPv6*
PRF : *Pseudo-Random Function*
RA : *Router Advertisement*
RD : *Router Discovery*
RFC : *Request for Comments*
RH2 : *Routing Header, type 2*
RO : *Route Optimization*
RPKI : *Resource Public Key Infrastructure*
RR : *Resource Record*
RS : *Router Solicitation*
SA : *Security Association*
SAVI : *Source Address Validation Improvements*
SAVI-B : *SAVI Binding*
SEND : *Secure Neighbor Discovery*
SEND SAVI DB : *SEND SAVI Data Base*

SEND SAVI PL : *SEND SAVI Prefix List*
SEND SAVI RL : *SEND SAVI Router List*
SIG(0) : *DNS Request and Transaction Signatures*
SLAAC : *Stateless Address Autoconfiguration*
SNMP : *Simple Network Management Protocol*
SPI : *Security Parameter Index*
TLS : *Transport Layer Security*
TP : *Trusted Port*
TSIG : *Secret Key Transaction Authentication for DNS*
TTL : *Time To Live*
ULA : *Unique Local Address*
uRPF : *unicast Reverse Path Forwarding*
VMAC : *Virtual MAC*
VP : *Validating Port*
VPN : *Virtual Private Network*