

Compromis performance/sécurité des passerelles très haut débit pour Internet

Ludovic Jacquin

▶ **To cite this version:**

Ludovic Jacquin. Compromis performance/sécurité des passerelles très haut débit pour Internet. Autre [cs.OH]. Université de Grenoble, 2013. Français. NNT : 2013GRENM041 . tel-01135182v2

HAL Id: tel-01135182

https://theses.hal.science/tel-01135182v2

Submitted on 24 Mar 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministérial : 7 août 2006

Présentée par

Ludovic Jacquin

Thèse dirigée par **Dr Claude Castelluccia**
et codirigée par **Dr Vincent Roca** et **Dr Jean-Louis Roch**

préparée au sein d' **INRIA Rhônes-Alpes, équipes Privatics et MOAIS**
et de **l'École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique**

Compromis performance/sécurité des passerelles très haut débit pour Internet.

Thèse soutenue publiquement le **20 novembre 2013**,
devant le jury composé de :

M. Pr Jean-Marc Pierson
Professeur, Université Paul Sabatier, Rapporteur et Président
M. Dr Olivier Festor
Directeur de recherche, Inria, Rapporteur
M. Pr Christophe Bidan
Professeur, Supélec, Examinateur
M. Dr Claude Castelluccia
Directeur de recherche, Inria, Directeur de thèse
M. Dr Vincent Roca
Chargé de recherche, Inria, Co-Directeur de thèse
M. Dr Jean-Louis Roch
Maître de conférences, Grenoble-INP/Université de Grenoble, Co-Directeur de thèse

Remerciements

Mes premiers remerciements sont bien évidement pour mes deux encadrants principaux, Vincent Roca et Jean-Louis Roch, qui m'ont épaulé et guidé tout au long de ma thèse. Je remercie aussi Claude Castelluccia d'avoir officiellement dirigé mes travaux.

Je remercie également les autres membres du jury de ma thèse : Dr Olivier Festor et Pr Jean-Marc Pierson, qui ont rapporté ce manuscrit ainsi que Pr Christophe Bidan qui l'a examiné. Un grand merci à Pr Kavé Salamatian d'avoir accepté de rapporter ma thèse, bien que les règles administratives ne l'aient pas permis.

Je suis reconnaissant envers le Fonds Unique Interministériel et l'Université Pierre–Mendès France qui ont financé ma thèse au travers du projet SHIVA et d'un ATER à l'IUT2 de Grenoble.

Merci à tous les partenaires du projet SHIVA pour les échanges que nous avons eu sur les nombreux sujets abordés dans ce projet. Plus particulièrement je voudrais remercier Florent Autréau, Florian Gleis et Benoît Badrignans pour nos discussions techniques toujours très fructueuses.

Je remercie grandement Cédric, Dali et Mathieu (qui m'a permis de faire cette thèse) pour leurs précieux conseils. Merci beaucoup à Fabrice, Matthieu et Max avec qui j'ai beaucoup échangé sur l'informatique en général.

Un grand merci aux membres des équipes Privatics et MOAIS que j'ai côtoyé quotidiennement pendant ces quatre années : Ferdaouss, Marie, Claude, Beri, Jonathan, Gergely, Augustin, Christophe, Joseph et tous les autres. Je remercie particulièrement Helen de m'avoir grandement facilité la vie au laboratoire.

Merci à tous les enseignants avec qui j'ai encadré pendant mon monitorat et mon ATER : Annick Montanvert, Cécile Roisin, Christophe Rippert, Franck Hetroy, François Broquedis, Jean–Sébastion Franco, Matthieu Chabanas, Olivier Muller, Pierre–François Dutot, Roland Groz, Yann Laurillau et les tous les autres.

Je remercie toute ma famille de m'avoir toujours soutenu dans mes études, et tous mes amis ainsi que les Centaures pour m'avoir permis de toujours garder un bon équilibre.

Finalement, un énorme merci à Barbara pour son support et soutien inconditionnel bien au–delà de ma thèse. Je remercie aussi Samuel dont l'arrivée a accéléré la fin de la rédaction de ce manuscrit.

Résumé

Dans cette thèse nous abordons le problème de la conception de passerelle IPsec très haut débit pour la sécurisation des communications entre réseaux locaux. Pour cela, nous proposons deux architectures : une passerelle purement logicielle sur un unique serveur, dite intégrée, et une passerelle utilisant plusieurs serveurs et un module matériel cryptographique, dite en rupture.

La première partie de nos travaux étudie l'aspect performance des deux architectures proposées. Nous commençons par montrer qu'un serveur générique est limité par sa puissance de calcul pour atteindre l'objectif de chiffrement et communication à 10 Gb/s. De plus, les nouvelles cartes graphiques, bien que prometteuses en terme de puissance, ne sont pas adaptées au problème du chiffrement de paquets réseau (à cause de leur faible taille). Nous mettons alors en place une pile réseau répartie sur plusieurs machines et procédons à sa parallélisation dans le cadre de l'architecture en rupture.

Dans un second temps, nous analysons l'intégration d'une passerelle dans un réseau, notamment l'interaction du protocole de contrôle ICMP avec IPsec. ICMP est particulièrement important pour atteindre le haut débit par son implication dans le mécanisme d'optimisation de la taille des paquets. Pour cela, nous avons développé IBTrack, un logiciel d'étude du comportement des routeurs, par rapport à ICMP, le long d'un chemin. Nous montrons ensuite qu'ICMP est un vecteur d'attaque contre IPsec en exploitant un défaut fondamental des normes IP et IPsec : le surcoût des paquets IP créé par le mode tunnel entre en conflit avec le minimum de la taille maximale prévue par IP.

Abstract

In this thesis, we explore the design of a high-bandwidth IPsec gateway to secure communications between local networks. We consider two gateway architectures: the first one, called "integrated gateway", is a purely software approach that uses a single server; the second one, called "split architecture", relies on a hardware security module and two standard servers.

The first contribution of this thesis consists in an evaluation of both architectures on the performance side. We show that a standard server lacks processing capacities to sustain 10 Gb/s networking and ciphering. Moreover, although new graphic card architectures seem promising, they are not appropriate to cipher network packets. Therefore we have designed and evaluated a prototype for the split architecture. Particularly, we show that the 10 Gb/s goal is hard to reach when using only the standards sizes and no software aggregation method, which creates jitter.

The second contribution of this thesis concerns the gateway integration inside a network, mainly at the ICMP/IPsec interaction level. Given the importance of ICMP in the Path Maximum Transmission Unit discovery (PMTUd), we developed IBTrack, a software which aims at characterizing router's behavior, with regards to their ICMP handling, along a path. Afterwards, we show that ICMP can be used as an attack vector against IPsec gateways by exploiting a fundamental flaw in the IP and IPsec standards: the IPsec tunnel mode overhead conflicts with the minimum maximal size of IP packets.

Liste des publications

Conférences et ateliers internationaux avec comité de lecture

"IBTrack : An ICMP Black holes Tracker" [65]
Ludovic Jacquin, Vincent Roca, Mohammed Ali Kaafar, Jean-Louis Roch, et Fabrice Schuler. Dans le comte rendu de *IEEE Global Communications Conference (GLOBECOM'12)*, 2012.

"Parallel arithmetic encryption for high-bandwidth communications on multicore/GPGPU platforms" [66]
Ludovic Jacquin, Vincent Roca, Jean-Louis Roch et Mohammed Al Ali. Dans le compte rendu de *International Workshop on Parallel and Symbolic Computation (PASCO'10)*, 2010.

En cours de soumission

"ICMP : an Attack Vector against IPsec Gateways"
Ludovic Jacquin, Vincent Roca et Jean-Louis Roch.

Rapports de recherche

"Implantation sur plate-forme PC standard du traitement des flux avec chiffrement simulé sur l'émulateur logiciel restreint du module SHIVA" [67]
Ludovic Jacquin et Fabrice Schuler. En tant que délivrable SHIVA № 4.1, 2011.

"Spécification des flux" [64]
Ludovic Jacquin. En tant que délivrable SHIVA № 2.1, 2010.

"État de l'art sur les serveurs réseaux 10 Gbits/sec" [63]
Ludovic Jacquin. En tant que délivrable SHIVA № 1.1, 2010.

Table des matières

Résumé ii

Abstract iii

Liste des publications iv

Liste des acronymes viii

Introduction 1

1 Vers une passerelle de chiffrement très haut débit sécurisée 3
- 1.1 Cahier des charges SHIVA . . . 3
  - 1.1.1 Présentation de SHIVA . . . 4
  - 1.1.2 Séparation rouge/noir . . . 5
  - 1.1.3 Modèle de l'attaquant . . . 6
  - 1.1.4 Contraintes et objectifs du projet SHIVA . . . 6
- 1.2 Architectures proposées pour une passerelle SHIVA . . . 7
  - 1.2.1 Normes réseaux choisies . . . 8
  - 1.2.2 Architecture intégrée . . . 8
  - 1.2.3 Architecture en rupture . . . 8
- 1.3 Méthodologie pour la spécification d'une passerelle SHIVA . . . 9

2 État de l'art des normes 11
- 2.1 CICM, interface standard pour modules de sécurité . . . 11
- 2.2 Les VPN IPsec . . . 13
  - 2.2.1 Les trois bases de données . . . 14
  - 2.2.2 Les modes . . . 14
  - 2.2.3 Les protocoles . . . 14
  - 2.2.4 L'échange de clés . . . 16
  - 2.2.5 Le chiffrement . . . 16
  - 2.2.6 Quelques attaques sur IPsec . . . 16
- 2.3 Chiffrement parallélisable . . . 16
  - 2.3.1 Le chiffrement symétrique . . . 16
  - 2.3.2 Le mode CTR . . . 17
  - 2.3.3 Le mode GCM . . . 18
- 2.4 L'environnement réseau . . . 19
  - 2.4.1 Ethernet . . . 19
  - 2.4.2 Infiniband . . . 20

I Intégration d'IPsec et performances **21**

3 Capacité de chiffrement d'une passerelle intégrée **23**
3.1 Défis pour la réalisation d'un VPN IPsec hautes performances sur une passerelle intégrée . . . 24
3.2 Possibilités de parallélisation au sein d'une passerelle intégrée . . . 26
3.2.1 Les nouvelles architectures matérielles génériques . . . 26
3.3 Évaluation de la puissance d'une passerelle intégrée . . . 29
3.3.1 Évaluation des performances natives . . . 29
3.3.2 Débit des processeurs multi-cœurs . . . 29
3.3.3 Puissance "utile" des cartes graphiques . . . 32

4 Conception et évaluation d'une passerelle en rupture **35**
4.1 Réalisation d'un VPN IPsec sur une passerelle en rupture . . . 35
4.1.1 Le routeur logiciel Click . . . 37
4.2 Parallélisation dans les modules de traitement réseau . . . 39
4.3 Évaluation de puissance d'une passerelle en rupture . . . 39
4.3.1 Méthodologie de test . . . 39

II Message de contrôle IP (ICMP) et sécurité **43**

5 Messages de contrôle réseau et performance **45**
5.1 Le protocole ICMP . . . 46
5.1.1 Format des paquets ICMP . . . 46
5.2 Les algorithmes d'optimisation de la taille des paquets . . . 47
5.2.1 Taille maximale d'un paquet . . . 48
5.2.2 Découverte du PMTU (PMTUd) . . . 48
5.2.3 Découverte du PMTU via la couche de paquetisation (PLPMTUd) 48
5.2.4 PMTUd vs PLPMTUd . . . 49
5.3 Des attaques à base de messages ICMP . . . 50
5.3.1 L'attaque Smurf . . . 50
5.3.2 L'attaque par TFN . . . 51
5.3.3 Le "Ping of Death" . . . 52

6 IBTrack : ICMP Black holes Tracker **53**
6.1 Modélisation du réseau . . . 54
6.1.1 Terminologie d'un chemin . . . 54
6.1.2 Terminologie des propriétés d'un routeur . . . 55
6.2 Algorithmes . . . 56
6.2.1 Caractérisation du comportement d'un routeur pour un protocole de test . . . 56
6.2.2 Algorithme d'affinage . . . 58
6.3 Techniques de mise en évidence d'équivalence de chemins . . . 60
6.3.1 Les routeurs "anonymes" . . . 61
6.3.2 Les tunnels . . . 61
6.4 Étude expérimentale . . . 62
6.4.1 Méthodologie . . . 62

6.4.2 Résultats 63

7 Attaque d'une implantation IPsec intégrée par exploitation des messages ICMP 67
7.1 Gestion d'ICMP par IPsec 68
7.2 ICMP comme vecteur d'attaque d'IPsec 69
7.2.1 Modèle de l'attaquant 69
7.2.2 Les attaques potentielles via ICMP 70
7.2.3 Contre-mesures 71
7.3 Déni de service sur les connexions TCP d'un utilisateur 72
7.3.1 Description de l'attaque 73
7.3.2 Attaque d'une passerelle IPsec générique 74
7.3.3 Comportement logiciel de la machine utilisateur 75
7.4 Effet de l'attaque sur un flux UDP 78
7.5 Surcoût des tunnels vs taille minimale des paquets IP 80
7.6 Recherche de PMTU dans IPsec 80

III Conclusions et persectives 83

8 Implantation d'un prototype de passerelle SHIVA 85
8.1 Choix de l'architecture 85
8.2 Prototypes de passerelle SHIVA 86
8.2.1 Prototype avec émulateur de chiffrement 86
8.2.2 Prototype avec carte de chiffrement 87

9 Enseignements et perspectives 89
9.1 Chiffrement de communications réseau à très haut débit 89
9.1.1 Nécessité d'une accélération matérielle pour le chiffrement 89
9.1.2 Sécurisation des passerelles de chiffrement par architecture en rupture 90
9.1.3 Parallélisation des traitements réseau 91
9.2 Interactions entre ICMP et IPsec 92
9.2.1 ICMP nécessaire pour des communications réseau performantes ? 92
9.2.2 Vulnérabilité des passerelles IPsec à ICMP 93

A Attaques et contre-mesures physiques I
A.1 État de l'art des principales attaques sur du matériel cryptographique . . . I
A.1.1 Attaques physiques I
A.1.2 Attaques logiques III
A.2 Considérations matérielles pour l'implantation du HSM IV
A.2.1 Gestion des clés et sessions IV
A.2.2 Contrôle et gestion d'une passerelle V
A.2.3 Initialisation cryptographique V
A.2.4 Rotation des aires de stockage V

Bibliographie VII

Liste des acronymes

ACK ”ACKnowledge“

AES ”Advanced Encryption Standard“

AH ”Authentication Header“

API ”Application Programming Interface“

ARP ”Address Resolution Protocol“

ASIC ”Application-Specific Integrated Circuit“

CICM ”Common Interface to Cryptographic Modules“

CPU ”Central Processing Unit“

CTR ”CounTeR“

CUDA ”Compute Unified Device Architecture“

DF ”Don’t Fragment“

DDoS ”Distributed Deny of Service“

DoS ”Deny of Service“

ESP ”Encapsulating Security Payload“

FAI ”Fournisseur d’Accès à Internet“

FCS ”Frame Check Sequence“

FIFO ”First In, First Out“

FPGA ”Field-Programmable Gate Array“

GCM ”Galois/Counter Mode“

GNU ”GNU’s Not Unix ! “

GPGPU ”General-purpose Processing on Graphics Processing Units“

HSM ”Hardware Security Module“

HPC ”High-Performance Computing“

IBTrack ”ICMP Black holes Tracker“

ICMP ”Internet Control Message Protocol“

IEEE ”Institute of Electrical and Electronics Engineers“

IETF ”the Internet Engineering Task Force“

IKE ”Internet Key Exchange“

IP ”Internet Protocol“

IPoIB ”IP over InfiniBand“

IPsec ”Internet Protocol Security“

ISAKMP ”Internet Security Association and Key Management Protocol“

LAN ”Local Area Network“

LRO ”Large Receive Offload“

LSO ”Large Segment Offload“

MAC ”Media Access Control“

MPLS ”MultiProtocol Label Switching“

MPTCP ”Multi-Path Transmission Control Protocol“

MSS ”Maximum Segment Size“

MTU ”Maximum Transmission Unit“

NIC ”Network Interface Card“

NIST ”National Institute of Standards and Technology“

OpenCL ”Open Computing Language“

OSI ”Open Systems Interconnection“

PAD ”Peer Authorization Database“

PCIe ”Peripheral Component Interconnect express“

PLPMTUd ”Packetization Layer Path MTU discovery“

PMTU ”Path MTU“

PMTUd ”Path MTU discovery“

PoD ”Ping of Death“

PTB ”Packet Too Big“

RFC ”Request For Comments“

RSA ”Rivest, Shamir and Adleman“

RTT ”Round Trip Time“

SAD ”Security Association Database“

SHIVA ”Secured Hardware Immune Versatile Architecture“

SIMD ”Single Instruction, Multiple Data“

SPD ”Security Policy Database“

SSH ”Secure SHell“

TCP ”Transmission Control Protocol“

TFN ”Tribe Flood Network“

TPM ”Trusted Platform Module“

TSDU ”Transport Service Data Unit“

TTL ”Time To Live“

UDP ”User Datagram Protocol“

VPN ”Virtual Private Network“

WAN ”Wide Area Network“

WiFi ”Wireless Fidelity“

Introduction

L'arrivée de l'accès à Internet partout et pour tous a révolutionné beaucoup de modes de fonctionnement à tous niveaux. D'excellentes illustrations sont la "4G" ainsi que les dernières générations de "smartphones", qui permettent actuellement en France d'avoir dans sa poche un accès Internet aussi rapide que celui de son logement et une haute qualité d'image.

L'échange d'informations sur le réseau est devenu colossal et alors que d'énormes efforts sont faits pour faciliter ces échanges, la sécurisation des informations transmises reste un vaste chantier demandant un travail pharaonique. Chaque jour offre son lot de scandales à la "Prism" et autres piratages de sites Web ou de gros volume de données personnelles (préférences en tout genre, amis, numéro de carte bancaire, personnes avec qui l'on communique, etc.) extrêmement précieuses.

Déployés à grande échelle en France (selon l'Union Internationale des Télécommunications, 83% de la population française avait accès à Internet en 2012), les boitiers - décodeurs (*set-top box*) des fournisseurs d'accès à Internet ont apporté au grand public la notion de réseau local, historiquement plutôt réservée aux entreprises. De leur côté, les entreprises utilisent désormais Internet pour connecter leurs différents sites, en lieu et place des solutions physiques (réservation de lignes de télécommunications dédiées par exemple). Il devient donc possible de décentraliser la sécurité des utilisateurs d'Internet au plus près, en chiffrant par exemple les communications à la sortie du réseau local et avant l'entrée sur Internet afin d'en dissimuler le contenu vis-à-vis de tiers espionnant les réseaux.

Parmi l'ensemble des mesures de sécurisation possibles, le travail présenté dans cette thèse adresse le chiffrement des communications utilisant les protocoles d'Internet. De plus, afin de suivre les évolutions des débits sur Internet, nous ajoutons un objectif de performance : ce travail doit viser le chiffrement de flux réseau avec un débit de 10 Gb/s (soit 500 fois plus rapide qu'un accès à Internet par "ADSL2+", ou encore 100 fois plus rapide qu'un accès par fibres optiques, communément déployées aujourd'hui).

La problématique est traitée en quatre étapes :

- définition des objectifs, présentation des solutions envisagées et des éléments existants qui seront utilisés ;
- confrontation des solutions proposées par rapport à l'objectif de performances ;
- confrontation des solutions proposées par rapport à l'objectif de sécurité réseau ;
- et enfin présentation des prototypes conçus.

Puis nous conclurons ce travail.

Chapitre 1

Vers une passerelle de chiffrement très haut débit sécurisée

.

Sommaire

1.1 Cahier des charges SHIVA 3
1.1.1 Présentation de SHIVA 4
1.1.2 Séparation rouge/noir 5
1.1.3 Modèle de l'attaquant 6
1.1.4 Contraintes et objectifs du projet SHIVA 6
1.2 Architectures proposées pour une passerelle SHIVA 7
1.2.1 Normes réseaux choisies 8
1.2.2 Architecture intégrée 8
1.2.3 Architecture en rupture 8
1.3 Méthodologie pour la spécification d'une passerelle SHIVA 9

Ce chapitre présente le projet Minalogic SHIVA, contexte et contrat support de cette thèse. Dans un premier temps le cahier des charges du projet SHIVA est présenté ainsi que le modèle d'attaque auquel SHIVA doit faire face. Deux solutions sont ensuite présentées ainsi que la méthodologie employée dans cette thèse pour choisir celle qui sera finalement implantée.

1.1 Cahier des charges SHIVA

L'environnement de communication termine sa mutation qui l'a transformé de réseaux rigides (dont l'aspect circuit limite les utilisations possibles) en un réseau ubiquitaire et tout-puissant. Son utilisation a, elle aussi, évolué passant de l'envoi d'un signal vocal à des flux vidéo haute définition sur des appareils extrêmement portatifs. Les moyens de sécurisation se doivent donc de suivre la tendance et la protection physique d'un câble ne suffit plus pour garantir la sécurité des utilisateurs.

Le projet SHIVA [26] se trouve à la jonction de ces trois constats :

– l'accès au réseau Internet est de plus en plus rapide et omniprésent ;

– les utilisateurs échangent de plus en plus et ainsi engendrent un trafic conséquent. Ainsi, alors que environ 54 exaoctets de données sont communiquées à travers Internet en 2013, CISCO [28] estime que l'on va faire face à une augmentation annuelle du trafic de 29% ;

– en conséquence des deux précédents points, le nombre de cyberattaques a fortement augmenté [30] et celles-ci ne sont plus réservées à des individus mais deviennent le champ de bataille d'États.

C'est dans ce contexte que le Ministère du Redressement Productif [9] (anciennement Ministère de l'Industrie) français et Minalogic [22] ont soutenu la conception et le développement d'une solution "forte" à la problématique de la sécurité de l'information sur les réseaux.

1.1.1 Présentation de SHIVA

SHIVA est un projet du pôle de compétitivité mondial Grenoble Rhône-Alpes Minalogic spécialisé dans les micro, nanotechnologies & logiciels. SHIVA s'appuie sur une collaboration entre entreprises (C-S [6], EASii-IC [10], Netheos [24] et iWall [17]) et laboratoires de recherche publics (Inria [13], Laboratoire d'Informatique de Grenoble [20], laboratoire TIMA [19], Institut Fourier [14], Laboratoire Jean Kuntzmann [18] et Verimag de l'université Joseph Fourier [29]).

Le but du projet est de fournir une passerelle de sécurité réseau très haut débit (10 Gb/s) programmable, reconfigurable, et basée sur des standards afin d'offrir aux entreprises, institutions et opérateurs, la possibilité de sécuriser leurs réseaux en utilisant leurs propres algorithmes de chiffrement (ou un des standards génériques existants).

En effet, il devient commun d'avoir des interfaces réseau capables de soutenir un trafic à 10 Gb/s sur les machines récentes, et l'utilisation des grilles de calcul devient alors plus aisée pour les applications gourmandes en ressources. Le besoin en sécurisation des informations devient alors crucial, comme par exemple lors de l'utilisation des grilles de calcul pour des besoins médicaux [106]. La Figure 1.1 illustre un cas d'application adressé par le projet SHIVA ; il s'agit de fournir aux utilisateurs (représentés ici par leur machine) une connexion sécurisée lors de la communication sur un réseau auquel on ne veut pas faire confiance aveuglément.

Le projet SHIVA considère uniquement une répartition statique des passerelles sur le réseau d'interconnexion. En cas de modification (ajout, suppression) d'une passerelle, les autres passerelles devront être reconfigurées par des administrateurs. Les problématiques de confiance entre passerelles ne seront donc pas adressées par des modèles et algorithmes, comme dans les travaux de Pierson et al. [59, 100], mais par intervention humaine.

Les communications réseau s'accompagnent de contraintes importantes, tout particulièrement lorsque l'on considère des réseaux à très haut débit :

– la taille maximale des trames Ethernet impose une limite assez faible sur la taille des données (comparé à ce qui est possible au niveau "supérieur") ;

– le débit génère un nombre important de paquets par seconde ;

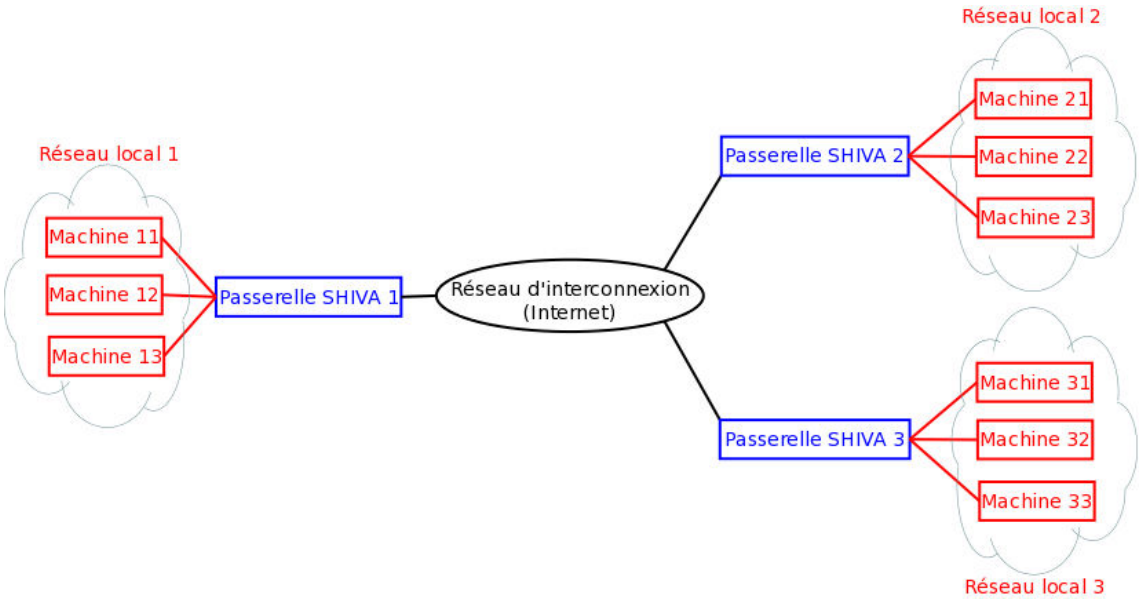


FIGURE 1.1 – Exemple d'utilisation d'une passerelle SHIVA dans une configuration à trois sites reliés par Internet.

– il ne faut pas créer de variation significative du délai de transmission des paquets d'un même flux (logique).

Bien que la garantie du respect de l'ordre des paquets ne soit pas dans les objectifs de la couche réseau, l'évolution de leur utilisation requiert désormais de prendre en compte cette contrainte. En effet, certains protocoles de transport n'ont pas la possibilité de réordonnancer les flux (UDP par exemple) et le mécanisme de récupération rapide de TCP repose sur une réception des paquets IP dans le même ordre que leur émission.

1.1.2 Séparation rouge/noir

En cryptologie, le concept rouge–noir [85] désigne la séparation des signaux contenant des informations sensibles (dits signaux rouges) de ceux transportant les informations chiffrées (dis signaux noirs). Le point critique est la séparation entres les signaux rouges et noirs.

Afin de fournir un service en phase avec le haut niveau de sécurité requis nous définissons les termes suivants :

un réseau rouge désigne un réseau local maîtrisé (on peut agir sur les machines et équipements le composant), considéré a priori de confiance. Le but général de SHIVA est de sécuriser les communications entre différents réseaux rouges distants les uns des autres ;

un réseau noir est un réseau d'interconnexion des différents réseaux rouges qui font partie du réseau privé virtuel (VPN) sécurisé qui est créé ;

une passerelle de sécurité est un équipement réseau qui en plus du routage effectue les opérations de chiffrement du VPN. Elle peut être constituée d'un ou plusieurs serveurs et éventuellement de matériels spécifiques.

Au niveau sécurité, le projet ambitionne un haut niveau de sécurité, évaluable au sens des critères communs [4] à EAL4/5 (Evaluation Assurance Level). Le niveau EAL4 correspond à un développement méthodique (selon l'état de l'art) à tous niveaux : conception, test et vérification. Pour la certification EAL5, un produit doit être conçu et testé de façon semi–formelle. Le retrait du CEA Leti [3] s'est accompagné d'une baisse du niveau de sécurité attendu.

1.1.3 Modèle de l'attaquant

Vu le niveau de sécurité visé par SHIVA (protection des communications d'un État en particulier), on considère qu'un attaquant a accès à des ressources illimitées, un temps infini ainsi qu'une fenêtre d'opportunité (attaque de type "jour 0") pour réaliser son attaque et dispose d'un haut niveau de connaissances techniques et scientifiques. Nous devons donc non seulement considérer les attaques directes, mais aussi les indirectes, c'est-à-dire les attaques qui ne donnent pas d'accès à des informations sensibles (telles que des clés de chiffrement, les informations protégées, etc) mais à des éléments qui seront utilisés ultérieurement pour continuer l'attaque. Similairement, les attaques matérielles, logicielles et logiques doivent être prises en compte. Des exemples (non exhaustifs) d'attaques dont une passerelle SHIVA devra se protéger sont listés en annexe A.1 (analyse des émissions électromagnétiques, consommation énergétique par exemple).

1.1.4 Contraintes et objectifs du projet SHIVA

1.1.4.1 Objectifs de performances

Les infrastructures réseaux évoluent désormais vers des liens à 10 Gb/s Ethernet (en particulier), et les futurs standards visent 40 Gb/s et 100 Gb/s. La passerelle SHIVA a donc pour objectif de pouvoir traiter le débit offert par les liens de plusieurs gigabits par seconde, avec un premier objectif de 10 Gb/s. Elle doit également être évolutive pour pouvoir s'adapter aux futurs standards.

1.1.4.2 Objectifs de sécurité

De plus, le niveau EAL 4/5 des critères communs se traduit par la volonté de valider certains sous-systèmes à l'aide de méthodes formelles. Les méthodes formelles sont utilisées afin de se protéger principalement des attaques dites logiques (erreurs dans le traitement d'un protocole par exemple). Dans le cas de SHIVA, cela signifie :

Vérification des protocoles
: Dans un premier temps les méthodes formelles vont servir à valider le fait que les enchaînements de messages ne peuvent pas amener le protocole dans un état inconnu ou incorrect par exemple. Puis nous chercherons à valider les implantations logicielles ou matérielles des protocoles utilisés afin de se prémunir d'attaques comme les dépassements de tampon.

Vérification des outils d'administration et de supervision
: Ils devront être validés en utilisant des méthodes formelles telles que les Méthodes B [1] ou l'assistant de preuve Coq [27]. Ces outils devront communiquer

avec une brique de gestion des droits, elle aussi vérifiée dans sa conception et son implantation, qui devra gérer certaines opérations critiques telles que :

- l'authentification des utilisateurs et la gestion de leurs droits ;
- vérification des droits d'accès préalable à l'utilisation des fonctionnalités offertes ;
- gestion de l'état général de la passerelle ;
- gestion des événements et alertes de sécurité ;
- communication avec les passerelles distantes pour leur gestion.

1.1.4.3 Contraintes de personnalisation

Il y a deux contraintes majeures pour le déploiement d'une passerelle SHIVA, une au niveau du choix des algorithmes cryptographiques et l'autre au niveau de l'interopérabilité réseau.

Les algorithmes cryptographiques doivent être aisément interchangeables au profit de n'importe quel algorithme (y compris ceux que nous ne connaissons pas) sans nécessiter l'intervention du fournisseur de la passerelle. C'est une forme de sécurité par opacification vis-à-vis du constructeur puisqu'il ne connait plus l'algorithme utilisé par les clients (contrairement aux méthodes de paramétrage des algorithmes).

L'intégration dans une infrastructure réseau d'une passerelle SHIVA doit pouvoir s'effectuer **sans modification de l'infrastructure**. Cela implique la possibilité d'utiliser les algorithmes cryptographiques déjà en place (voir le point précédent) et de pouvoir accepter les différentes technologies d'interconnexion réseau.

1.1.4.4 Contraintes de gestion des flux

Dans le modèle d'une passerelle VPN avec chiffrement, en plus du flux des utilisateurs à traiter (en chiffrement ou déchiffrement généralement) viennent s'ajouter deux grandes familles de flux. Le premier type de flux concerne les flux de contrôle cryptographique, comme les échanges de clés entre les passerelles. Pour gérer ces clés de chiffrement, des protocoles comme IKEv2 [70] sont utilisés. Enfin, le deuxième type de flux permet de contrôler les opérations effectuées par la passerelle et aussi pouvoir la gérer (configuration, mise en marche/arrêt, etc).

1.2 Architectures proposées pour une passerelle SHIVA

Pour conclure ce chapitre, nous allons présenter (sans justification) les choix des normes utilisées (une présentation détaillée sera faite au chapitre 2) et les architectures de passerelle proposées.

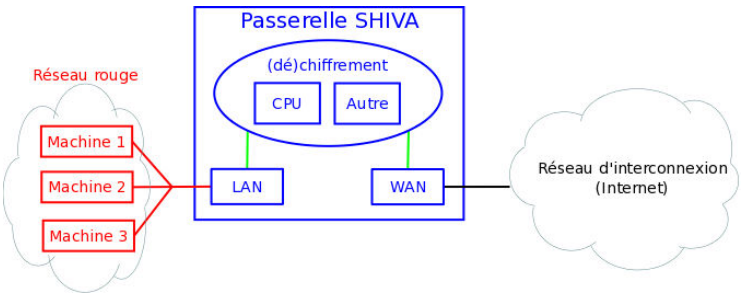
1.2.1 Normes réseaux choisies

Pour résoudre la contrainte d'intégration dans les infrastructures existantes, nous utiliserons Ethernet au niveau liaison et la sécurisation des communications se fera au niveau réseau par la mise en place d'une solution respectant la norme IPsec [73].

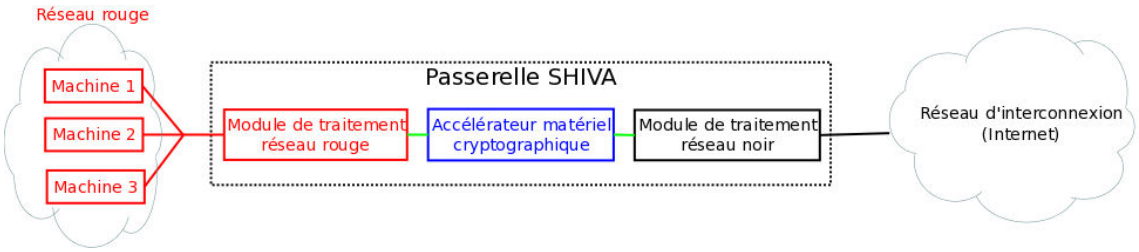
Conformément au cahier des charges SHIVA, nous allons donc proposer deux architectures (matérielles et logicielles) possibles pour la réalisation d'une passerelle de chiffrement très haut débit sécurisée.

1.2.2 Architecture intégrée

La première solution que nous avons considérée se base sur l'utilisation d'une seule machine par passerelle. C'est une approche similaire aux routeurs/VPN Cisco 7600 series [5]. Elle est connectée aux deux réseaux, privé et public, et effectue les traitements en son sein, de manière logiciel et/ou matériel (via une carte fille par exemple), tel qu'illustré en Figure 1.2a. Une implantation naïve de cette architecture est la solution logicielle reposant sur la puissance de calcul des processeurs de la passerelle. Dans la suite de cette thèse nous parlerons d'*architecture ou passerelle intégrée*.



(a) Architecture d'une passerelle intégrée.



(b) Architecture d'une passerelle en rupture.

FIGURE 1.2

1.2.3 Architecture en rupture

L'architecture en rupture est présentée sur la Figure 1.2b. Elle doit permettre de concentrer les traitements cryptographiques sur un accélérateur matériel (dit *Hardware Security Module* ou HSM) en interposant des modules de traitement réseau qui gèreront notamment l'intégration dans l'infrastructure réseau existante. Un module de plateforme de confiance (TPM, *Trusted Platform Module*) peut être vu comme une instance d'un HSM. Un TPM est un coprocesseur cryptographique permettant de stocker des clés, chiffrer des disques et

attester l'authenticité des logiciels exécutés sur une plateforme. Il a été conçu pour traiter des problèmes cryptographiques sur des cartes mères. Il réalise les principales fonctions d'un HSM mais n'est pas conçu pour chiffrer des flux réseau à très haut débit.

Cette architecture permet de séparer au maximum les tâches par type afin de permettre l'utilisation de techniques différentes pour traiter chacun des problèmes à adresser. En effet, des solutions matérielles seront plus adaptées aux questions de performance de chiffrement alors qu'on préférera des techniques logicielles pour les traitements protocolaires réseau, le logiciel étant plus adapté pour suivre les évolutions des normes.

Mais la principale nouveauté sous–jacente à cette architecture est la séparation des traitements réseau dans différents domaines de sécurité. En effet, les passerelles utilisent généralement les HSM comme simple accélérateurs pour le chiffrement et traitent les différentes étapes des protocoles réseaux (avant et après le chiffrement) au sein d'une même entité. Nous pensons qu'une telle approche permet une meilleure sécurité tout en étant capable de soutenir un trafic élevé.

Le but de cette thèse est d'étayer les avantages et inconvénients de chacune de ces deux solutions et de proposer la spécification pour l'architecture retenue. La prochaine section présente la méthode de confrontation de ces deux solutions architecturales.

1.3 Méthodologie pour la spécification d'une passerelle SHIVA

Avant de comparer les deux types d'architecture, nous rappelons dans le chapitre 2 les différentes normes qui impacteront la solution finalement choisie.

Ensuite nous étudions les deux solutions selon deux critères :

– La partie I évalue la capacité de traitement cryptographique et réseau de chaque type de passerelle.

  Le chapitre 3 traite de la version intégrée et le chapitre 4 de l'implantation en rupture.

– La partie II se concentre sur les aspects de sécurité "réseau" ; essentiellement elle analyse le comportement de chaque solution par rapport aux attaques via le protocole ICMP.

  Après la présentation d'ICMP (et des algorithmes de découverte du PMTU associés) au chapitre 5, le chapitre 6 présente l'outil IBTrack développé afin d'évaluer le fonctionnement des routeurs d'Internet par rapport à ICMP.
  Une attaque à base de messages ICMP sur les passerelles IPsec est exposée au chapitre 7 ainsi que sa mise en œuvre sur des machines Linux.

La thèse se conclue avec la partie III qui résume le choix d'architecture pour le projet SHIVA et présente les prototypes réalisés au chapitre 8 Finalement le chapitre 9 conclue en présentant les perspectives suite à cette thèse.

Chapitre 2

État de l'art des normes

Sommaire

2.1 CICM, interface standard pour modules de sécurité 11
2.2 Les VPN IPsec 13
2.2.1 Les trois bases de données 14
2.2.2 Les modes 14
2.2.3 Les protocoles 14
2.2.4 L'échange de clés 16
2.2.5 Le chiffrement 16
2.2.6 Quelques attaques sur IPsec 16
2.3 Chiffrement parallélisable 16
2.3.1 Le chiffrement symétrique 16
2.3.2 Le mode CTR 17
2.3.3 Le mode GCM 18
2.4 L'environnement réseau 19
2.4.1 Ethernet 19
2.4.2 Infiniband 20

Dans ce chapitre nous détaillons les différentes normes en rapport avec les composants cryptographiques et réseaux pour les passerelles VPN.

2.1 CICM, interface standard pour modules de sécurité

Les détails d'implantation de composants HSM sont extrêmement rares car ils sont conservés secrets par leur fabricant. La suite de cette section se concentre sur un des efforts de normalisation porté par l'initiative "Common Interface to Cryptographic Modules" (CICM) de l'IETF.

L'initiative CICM a été lancée en janvier 2010 au sein de l'organisme de normalisation IETF (qui gère principalement les protocoles réseaux de niveau 3 et plus) ; elle était portée par MITRE [23] qui est une organisation à but non lucratif américaine, qui travaille pour les intérêts publics. CICM se concentre sur un travail similaire à l'accélérateur matériel de cryptographie de la passerelle en rupture, mais centré sur le module de sécurité matériel.

Le but de CICM est de définir une API pour un HSM, qui soit indépendante du module lui-même, du fabricant, et d'un langage de programmation, mais qui puisse permettre une génération automatique des interfaces dans les langages courants. Cette norme doit donc définir des moyens d'accéder aux différents services d'un HSM :

- module cryptographique ; contrôle des accès, gestion des logiciels installés et exécutés, management de l'état du module, etc ;
- management de clés ; génération, stockage, protection, suppression de clés ainsi que tout ce qui concerne les protocoles d'échange ;
- gestion des "canaux" ; par canal, cette norme entend toute transformation cryptographique sur des données, tels que le chiffrement, le hashage, etc.

A contrario, CICM n'adresse pas les éléments suivants :

- la conception du HSM ;
- les détails spécifiques du traitement des protocoles (la norme ne traite que des entrées et sorties des messages protocolaires sur un module) ;
- la mise en place des politiques de sécurité ;
- l'organisation et traitement des informations stockées sur le HSM ;
- si un seul module est utilitsé par plusieurs instances à sécuriser, CICM ne traite pas de la séparation des données et commandes issues des différents domaines de sécurité.

Une des particularités principales de CICM est la prise en charge des HSM séparant deux domaines de sécurité, dont voici des exemples :

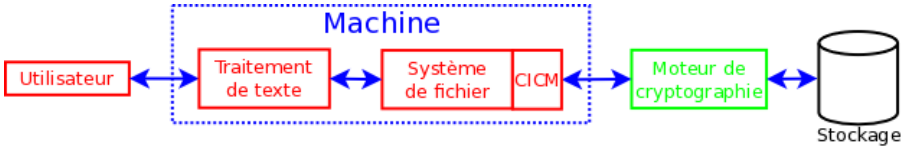
- transformation cryptographique de données d'un domaine dont le résultat est disponible dans un autre domaine de sécurité, dit information en transit. Une passerelle VPN avec chiffrement en est un très bon exemple, tout comme un transmetteur radio sécurisé (Figure 2.1a) ;
- traitement cryptographique avec utilisation du résultat au sein du même domaine de sécurité. Par exemple, des données stockées chiffrées comme présenté sur la Figure 2.1b.

Durant cette thèse nous avons participé aux discussions de CICM afin d'apporter l'expérience de notre situation qui correspond à un des cas d'utilisation possible.

Nous allons maintenant présenter IPsec, une norme permettant de sécuriser les communications réseau.



(a) Information en transit



(b) Données stockées

FIGURE 2.1 – Modèles d'utilisation de CICM

2.2 Les VPN IPsec

Le but des réseaux privés virtuels est d'étendre un réseau local au dessus d'un réseau WAN, comme Internet. Les VPN utilisent généralement des connexions point à point virtuelles via des liens dédiés ou du chiffrement. Du point de vue des utilisateurs, l'accès aux machines de leur réseau local ou via le VPN doit être identique et transparent. Ainsi toutes les machines de la Figure 2.2 doivent communiquer comme si elles faisaient partie d'un unique réseau local.

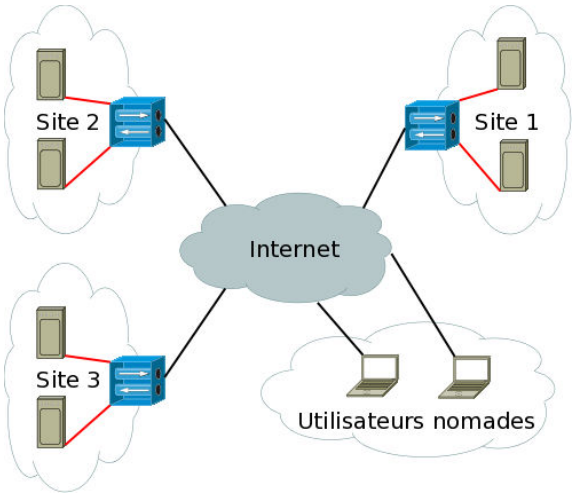


FIGURE 2.2 – Architecture générique des VPN

Il existe de nombreuses techniques pour mettre en place un VPN, mais nous nous concentrons principalement sur celles basées sur le protocole IPsec considéré dans cette thèse. En effet, IPsec est standardisé et sans brevet ce qui permet une mise en œuvre "simple" (la conception du protocole existe déjà) et économique.

IPsec [73] est un protocole de niveau réseau, complémentaire au protocole IP, qui spécifie les aspects mise en place des politiques de sécurité, encapsulation, chiffrement et traitement des données (les algorithmes de chiffrement sont aussi normés, mais de façon annexe, comme AES [43, 61] par exemple).

2.2.1 Les trois bases de données

Les implantations d'IPsec doivent mettre en place trois bases de données afin de gérer les différentes informations de configuration du VPN :

- la base de données des politiques de sécurité (SPD) contient pour chaque couple (source, destination) le traitement à effectuer (rejet, chiffrement, déchiffrement, aucun traitement, etc). La SPD doit être interrogée pour tout trafic, y compris les paquets qui ne font pas partie d'un flux protégé mais qui "traversent" une implantation d'IPsec. En effet, dans le cas où une passerelle IPsec est aussi le routeur vers Internet, il faut chiffrer le trafic entre les réseaux locaux et/ou appareils nomades, mais le trafic directement vers Internet (courriel, web, etc) ne doit pas être traité par la passerelle (mais nécessite une politique l'autorisant) ;
- la base de données des associations de sécurité (SAD) est peuplée des clés de chiffrement (généralement symétrique) utilisées pour le traitement des paquets réseau ;
- la base de données des machines autorisées (Peer Autorization Database, dit PAD) contient les identités des machines qui ont le droit de communiquer avec l'implantation d'IPsec et la manière de le faire (en terme de protocole, méthode d'authentification, etc).

Nous allons maintenant étudier les types de traitements possibles.

2.2.2 Les modes

Au sein d'IPsec, il existe deux modes qui spécifient comment est appliqué le chiffrement :

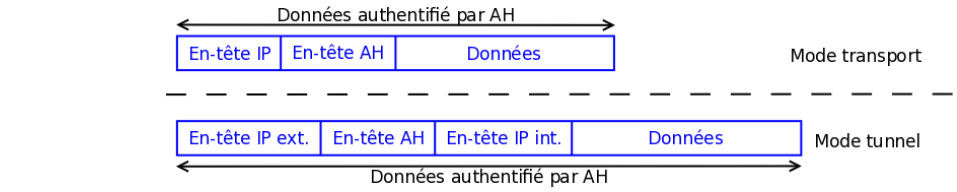
- le mode transport applique le chiffrement au paquet IP et garde l'en-tête IP initial ;
- le mode tunnel procède systématiquement à l'encapsulation du paquet IP dans un nouveau paquet IP. On parle de paquet IP intérieur lorsque l'on fait référence au paquet initial, et de paquet IP extérieur lorsque l'on fait référence au paquet encapsulant le paquet intérieur.

En pratique, seul le mode tunnel sera utilisé dans notre travail car cacher l'en-tête IP du paquet intérieur est essentiel pour garantir la confidentialité des adresses des utilisateurs.

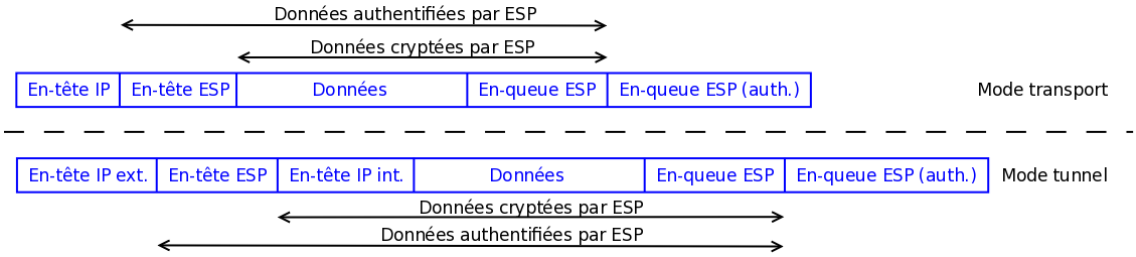
2.2.3 Les protocoles

Les modes ne fixant que les limites de l'application du traitement, IPsec utilise deux protocoles pour définir le type de service de sécurité apporté :

- le protocole AH [71] permet de garantir l'intégrité des paquets et l'authentification de la source, et peut protéger contre les attaques de type rejeu (optionnel) ;
- le protocole ESP [72] apporte en plus la confidentialité des informations traitées (selon le mode cela ne concerne pas nécessairement la totalité du paquet) et la protection contre le rejeu devient obligatoire.



(a) AH avec les modes transport et tunnel



(b) ESP avec les modes transport et tunnel

FIGURE 2.3 – Modes et protocoles d'IPsec

La protection anti-rejeu est permise grâce au champ "Numéro de séquence" des en-têtes AH (Figure 2.4) et ESP (Figure 2.5).

Les combinaisons des protocoles AH et ESP avec les modes transport et tunnel sont présentées sur la Figure 2.3.

0-7	8-15	16-31
En-tête suivant	Longueur des données	Réservé
Index du paramètre de sécurité		
Numéro de séquence		
Valeur de vérification d'intégrité		
Données		
...		

FIGURE 2.4 – Format général de l'en-tête AH

0-15	16-23	24-31
Index du paramètre de sécurité		
Numéro de séquence		
Données		
...		
Bourrage (0-255 octets)		
Bourrage	Longueur du bourrage	En-tête suivant
Valeur de vérification d'intégrité		

FIGURE 2.5 – Format général de l'en-tête ESP

En pratique, seul le protocole ESP sera considéré dans notre travail car nous voulons garantir la confidentialité.

2.2.4 L'échange de clés

Finalement IPsec définit la méthode d'échange des clés symétriques de chiffrement qui peuplent la SAD.

IPsec imposait originellement le protocole ISAKMP pour régir les communications entre machines autorisées par la SPD et la PAD à échanger des clés pour remplir la SAD. ISAKMP était seulement responsable de l'authentification des machines et de l'échange de clés, l'authentification des clés elles-mêmes étant laissée à la charge d'autres protocoles. Depuis toutes ces fonctionnalités ont été regroupées dans une seule norme : IKEv2.

Dans le cas du projet SHIVA, nous considérons uniquement le mode tunnel avec protocole ESP, et IKEv2 pour la partie échange de clés.

2.2.5 Le chiffrement

À partir de clés échangées par le protocole IKEv2, IPsec utilise un algorithme de chiffrement afin de protéger la confidentialité des flux. Il existe deux types de chiffrement :

- le chiffrement asymétrique ;
- le chiffrement symétrique.

Une rapide comparaison de la performance de ces différents types d'algorithmes, en utilisant OpenSSL 1.0.0 sur un processeur Intel core i7 nous donne un rapport de x40 pour AES 128 sur RSA 1024. Par la suite, nous ne considérerons que le chiffrement symétrique pour protéger des flux réseaux à très haut débit.

2.2.6 Quelques attaques sur IPsec

IPsec a fortement attiré l'attention des communautés sécurité et cryptologie, qui se sont principalement concentrées sur la mauvaise utilisation des primitives cryptographiques. L'équipe de Paterson [96, 95] a mis en évidence de nombreuses faiblesses dues au mauvais usage du chiffrement. D'autres travaux [86, 93] mettent en avant des attaques de déni de service (DoS) sur le mode "chiffrement seul" ; ces travaux utilisent la possibilité de modifier le paquet intérieur grâce à une attaque sur le vecteur d'initialisation.

Nous présentons maintenant l'algorithme de chiffrement que nous allons utiliser .

2.3 Chiffrement parallélisable

2.3.1 Le chiffrement symétrique

Les algorithmes de chiffrement symétrique se divisent en deux familles : le chiffrement par bloc et le chiffrement par flot. Les principales caractéristiques de ces chiffrements sont résumées dans la Table 2.1.

Les chiffrements par flot ont été très employés en télécommunication, mais la sécurité des algorithmes déployés a toujours été mise à mal : CRYPTO1 [52], A5/1 [36], RC4 [37], etc.

	Flot	Bloc
Format d'entrée	variable	fixé
Bourrage	non	oui
État interne	oui	non
Modèle de sécurité	one-time pad [103]	diffusion/confusion [51]
Implantation	parallélisable	parallélisable
Chiffement/déchiffement	identiques	différents

TABLE 2.1 – Comparatif des modes de chiffrement symétrique

Les chiffrements par bloc ont connus un grand essor avec la standardisation de l'AES [43] par le NIST. Un algorithme de chiffrement par bloc ne peut pas être utilisé en l'état. Ils sont exploités via un mode opératoire : Electronic Code-Book, Cipher Block Chaining, Cipher-FeedBack, Output FeedBack et Counter Mode (CTR). Nous écartons immédiatement le mode Electronic Code-Book, le plus simple des modes, car deux blocs identiques donnent le même chiffré, facilitant les attaques.

Par la suite nous allons nous intéresser plus particulièrement au mode CTR, car il est le plus à même d'être parallélisé.

2.3.2 Le mode CTR

Le mode CTR transforme un chiffrement par bloc en chiffrement par flot.

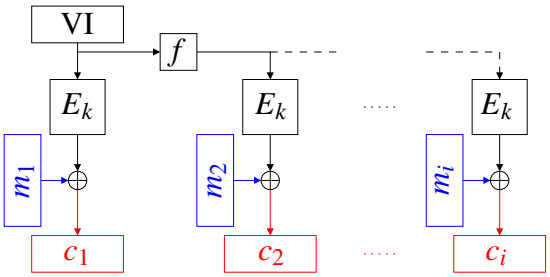


FIGURE 2.6 – Chiffrement en mode CTR

Dans la Figure 2.6, E_k est l'algorithme de chiffrement paramétré par la clé secrète k. Un compteur (obtenu grâce à f et IV) est chiffré avec E_k afin d'obtenir une suite chiffrante qui est combiné (`XOR`) au bloc du texte clair m_i pour obtenir le chiffré c_i. La valeur initiale du compteur est déterminé par un paramètre public : VI, vecteur d'initialisation. Cette valeur permet de rendre "aléatoire" le chiffrement.

Le déchiffrement (Figure 2.7) est identique : l'implantation complète du mode CTR consiste à la réalisation du compteur et de E_k.

Le calcul des valeurs du compteur et les traitements des blocs sont indépendants : le mode CTR est donc facilement parallélisable et permet des implantations haut débit [104, 76].

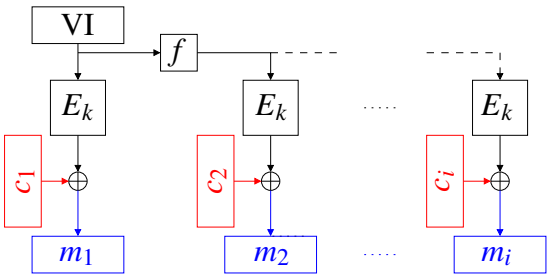


FIGURE 2.7 – Déchiffrement en mode CTR

Le mode CTR est la base des modes de chiffrement authentifié par l'ajout d'authentification. Le mode Galois compteur (GCM) permet d'authentifier les données chiffrées par le mode CTR et des données qui seront uniquement authentifiées.

2.3.3 Le mode GCM

Le mode GCM a été standardisé par le NIST [48]. Ce mode est considéré comme un des chiffrements authentifiés les plus performants.

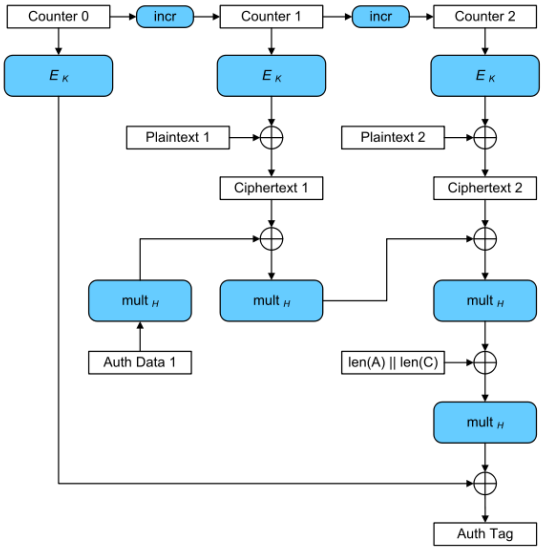


FIGURE 2.8 – Mode de chiffrement à compteur avec mode d'authentification Galois

Notons : $H = E_k(0^{128})$ le chiffré de 0 sur 128 bits, A le message clair et C le message chiffré à authentifer, m (respectivemnt n) le nombre de blocs de 128 bits dans A (resp. C), A_i (resp. C_i) l'i-ième blocs de 128 bits dans A (resp. C), u (resp. v) le nombre de bits du dernier blocs de A (resp. C) et $||$ la concaténation de bits. Le message d'authentification,

X_{m+n+1}, est alors obtenu par la formule récurrente suivante :

$$X_i = \begin{cases} 0 & \text{pour } i = 0 \\ (X_{i-1} \oplus A_i) \cdot H & \text{pour } i = 1, \ldots, m-1 \\ (X_{m-1} \oplus (A_m^* || 0^{128-v})) \cdot H & \text{pour } i = m \\ (X_{i-1} \oplus C_{i-m}) \cdot H & \text{pour } i = m+1, \ldots, m+n-1 \\ (X_{m+n-1} \oplus (C_n^* || 0^{128-u})) \cdot H & \text{pour } i = m+n \\ (X_{m+n} \oplus (longueur(A) || longueur(C))) \cdot H & \text{pour } i = m+n+1 \end{cases}$$

que l'on peut développer comme suit :

$$\begin{aligned} X_{m+n+1} = & A_1 \cdot H^{m+n+1} \oplus A_2 \cdot H^{m+n} \oplus \cdots \oplus A_{m-1} \cdot H^{n+3} \oplus (A_m^* || 0^{128-v}) \cdot H^{n+2} \\ & \oplus C_1 \cdot H^{n+1} \oplus C_2 \cdot H^n \oplus \cdots \oplus C_{n-1} \cdot H^3 \oplus (C_n^* || 0^{128-u}) \cdot H^2 \\ & \oplus (longueur(A) || longueur(C))) \cdot H \end{aligned}$$

En plus de la parallélisation des multiplications elles-mêmes [83], il est donc possible de précalculer les H^j et d'effectuer les produits $A_i \cdots H^j$ et $C_i \cdots H^j$ en parallèle.

La prochaine section présente les protocoles physiques et liaison que nous avons considéré.

2.4 L'environnement réseau

Considérons maintenant les couches basses d'une passerelle SHIVA, elles doivent permettre un interfaçage avec tout type de lien physique. Dans cette thèse (et le projet SHIVA) nous nous limitons à Ethernet et Infiniband uniquement aux niveaux physique et liaison.

2.4.1 Ethernet

Ethernet [11] est un ensemble de normes de niveau liaison et physique principalement axées sur les réseaux locaux. Développé depuis les années 1970, les normes ont connu nécessairement de nombreuses évolutions passant en particulier d'un débit d'émission de 10 Mb/s à 100 Gb/s pour les dernières normes (et des études sont menées actuellement pour préparer la prochaine itération à 400 Gb/s).

Bien que les techniques, en particulier de codage, ont évolué une contrainte majeure reste : la taille maximum du champ d'information d'une trame Ethernet est toujours de 1500 octets, comme illustré sur la Figure 2.9. Cette limitation est problématique car la plupart des traitements des nœuds réseaux intermédiaires sont indépendants de la taille des trames.

Préambule	MAC dest.	MAC source	EtherType	Données	FCS	Silence
8 oct.	6 oct.	6 oct.	2 oct.	42-1500 oct.	4 oct.	12 oct.

FIGURE 2.9 – Format de la trame Ethernet (taille en octets)

Taille des trames Ethernet

Le problème de limitation de la taille vient du champ "EtherType" (sur deux octets) des trames Ethernet. Celui-ci est en effet utilisé de deux manières : (i) pour les valeurs inférieures à 1500, cela représente la taille du champ information d'une trame Ethernet II alors que (ii) les valeurs supérieures à 1536 correspondent au type des données du champ information de la norme IEEE 802.3 (typiquement IPv4, IPv6, ARP, etc), mais alors la taille des informations est limitée à 1500 octets.

Un consensus, issu de l'industrie, augmente la taille utilisable à 9000 octets via les trames "jumbos". Cette évolution ne fait pas partie des normes à cause du conflit sur le champ "EtherType" créé. De plus, bien qu'IP autorise des paquets jusqu'à 64 Kio, cette extension officieuse d'Ethernet se limite à 9 Ko afin de ne pas avoir trop de problèmes de collisions sur le code détecteur d'erreur (FCS) qui lui reste limité à 4 octets.

Nous allons maintenant présenter une alternative issue du calcul haute performance (HPC), Infiniband, qui devrait être moins sujet aux problèmes de limitations dus à l'historique de son développement.

2.4.2 Infiniband

Infiniband a été développé dans l'optique de fournir aux nœuds de grappes de calcul des connexions à faible latence (en moyenne deux fois moins qu'Ethernet au niveau des commutateurs par exemple) et à haut débit. Sa conception a débuté à la fin des années 1990, lui permettant d'avoir des débits débutant à 2 Gb/s par ligne, avec la possibilité de paralléliser jusqu'à douze lignes. Un des inconvénients est l'absence d'API normalisée et de pile réseau.

IP sur Infiniband

Pour palier à ces absences, l'IETF a standardisé l'utilisation d'IP sur un réseau physique Infiniband [41] (IPoIB). Toutefois, comparé aux débits d'émission utilisés avec des implantations de communication HPC, IPoIB est moins performant d'un facteur 3 ou plus selon les cas.

Bilan

Malgré les performances natives élevées d'Infiniband, et la limitation de la taille des trames Ethernet qui peut être un obstacle majeur pour atteindre l'objectif de débit fixé, le déploiement à plus grande échelle d'Ethernet en fait le choix principal à considérer dans le projet SHIVA.

Aprés cette revue des différentes normes réseaux potentiellement utiles au projet SHIVA, le prochain chapitre va évaluer les capacités des nouveaux matériels lors de traitements réseau.

Première partie

Intégration d'IPsec et performances

Chapitre 3

Capacité de chiffrement d'une passerelle intégrée

Sommaire

3.1 Défis pour la réalisation d'un VPN IPsec hautes performances sur une passerelle intégrée 24

3.2 Possibilités de parallélisation au sein d'une passerelle intégrée . . . 26

3.2.1 Les nouvelles architectures matérielles génériques 26

3.3 Évaluation de la puissance d'une passerelle intégrée 29

3.3.1 Évaluation des performances natives 29

3.3.2 Débit des processeurs multi-cœurs 29

3.3.3 Puissance "utile" des cartes graphiques 32

Dans ce chapitre nous explorons les capacités des serveurs génériques dans l'optique de les utiliser comme support matériel au chiffrement de trafic réseau très haut débit. Il y est montré que malgré les efforts de parallélisation dès le niveau matériel (processeur, mémoire, carte réseau) le chiffrement (AES 128 bits) d'un flux réseau à très haut débit n'est pas immédiat : sur un serveur bi–processeurs de type Nehalem, les performances sont physiquement limitées à 5,75 Gb/s. De plus, les cartes graphiques de génération récente (avec une forte capacité de calcul parallèle) ne sont pas appropriées à notre problématique : il est possible de recouvrir une partie de la latence des communications entre mémoire centrale et graphique, mais le fonctionnement intrinsèque des unités de calcul des cartes graphiques (qui sont optimisées pour les réels) ne permet pas de chiffrer un trafic réseau à 10 Gb/s.

La loi de Moore [89] (dont la version revisitée prédit du doublement du nombre de transistors sur une puce de silicium tous les 18 mois) correspond à une augmentation annuelle de 60 % de la puissance de calcul, alors que la loi de Nielsen [92] prévoit une augmentation annuelle de 50 % du débit des communications. Pourtant, les problèmes liés au traitement du trafic réseau (routage, chiffrement, etc) laissent en suspens la question de la capacité des machines à effectuer ces traitements. Des travaux ont été effectués par Egi et al. [45, 49] dans ce sens mais se sont principalement concentrés sur les aspects routage sur serveur générique. Leurs deux principales conclusions sont :

– le routage à 10 Gb/s est possible bien qu'il requiert toutes les ressources processeur ;

– l'utilisation d'IPsec avec un chiffrement AES 128 bits fait chuter le débit utile entre 1,5 et 4,5 Gb/s.

Dans ce chapitre nous allons donc présenter la méthode d'implantation d'un VPN sécurisé par IPsec sur une architecture intégrée. Puis, après la présentation des nouvelles architectures disponibles sur les dernières générations de serveurs, nous allons mesurer les capacités des serveurs génériques à chiffrer des communications réseau. Nous attirons l'attention du lecteur sur le fait que l'évaluation faite porte sur la capacité de serveurs génériques à soutenir à la fois un trafic réseau conséquent (on vise un débit de 10 Gb/s) et son chiffrement, pas sur l'implantation complète d'un VPN sécurisé avec IPsec.

3.1 Défis pour la réalisation d'un VPN IPsec hautes performances sur une passerelle intégrée

La solution intégrée repose majoritairement sur l'exécution de logiciels sur un serveur standard. Ses principaux avantages sont la flexibilité du logiciel (en comparaison aux implantations matérielles), une grande quantité de codes disponibles et une intégration de matériels facilitée (via la mise en place de cartes filles). La solution logicielle retenue est aussi générique, avec une implantation d'IPsec de type GNU/Linux.

Les défis des traitements réseau très haut débit
Obtenir le haut débit à partir du protocole de communication standard IP/IPSec soulève plusieurs défis :

– la gestion du trafic réseau par interruptions (logicielles ou matérielles), généralement inhérentes au traitement réseau, est un facteur monopolisant fortement le processeur et donc limitant le nombre de paquets traités par seconde ;

– les traitements réseau tels que l'encapsulation ou la fragmentation de paquets impactent fortement les performances car ils nécessitent un traitement "en profondeur" du paquet ;

– la parallélisation du traitement de paquets d'un flux réseau s'accompagne de problèmes d'ordonnancement des traitements, de gestion des pipelines et tampons réseau, ainsi que l'éventuel ré-ordonnancement du flux.

Une solution logicielle, théoriquement prouvée asymptotiquement optimale, est introduite dans [35]. Elle est basée sur un algorithme de vol de travail afin de pouvoir répartir le traitement parallèle d'un flux séquentiel sur plusieurs processeurs tout en garantissant le respect de l'ordre séquentiel dans le flux de sortie. Nous appliquons ici ces résultats au cas du chiffrement d'un flux.

Borne supérieure théorique de débit pour un chiffrement parallèle

Le traitement d'un flux séquentiel imposant de respecter l'ordre (premier arrivé, premier sorti dit FIFO) sur les données constituant le flux, tout algorithme parallèle entraîne un surcoût qui se traduit à deux niveaux :

- au niveau du chiffrement du flux lui-même. En effet, certains modes de chiffrement sont intrinsèquement parallèle (comme CTR par exemple), d'autres modes ne peuvent pas être parallélisés sans surcoût (comme Cipher Feedback ou Output Feedback qui se réduisent à un calcul de préfixes pour `XOR`) ;
- au niveau des opérations de synchronisation nécessaires pour garantir l'ordre FIFO sur les blocs chiffrés. Généralement ce surcoût se traduit par l'utilisation d'un tampon intermédiaire où un bloc chiffré est stocké temporairement, puis transmis sur la sortie lorsque tous les blocs chiffrés qui le précèdent l'ont déjà été.

Aussi nous modélisons le chiffrement séquentiel d'un flux sur p unités de calcul comme suit.
Soit t_c le temps de référence (et $f_c = \frac{1}{t_c}$) la fréquence associée) pour le traitement d'un bloc unité du flux séquentiel ; un tel traitement consiste à la lecture du bloc sur le flux en entrée, son chiffrement et son écriture sur le flux de sortie, tout en minimisant le nombre d'opérations, donc sans tampon intermédiaire en particulier.
Soit t_o le temps de référence (et f_o la fréquence associée) du surcoût requis par la parallélisation du traitement (gestion du tampon intermédiaire par exemple).

Théorème 1 *Sur une machine avec p processeurs identiques, soit t_c (resp. t_o) le temps minimal de traitement séquentiel (resp. surcoût parallèle) d'un bloc unité du flux sur un processeur. Alors le temps de chiffrement de n blocs sur p processeurs est supérieur à*

$$\frac{n \cdot t_c}{p} \cdot \frac{t_c + t_o}{t_c + \frac{t_o}{p}}.$$

Cette borne inférieure est asymptotiquement atteinte par un ordonnancement aléatoire par vol de travail.

Corollaire 1 *Sur une machine avec p processeurs identiques, soit f_c la fréquence maximale de chiffrement séquentiel sur une unité et f_o la fréquence maximale du surcoût parallèle. Alors la fréquence de chiffrement sur p processeurs est inférieure à*

$$\left(p + \frac{f_c}{f_o}\right) \cdot \frac{f_c f_o}{f_c + f_o}.$$

Preuve Pour un algorithme parallèle de chiffrement donné, soit T le temps pour traiter n blocs unité sur les p processeurs. Soit x le nombre de blocs traités sans surcôut (donc lus sur le flux d'entrée, chiffrés puis écrits sur le flux de sortie comme dans le traitement séquentiel, sans passage par des tampons intermédiaires). Comme le flux d'entrée et le flux de sortie sont séquentiels, on a donc (1) : $T \geq x \cdot t_c$. Chacun des $n - x$ autres blocs a requis un traitement avec surcoût, donc un temps cumulé $(n - x) \cdot (t_c + t_o)$; par suite le temps cumulé $p \cdot T$ de traitement des $n = x + (n - x)$ blocs vérifie (2) : $p \cdot T \geq (n - x) \cdot (t_c + t_o) + x \cdot t_c = n \cdot (t_c + t_o) - x \cdot t_o$. Or, d'après (1) : $x \cdot t_o \leq T \frac{t_o}{t_c}$. Remplaçant dans (2) on obtient $\left(p + \frac{t_o}{t_c}\right) T \geq n \cdot (t_c + t_o)$, *qed*.
Le corollaire s'obtient directement en divisant par n et en passant aux fréquences.

Le théorème montre donc que l'accélération (*i.e.* le temps séquentiel divisé par le temps parallèle) est nécessairement inférieure à $\frac{p \cdot t_c + t_o}{t_c + t_o}$. En particulier dans le cas où $t_o = t_c$, l'accélération est inférieure à $\frac{p+1}{2}$.

Cette limite minimale de temps de chiffrement parallèle est atteinte lorsque tous les processeurs sont actifs. Aussi, en exploitant un ordonnancement par vol de travail adaptatif [44] dans lequel un processeur suit le chiffrement séquentiel optimal et les autres processeurs volent aléatoirement à ce processeur les blocs restant à chiffrer avec surcoût, il est possible d'atteindre asymptotiquement cette fréquence maximale. Par exemple, pour $p = 4$, $n = 40$ et en supposant que $t_c = t_o = 1$ unité de temps, le temps de chiffrement obtenu avec un tel ordonnancement est de 16 unités de temps ; alors qu'avec une parallélisation par découpe en $p = 4$ groupes de $\frac{n}{p} = 10$ blocs, chacun affecté à 1 processeur, le temps serait de $\frac{n}{p}(t_c + t_o) = 20$.

Au-delà de cette limite théorique pour un chiffrement abstrait, nous cherchons dans ce chapitre à évaluer une borne supérieure du débit pour des communications chiffrées par IPsec.

3.2 Possibilités de parallélisation au sein d'une passerelle intégrée

Nous présentons d'abord les évolutions matérielles vis-à-vis du parallélisme.

3.2.1 Les nouvelles architectures matérielles génériques

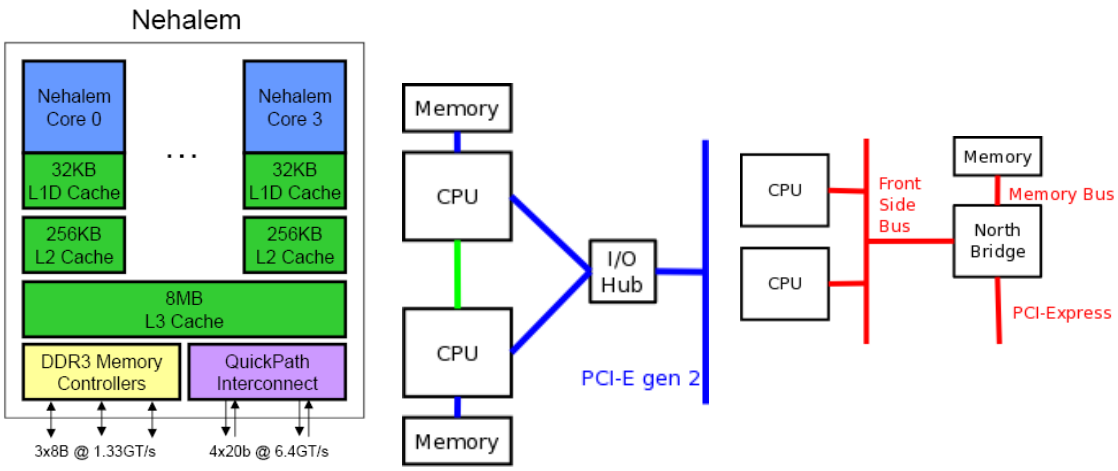
Dans un premier temps, nous nous concentrons sur les processeurs, la mémoire et les cartes réseau (ou "Network Interface Card" NIC). Ensuite nous nous pencherons sur un nouveau type de matériel présent sur les serveurs génériques : les cartes graphiques dites GPGPU.

3.2.1.1 Processeur, carte mère et mémoire

L'architecture Nehalem [16] d'Intel implante de nouveaux mécanismes de parallélisation, principalement concentrés sur l'accès à la mémoire (centrale et cache) et sur les entrées/sorties des périphériques.

Accès aux données La présence de plusieurs cœurs sur un processeur n'est pas nouvelle, mais la mise en commun du cache de niveau 3 permettant de ne pas passer par la mémoire centrale en cas de partage de données entre cœurs d'un même processeur est une nouveauté de l'architecture Nehalem. De plus, chaque processeur a accès à une banque privilégiée de la mémoire centrale (Figure 3.1a). En cas d'accès à des données dans une autre banque mémoire, ce processeur devra passer par le processeur relié à la banque mémoire concernée. Cette nouvelle organisation de la mémoire permet de retirer le goulot d'étranglement du "Front Side Bus" des anciennes architectures, illustrées par la Figure 3.1c.

Entrées/sorties et cartes filles Sur le même principe de suppression des goulots d'étranglement, l'architecture Nehalem n'utilise plus un unique port ("North Bridge") pour les accès aux cartes filles (et entrées/sorties en général) mais mutualise un hub d'entrées/sorties



(a) Architecture Nehalem de la mémoire cache

(b) Architecture Nehalem (à partir de novembre 2008) des cartes mères

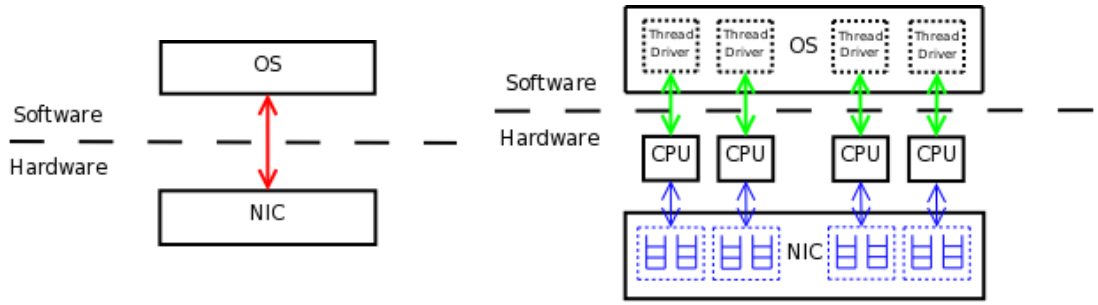
(c) Ancienne architecture (avant 11/2008) des cartes mères

FIGURE 3.1 – Évolution de l'architecture des processeurs

pour deux processeurs. La Figure 3.1b présente une architecture Nehalem à deux processeurs, mais les mêmes principes s'appliquent pour les cartes mères à quatre processeurs.

3.2.1.2 Les cartes réseau

L'arrivée de la virtualisation sur les serveurs a permis le développement de nouvelles cartes réseau favorisant le parallélisme. Il existe maintenant des alternatives au modèle classique séquentiel (Figure 3.2a), fortement basé sur les interruptions et une gestion centralisée à la fois par la carte réseau mais aussi par le système d'exploitation. En effet, les nouvelles architectures des cartes réseau [12] exploitent un système de queues matérielles ainsi que les affinités des interruptions avec un processeur. Ceci permet d'envisager un mode de fonctionnement complètement parallèle : de la carte réseau jusqu'au logiciel de traitement des paquets réseau (Figure 3.2b).



(a) Ancienne architecture des cartes réseau

(b) Nouvelle architecture des cartes réseau

FIGURE 3.2 – Évolution récente des cartes réseau

Pour ce faire, les cartes utilisent un algorithme de répartition de paquets entre les queues qui repose sur le quintuplet : (protocole transport, adresse IP source, port source, adresse IP destination et port destination). Un avantage est de permettre de garder l'ordonnancement au sein d'un flux réseau (sous réserve de ne pas utiliser d'extension telle que

"Multi-Path TCP" [34], dit MPTCP, qui délègue alors au niveau transport le gestion de l'ordonnancement du flux).

Au niveau logiciel, et particulièrement dans Linux, un travail important a été fait depuis 2001 dans l'implantation de la pile réseau afin de supprimer les interruptions au profit d'un système dans lequel le noyau demande à la carte réseau (via son pilote) si des paquets sont disponibles (mode "polling" [101]).

De plus, avec cette nouvelle pile et ces nouvelles cartes réseau, des optimisations ont été introduites : LSO pour l'envoi et LRO pour la réception. Elles permettent de traiter au niveau logiciel des tampons les plus grands possible (afin de maximiser l'amortissement du temps du temps de traitement des en–têtes) en laissant au matériel la segmentation/agrégation en paquets réseau.

3.2.1.3 Les cartes graphiques

Fortement spécialisées afin de permettre des calculs d'affichage intenses, les cartes graphiques sont devenues extrêmement puissantes, disposant de plusieurs dizaines de cœurs de calcul permettant l'exécution physiquement parallèle de milliers de threads. Initialement elles ont souvent été ignorées comme coprocesseur arithmétique. Le principal problème était leur lien avec des interfaces de programmation (API) fortement liées au monde de l'affichage graphique.

Un premier effort de l'université de Stanford permit le développement de BrookGPU [38] [2] (2004) qui bien que s'appuyant sur les librairies graphiques fournissait un environnement de développement standard aux utilisateurs, donnant naissance au traitement générique sur carte graphique (GPGPU). Les principaux constructeurs ont alors fourni des environnements standards nativement avec leur matériel ; CUDA [7] fut le premier kit de développement disponible, en 2007, et finalement un standard émergea en 2009 : OpenCL dont l'un des buts est d'unifier la programmation sur tout type d'unité de calculs. Aujourd'hui la tendance est l'intégration d'unités de type processeur dans les cartes massivement parallèles, par exemple avec l'architecture Intel ("Many Integrated Core") [15].

Le paradigme GPGPU. Bien que la programmation sur GPGPU se fasse dans des langages classiques (avec une interface `C` par exemple), un développeur doit prendre en compte les spécificités matérielles des processeurs de cartes graphiques pour en tirer pleinement partie. Le fonctionnement classique d'un programme utilisant CUDA est le suivant : le processeur transfert les données et le programme de la mémoire centrale vers la carte graphique et lance l'exécution. Après l'exécution du programme, le processeur récupère les résultats qui sont transférés de la mémoire graphique vers la mémoire centrale (Figure 3.5b). Il existe aussi des extensions permettant des accès en mémoire distante (par exemple en interfaçant une carte PCIe avec une carte Infiniband).

Au-delà de l'aspect échange des données, une des principales différences est l'utilisation des processeurs. Alors que les processeurs actuels génériques (comme les Nehalem par exemple) permettent l'exécution simultanée de plusieurs programmes sur leurs différents cœurs, le principe fondamental de la programmation sur GPGPU est d'exécuter un unique programme sur l'ensemble des cœurs (SIMD ou "Single Instruction, Multiple Data").

Bien que la comparaison en terme de performance entre processeurs et GPGPU [77] soit sujet à discussions, ce paradigme de calcul sur serveur générique est intéressant à évaluer dans notre cas d'utilisation : à savoir le chiffrement de paquets réseau. En effet, un

GPGPU peut simuler un module de chiffrement sur bus PCIe (comme la carte FPGA du partenaire Netheos de SHIVA) et permettre ainsi le prototypage pour l'intégration.

Nous allons maintenant présenter notre méthode d'évaluation de la puissance d'un serveur intégré dans le cadre du chiffrement de flux réseaux.

3.3 Évaluation de la puissance d'une passerelle intégrée

Cette section cherche à borner les capacités de la solution intégrée déployée sur serveur générique, basé sur une architecture Nehalem et équipé de cartes réseau multi-queues, comme présenté dans les sections 3.2.1.1 et 3.2.1.2.

3.3.1 Évaluation des performances natives

Dans un premier temps, nous évaluons la performance du serveur générique, tant au niveau matériel qu'au niveau logiciel. Précisément les serveurs sont basés sur deux processeurs Xeon 5530 chaque et équipés de carte réseau avec une puce Intel 82598EB (10 Gb/s et multi–queue). Au niveau logiciel, ils utilisent un noyau Linux 2.6. Nous activons donc simplement le réseau et l'implantation noyau d'IPsec. Le logiciel Iperf permet alors d'obtenir les débits atteignables au niveau applicatif entre deux serveurs (c'est-à-dire en retirant les en-têtes Ethernet, IP et TCP). Nous avons aussi développé un outil de mesure du débit de l'implantation du chiffre AES–CTR 128 bits. Les mesures sont effectuées avec une exécution de 10 secondes, et sont répétés 10 fois. La Table 3.1 résume les résultats.

TABLE 3.1 – Évaluation initiale des performances

Réseau sans IPsec	9,2 Gb/s
IPsec en mode tunnel (AES–CTR 128 bits)	0,8 Gb/s
Chiffrement AES–CTR 128 bits (mono-thread)	0,8 Gb/s

Sans IPsec, la valeur de 9,2 Gb/s correspond à la saturation physique du lien à 10 Gb/s. Mais lorsqu'IPsec est activé, le débit chute brusquement ce qui correspond aux performances de la librairie système d'AES en mode CTR (utilisé en mode mono-thread). Ceci montre l'importance qu'il y a à paralléliser (et optimiser) les traitements fortement consommateurs en puissance de calcul.

La prochaine étape consiste donc à estimer la puissance "brute" de la nouvelle génération de processeurs.

3.3.2 Débit des processeurs multi-cœurs

Vue la différence conséquente de performance sans et avec IPsec, nous allons dans un premier temps chercher à borner la puissance crête des processeurs.

3.3.2.1 Méthodologie de test

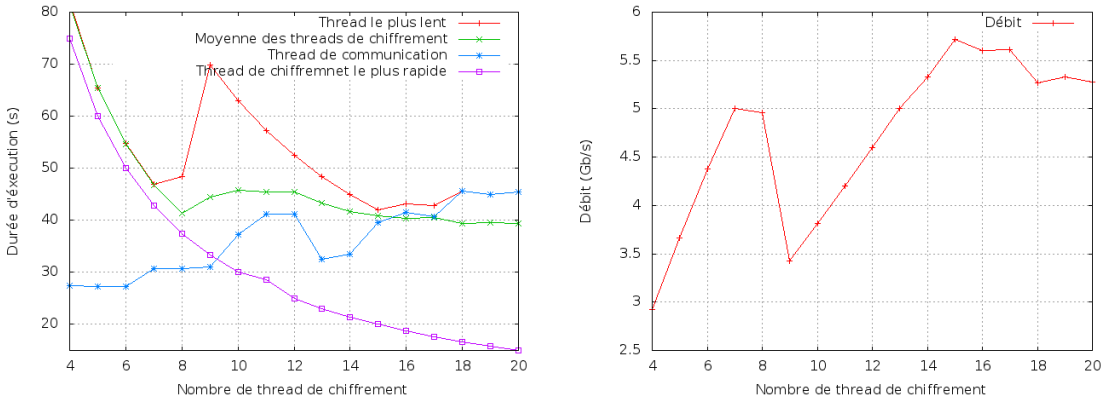
Premièrement, il faut préciser que nos machines de test ont deux processeurs à quatre cœurs, soit huit unités de calcul physique extensibles à seize logiques (via l'HyperTheading).

Notre méthodologie consiste à nous positionner dans une configuration idéale d'un point de vue de l'émetteur d'un flux IP. Nous allons simuler le chiffrement IPsec en mode tunnel par un logiciel "multi-thread" avec la répartition suivante : un thread sera dédié à la communication IP et plusieurs threads serviront au chiffrement de données. Vu l'évaluation initiale, un seul thread devrait suffire pour la communication alors qu'il en faudrait plus de dix pour le chiffrement.

Nous cherchons dans un premier temps une borne supérieure du débit atteignable, donc les données à chiffrer et celles transmises peuvent être différentes. Nous choisissons de travailler sur des données de taille 240 Gb qui seront donc réparties sur les différents threads de chiffrement, le temps minimal pour l'émission des données sera donc de 24 secondes et pour dix threads de chiffrement chaque thread nécessitera a minima 30 secondes (chaque thread devant chiffrer 24 Gb à un débit maximal de 0,8 Gb/s comme mesuré sur la Table 3.1). Nous mesurons les temps d'exécution de chaque thread, et déduisons le débit en fonction du thread le plus lent. Chaque résultat présenté est la moyenne de dix mesures. La variance observée lors de ces mesures n'apparait pas significative, nous ne l'avons pas représentée. Toutes les mesures ont été prises en compte dans la moyenne.

3.3.2.2 Optimisation du nombre de threads de chiffrement

La Figure 3.3a présente les temps d'exécution des threads, alors que la Figure 3.3b présente le débit maximal possible extrapolé à partir du thread le plus lent.



(a) Temps d'exécution des threads en fonction du nombre de thread de chiffrement

(b) Débit en fonction du nombre de thread de chiffrement

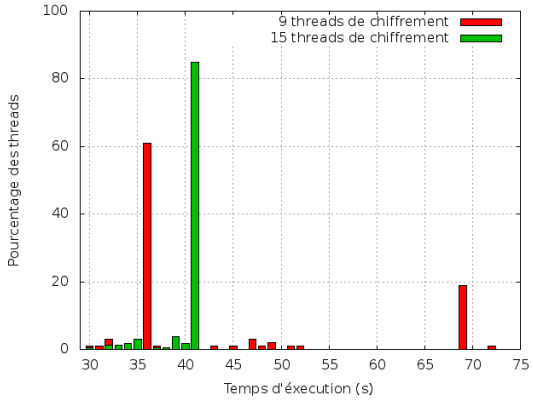
FIGURE 3.3

Deux principales observations ressortent de ces figures ; (i) on obtient un premier maxima local du débit avec 7 threads de chiffrement et (ii) au-delà de 14 threads de chiffrement, on obtient un régime stable de fonctionnement. La première observation du premier maxima avec 8 threads dont 7 de chiffrement et un autre de communication est cohérente avec la présence de 8 cœurs physiques.

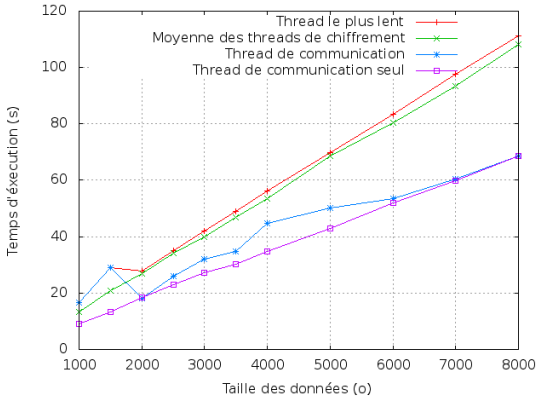
Pour analyser la chute du débit entre 9 et 13 threads, nous allons étudier plus en détail l'évolution du temps d'exécution des threads, en comparant les cas avec 9 et 15 threads

de chiffrement. Nous n'avons pas sélectionné le cas avec 7 threads de chiffrement car leurs temps d'exécution sont en fait quasiment identiques vu la proximité de leur valeur moyenne avec celle du thread le plus lent.

La Figure 3.4a représente la répartition du temps d'exécution des threads selon ces deux cas, sur un ensemble de dix mesures. On constate que la distribution est très uniforme dans le cas des 15 threads. Le débit obtenu est 15 % supérieur que pour 7 threads grâce à l'hyperthreading (8 cœurs physique, mais 16 unités de calcul au niveau logiciel). On voit que 20 % des 9 threads sont deux fois plus lent à s'exécuter. Alors que la répartition du temps de calcul est équitable dans le cas où l'on a beaucoup de threads, dans l'autre cas on observe un phénomène de famine où 20 % des threads sont exécutés de manière sérialisée avec les autres threads.



(a) Distribution du temps d'exécution des threads



(b) Temps d'exécution des threads en fonction de la taille des données

FIGURE 3.4 – Impact de la taille des données et du nombre de threads sur le temps d'exécution

Lors de ces tests nous n'avons pas mis en place d'algorithme d'ordonnancement par vol de travail pour répartir la charge de calcul sur plusieurs processeurs. La Figure 3.3a montre que le thread le plus lent est très proche du temps moyen d'exécution jusqu'à 7 threads de chiffrement. On note la même chose à partir de 15 threads de chiffrement ; de plus le thread de communication devient de plus en plus limitant lui aussi, jusqu'à être le goulot d'étranglement.

3.3.2.3 Influence de la taille des données

Finalement, nous avons étudié le comportement du chiffrement sur processeur par rapport à la taille des données à chiffrer. Jusqu'à maintenant nous utilisions une taille arbitraire de 3 ko lors du chiffrement, or la taille typique d'un paquet réseau est bien inférieure : 1480 octets (taille maximale d'un paquet chiffré par un VPN IPsec en mode tunnel sur un lien Ethernet classique). Pour ce faire, nous fixons le nombre de paquets à envoyer (700000 paquets) et nous mesurons le temps en fonction de la taille des paquets. La Figure 3.4b représente l'évolution du temps d'exécution des différents threads, en se plaçant dans le meilleur cas (c'est-à-dire 15 threads de chiffrement et 1 de communication). Comme on peut le constater, l'évolution du temps d'exécution est linéaire par rapport à la taille des données. Le temps d'exécution du thread le plus lent (chiffrement ou

communication) prend 17 secondes à 1 ko, 41 secondes à 3 ko ou encore 82 secondes à 6 ko pour s'exécuter, ce qui correspond à un débit pour l'utilisateur de 5,5 Gb/s.

Du point de vue du chiffrement des paquets, leur taille n'a pas de conséquence sur le débit du chiffrement.

En saturant les processeurs et sans tenir compte d'un ensemble de problématiques liées au réseau (ordonnancement des paquets, passage du thread de communication à un thread de chiffrement, etc), on a montré que le serveur bi–processeurs considéré n'est pas capable de soutenir un trafic supérieur à 5,75 Gb/s en mode IPsec avec chiffrement. Il faut donc trouver une autre solution.

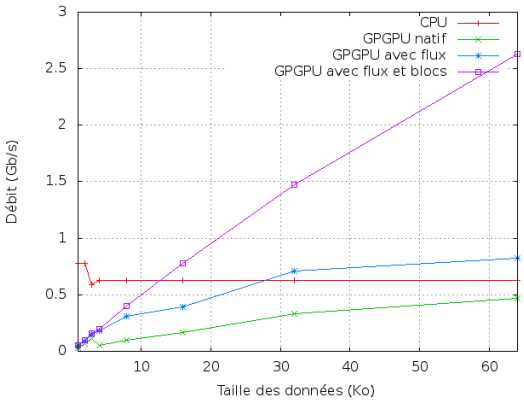
3.3.3 Puissance "utile" des cartes graphiques

Suite à la présentation du mode de chiffrement CTR (section 2.3) et du mode d'exécution en GPGPU (section 3.2.1.3), un premier niveau de parallélisation immédiat se trouve au niveau des blocs puisque leur chiffrement est indépendant. Néanmoins, les tours de chiffrement d'AES étant dépendants des tours précédents, il n'est pas possible de paralléliser à ce niveau là.

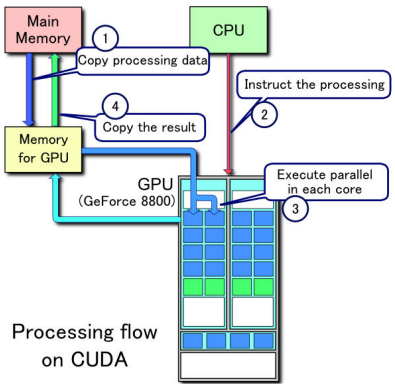
L'implantation d'AES avec CUDA [39] a été modifiée afin de prendre en compte la spécificité réseau des tailles de données (un paquet IP étant relativement petit). En effet d'une part une taille faible ne permet pas d'utiliser pleinement tous les cœurs du GPGPU et d'autre part la latence supplémentaire due aux transferts entre CPU et GPGPU n'est pas favorable au traitement de paquets réseau. Nous avons donc mis en œuvre deux techniques d'optimisation de cette implantation pour notre cas d'utilisation :

– la séparation de l'ensemble des cœurs disponibles en groupes (ou *cluster*, ensemble de cœurs travaillant sur la même zone mémoire) afin de permettre le chiffrement de plusieurs paquets réseau sur une même carte graphique ;

– un recouvrement communication/calcul : en effet l'utilisation des flux CUDA permet de transférer des données pendant que la carte graphique exécute un programme.



(a) Débits de chiffrement AES en mode CTR



(b) Flux d'exécution dans CUDA (auteur : Tosaka, Wikipedia)

FIGURE 3.5

	CPU	GPGPU	GPGPU - flux	GPGPU - flux et blocs
Débit d'AES (Gb/s)	0.63	0.47	0.82	2.63

TABLE 3.2 – Débits de chiffrement d'AES en mode CTR, pour une taille de 64 ko, sur processeur et carte graphique

La Figure 3.5a et la Table 3.2 présentent les différentes mesures du débit de l'implantation d'AES en mode CTR sur CPU (bi–Xeon 5530 à 2,4 Ghz) et sur GPGPU (GeFore GTX 295) via CUDA. La version CPU est mono-thread ce qui explique le faible débit constant. Pour les débits obtenus sur la carte graphique, on voit bien l'importance des optimisations effectuées, qui permettent d'obtenir un facteur 4 d'amélioration par rapport à l'implantation initiale pour des tampons de données de 4 ko. On monte jusqu'à un facteur 6 pour les traitements sur des données de taille 64 ko. Malgré ces améliorations, le débit de l'algorithme de chiffrement reste faible (2,6 Gb/s sur les tampons de 64 ko), ce qui oblige à agréger les paquets réseaux en groupe de 256 ko pour obtenir un chiffrement à 10 Gb/s. Une telle agrégation de paquets créerait un "jitter" (différence du temps de traitement entre des paquets d'un même flux) important. Les raisons qui limitent les performances dans notre cas sont les suivantes :

– les cartes graphiques ne sont pas optimisées pour traiter des données entières. De plus, les opérations logiques sont coûteuses en temps ;

– la fréquence des unités de calcul est faible (de l'ordre de 1 GHz), les performances venant essentiellement du nombre d'unités. Il faut donc absolument pouvoir paralléliser les traitements ;

– la latence de la communication entre la mémoire centrale et la mémoire graphique est très élevée. Malgré la possibilité de recouvrir (en partie) les communications et les calculs sur la carte graphique (étapes 1 et 4 en parallèle de la 3 sur la Figure 3.5b), le recouvrement n'est effectif qu'avec des grandes tailles de données.

Conclusions

Malgré l'attention portée à tous niveaux du matériel et du logiciel, en particulier pour bénéficier d'un parallélisme de traitement important et résoudre les points de blocage habituels (interruptions, accès mémoire, bus unique, etc.), force est de constater que l'on n'atteint pas les objectifs de haut débit avec un serveur générique, même équipé d'un accélérateur matériel de type GPGPU.. Dans un usage type réseau, la petite taille des paquets (1,5 ko au niveau Ethernet, 64 Kio au niveau IP) est un goulot d'étranglement majeur. Dans ce cas, les approches CPU sont supérieures aux approches GPGPU.

Afin de passer outre, l'agglomération des paquets pour passer à des tampons de plus grande taille pourrait être une solution dans ce contexte. En effet, il serait possible par un tel regroupement de recouvrir les temps de transferts mémoire entre processeurs et carte graphique. Cependant ce recouvrement requiert une augmentation de la latence des communications qui peut-être très pénalisante pour les applications interactives en particulier. Pour arriver à une taille agrégée de 128 ko (en cas de saturation du réseau à 10 Gb/s), il faut regrouper le trafic pendant 0,1 ms.

Néanmoins, dans ce chapitre nous avons montré qu'il était possible d'augmenter le débit de chiffrement en augmentant le nombre de cœurs dédiés au chiffrement des paquets, et ainsi d'approcher le débit crête du réseau.

Notre évaluations effectué en 2009 a montré que les serveurs souffraient donc d'une capacité de chiffrement limitée, même avec l'aide de cartes graphiques.

Mais, indépendamment des évolutions technologiques qui permettraient aujourd'hui d'atteindre les 10 Gb/s, un autre aspect critique est celui de la sécurité, le module de chiffrement (sauf sur GPGPU) n'étant pas physiquement distinct de la partie communication.

Aussi nous allons dans le chapitre suivant étudier les capacités de calcul dans le cas de l'architecture en rupture.

Chapitre 4

Conception et évaluation d'une passerelle en rupture

Sommaire

4.1 Réalisation d'un VPN IPsec sur une passerelle en rupture 35

4.1.1 Le routeur logiciel Click 37

4.2 Parallélisation dans les modules de traitement réseau 39

4.3 Évaluation de puissance d'une passerelle en rupture 39

4.3.1 Méthodologie de test 39

Dans ce chapitre nous détaillons l'architecture en rupture et son implantation avec Click. Après la mise en parallèle de notre implantation, nous évaluons les performances obtenues, en particulier dans le cas d'un grand nombre de flux différents. Un enseignement essentiel de ce chapitre est la difficulté d'obtenir un débit de 10 Gb/s lorsque les options d'agrégation/augmentation des tailles de paquets réseau et des tampons logiciels ne sont pas utilisées.

4.1 Réalisation d'un VPN IPsec sur une passerelle en rupture

Une architecture de passerelle en rupture se base sur trois entités physiques distinctes, comme illustré sur la Figure 4.1. Cela nécessite donc le développement d'une carte matérielle (HSM), son intégration dans une plate-forme constituée de serveurs, mais aussi un portage d'algorithmes cryptographiques sur la carte et leur vérification formelle, le tout avec l'objectif de pouvoir soutenir un trafic de 10 Gb/s.

Les modules de traitement réseau ont pour but de décharger le module cryptographique des traitements des protocoles réseaux. Ils réduisent ainsi la surface d'attaque du module cryptographique, tout en permettant l'utilisation de toute la puissance disponible pour les parties cryptographiques. Voici quelques exemples des tâches traitées sur ces modules :

– encapsulation et décapsulation des protocoles réseau.

– Application de politiques de sécurité simples, telle que le filtrage.

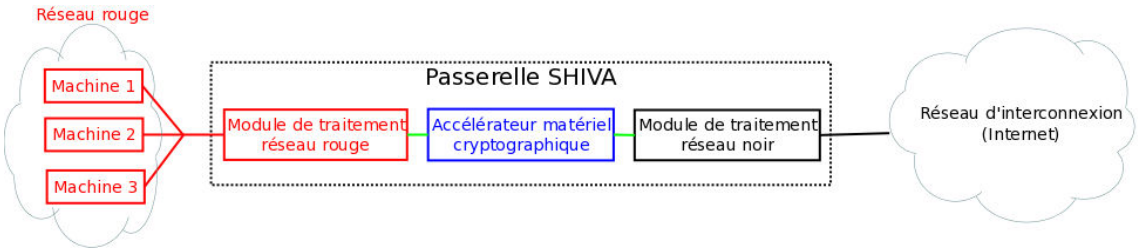


FIGURE 4.1 – Architecture d'une passerelle en rupture.

– Séparation des flux de données et mise en place de la qualité de service le cas échéant.

– Interface réseau du HSM pour la gestion de celui-ci.

Pour SHIVA, nous avons réalisé les modules de traitement réseau, sous la responsabilité du partenaire Inria. Pour la validation des développements, nous avons aussi développé un émulateur logiciel de l'accélérateur matériel.

Le module matériel de sécurité (HSM) doit inclure tous les aspects cryptographiques (gestion des certificats et des clés, stockage d'éléments sensibles, traitement des opérations cryptographiques, etc). En particulier, il implante les fonctionnalités suivantes :

– chiffrement et déchiffrement d'un flux de données (en particulier de niveau liaison ou routage).

– Gestion de fonctions cryptographiques telles que la génération de clés ou les opérations de signature.

– Marquage d'informations en fonction de leur accréditation de sécurité.

– Filtrage selon le marquage ou par autorisation basée sur source et destination.

– Accélération des traitements cryptographiques.

– Séparation des différents domaines de sécurité par architecture en rupture rouge/noir.

Nous ne rentrerons pas dans les considérations techniques du choix matériel pour l'implantation du HSM dans cette thèse. Nous considérons donc le HSM comme une boîte noire et ne détaillerons pas les particularités des modules SHIVA. Toutefois les éléments qui motivent le choix d'une implantation sur FPGA sont apportés dans l'annexe A.2. Dans le cadre du projet SHIVA, deux partenaires ont développé un prototype d'HSM ; Inria a réalisé la plateforme de démonstration en intégrant le module matériel livré.

4.1.0.1 CICM dans l'architecture en rupture

L'architecture retenue, à savoir une passerelle en rupture utilisant un HSM entre deux modules de traitement réseau, correspond partiellement au mode "information en transit" (Fig 2.1a) de l'effort de standardisation CICM. En effet, la particularité de l'architecture en rupture que nous proposons est l'accès du HSM depuis plusieurs domaines de sécurité. Cependant, la spécification de départ pour le travail de standardisation CICM modélisait

un module de sécurité comme un coprocesseur ; nous avons interagi le groupe CICM pour que la définition puisse être plus générale et abstraite pour s'adapter à un module en rupture, ce qui a été partiellement accepté. Toutefois, la proposition CICM étant en cours de définition au moment du projet, son implantation au sein de SHIVA a été repoussée (notre travail au sein du groupe a été pris en compte officiellement en juillet 2007).

Il existe plusieurs possibilités pour l'implantation logicielle des modules de traitement réseau. La première solution serait de faire une pile réseau sur mesure pour chaque module (dans le cadre de la mise en place d'un VPN de type IPsec, le fonctionnement d'une passerelle n'est pas symétrique selon le réseau par lequel un paquet arrive) mais cela serait contre-productif alors que de nombreuses implantations existent et sont éprouvées qu'elles soient propriétaires ou libres. Parmi les solutions libres nous avons concentré notre choix entre la pile réseau présente dans Linux [21] et un routeur logiciel, Click [75]. Pour des questions de portabilité et de flexibilité, nous avons décidé d'implanter la pile réseau logicielle sur Click qui est présenté dans le paragraphe suivant.

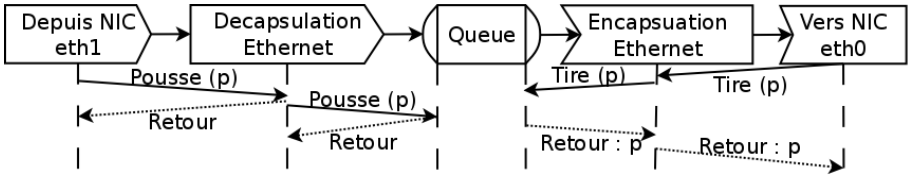
4.1.1 Le routeur logiciel Click

Click est une solution de développement logiciel, originellement pour routeur, mais qui peut-être détournée pour tout traitement réseau. Cette section introduit le fonctionnement global de Click, dont la thèse de Kohler [74] donne une description complète.

Click fonctionne par enchaînement d'"éléments" élémentaires que l'on compose. Ces éléments sont ceux qui assurent les traitements réseau et l'utilisateur gère leur ordre par configuration. Les éléments vont se concentrer sur des tâches simples et basiques telles que : classification par type, encapsulation, décapsulation, mise à jour d'un champ IP, etc. Les configurations utilisateur sont généralement représentées par un graphe dont les nœuds sont les éléments, et les arêtes les connexions entre éléments. La Figure 4.2a illustre une utilisation de Click pour mettre en place un pont Ethernet de l'interface "eth0" vers "eth1".



(a) Configuration Click d'un pont Ethernet "eth0" vers "eth1"



(b) Configuration Click d'un pont Ethernet "eth1" vers "eth0"

FIGURE 4.2

Une des caractéristiques principales des éléments est le type des entrées/sorties qui sont soit "pousse" (*push*), soit "tire" (*pull*) . Cela permet de décomposer les traitements en deux catégories : réception et émission. À l'arrivée d'un paquet, les éléments vont décapsuler les différentes couches réseau en "poussant" le paquet au protocole de niveau supérieur. Lorsqu'un paquet est émis par une application, il est mis à disposition des couches inférieures (c'est-à-dire en attente) qui, lorsqu'elles sont prêtes, vont le "tirer" depuis l'élément qui sera au final responsable de son envoi. Les deux types ne sont pas

compatibles (une sortie "pousse" doit nécessairement être connectée à une entrée "pousse"). Dans le cas d'un routeur, on fera la transition entre le mode "pousse" de réception et le mode "tire" d'émission via l'utilisation d'une queue de type FIFO. La Figure 4.2b montre l'utilisation des types des entrées/sorties dans une configuration simple.

Les traitements étant effectués par des éléments s'enchaînant selon la configuration de l'utilisateur, Click permet de construire des routeurs et d'autres applications réseau (en particulier de filtrage) très facilement, le tout avec un très haut niveau de personnalisation.

4.1.1.1 Performances de traitements réseau avec Click

Egi et al. [45, 49] ont développé Routebricks, un routeur logiciel basé sur Click, qui s'exécute sur serveur générique. Deux conclusions majeures se dégagent de ces travaux :

- Click soutient le routage à 10 Gb/s ;
- l'utilisation d'IPsec avec un chiffrement AES 128 bits fait chuter le débit entre 1,5 et 4,5 Gb/s.

4.1.1.2 Réalisation des modules de traitements réseau avec Click

Sans détailler les éléments développés (ni les en–têtes propres à SHIVA) dans le cadre du projet SHIVA, nous présentons pour exemple la configuration Click en rapport avec le traitement d'IPsec pour le module de traitement côté noir. La Figure 4.3 présente l'enchaînement des tâches.

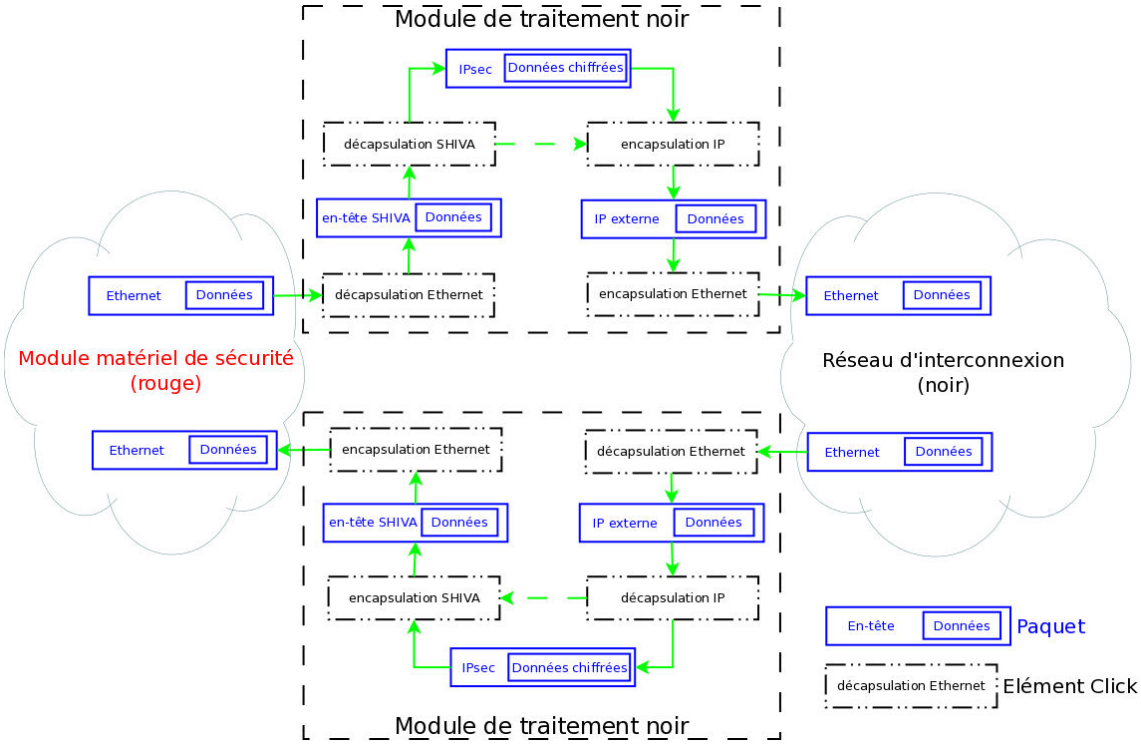


FIGURE 4.3 – Étapes du traitement des paquets IPsec dans le module réseau noir

À cette pile de traitement d'IPsec il faut rajouter la gestion du protocole de gestion des échanges de clés (dans notre cas IKE) et les parties nécessaires au bon fonctionnement réseau de la passerelle, en gérant le protocole ARP par exemple.

La partie fonctionnelle des modules de traitement réseau rouge et noir étant opérationnelle (ce point n'est pas démontré dans cette thèse), nous devons désormais garantir les performances ; nous allons donc paralléliser l'exécution de notre configuration de Click.

4.2 Parallélisation dans les modules de traitement réseau

Contrairement à l'architecture intégrée, le pipeline des unités de calcul est moins sujet au problème de saturation d'un goulot d'étranglement (dans notre cas le(s) processeur(s)), mais il faut néanmoins valider les capacités réelles de Click. Nous nous concentrons exclusivement sur les modules de traitement réseau en considérant que le HSM sera capable de supporter la charge.

Pour la parallélisation du traitement des flux, nous allons nous reposer sur la fonctionnalité multi-queues des cartes réseau. En effet, le cas principal d'utilisation envisagé consiste à agréger (ou démultiplexer) les flux de plusieurs utilisateurs d'un réseau local vers un autre.

Les optimisations que nous avons développées portent donc sur la parallélisation de plusieurs instances de Click. La prochaine section expose les résultats natifs et nos optimisations pour un module de traitement réseau implanté avec Click.

4.3 Évaluation de puissance d'une passerelle en rupture

Les serveurs utilisés pour mettre en place une passerelle SHIVA sont similaires à ceux présentés au chapitre 3 : ils intègrent des processeurs récents ainsi que des cartes réseau multi-queue permettant un parallélisme au plus proche du matériel. Du point de vue logiciel, les tests ont été réalisés sur une distribution Debian [8] 6.0.2 utilisant le noyau Linux 3.2.1 avec Click 2.0.1, soit les versions à jour et stables au moment de leurs utilisations.

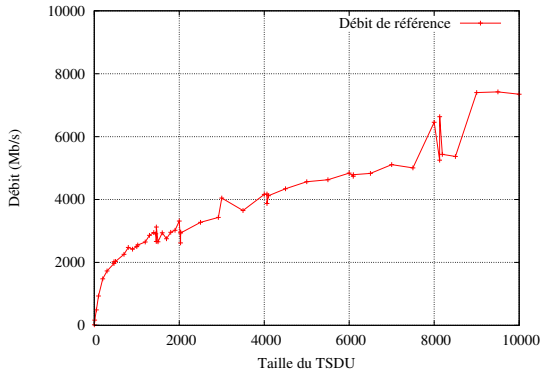
Au niveau réseau, nous avons choisi de respecter les valeurs imposées par les normes, en particulier au niveau Ethernet qui limite la taille d'une trame à 1500 octets pour le champ de données. De fait, on va donc désactiver les optimisations logicielles dont le but est d'agréger les paquets d'un même flux laissant à la carte réseau le soin de faire la séparation en trames Ethernet de taille correcte. Certes les performances sont améliorées par ces optimisations (même avec la limitation de la taille des trames par Ethernet), mais lors du développement d'une pile réseau sur mesure on rencontre alors des problèmes de taille de tampon ; bien que cela soit contournable en modifiant certains logiciels, nous avons choisi dans un premier temps de garder les versions génériques des logiciels.

Nous allons mesurer les performances de nos implantations de la passerelle en simulant un utilisateur avec une application TCP.

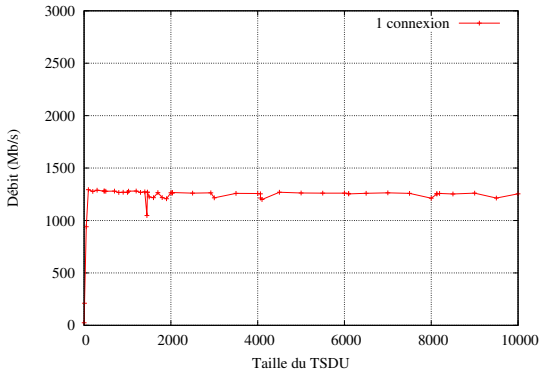
4.3.1 Méthodologie de test

La passerelle en rupture dont on souhaite évaluer les performances comporte deux cartes réseau, chacune connectée à une machine différente. Sur chacune de ces deux machines, deux applications s'exécutent : une pondeuse et un receveur de trafic réseau. L'application pondeuse est paramétrable pour contrôler la taille des données passées à la couche transport (TSDU), lors des accès en écriture sur la "socket" TCP ou UDP. Les mesures sont effectuées dix fois et chaque résultat représenté est la moyenne de ces dix mesures.

La Figure 4.4a sert de référence. Elle représente le débit entre deux machines avec une machine en mode pont Ethernet (passerelle réseau de plus bas niveau). Alors qu'il est possible de saturer un lien 10 Gb/s en connexion directe avec la configuration native (et donc les optimisations LRO et LSO activées) comme montré au chapitre 3, on voit que la présence d'une passerelle (même minimale) dégrade déjà les performances. La Figure 4.4b présente les résultats obtenus lorsque l'on met en place notre implantation naïve d'un module type passerelle avec Click.



(a) Débit de référence (deux machines sans Click)



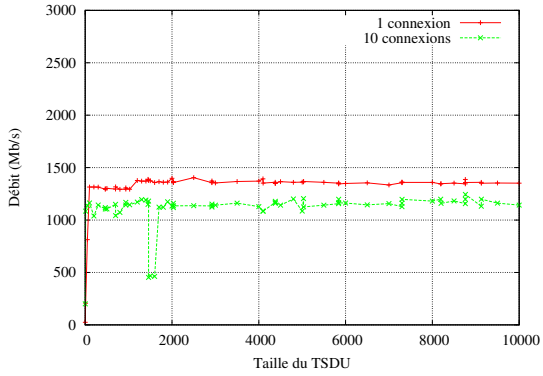
(b) Débit entre deux machines avec une seule instance monodirectionnelle de Click

FIGURE 4.4

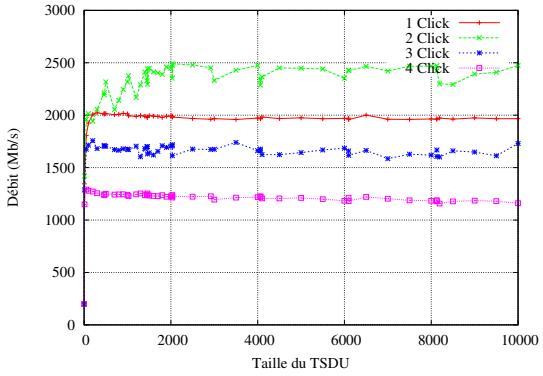
Après cette évaluation initiale de nos modules de traitement, nous avons cherché à augmenter les performances en parallélisant les instances de Click.

4.3.1.1 Optimisations et connexions multiples

Dans un premier temps, nous remarquons que l'utilisation (plus réaliste) de plusieurs connexions par l'utilisateur dégrade les performances (Figure 4.5a). Nous n'avons pas d'explication à la chute de performances pour les TSDU autour de 1500 octets dans ce cas (qui ne sera pas retrouvée dans d'autres mesures, par exemple sur la Figure 4.5b).



(a) Débit entre deux machines avec une instance de Click



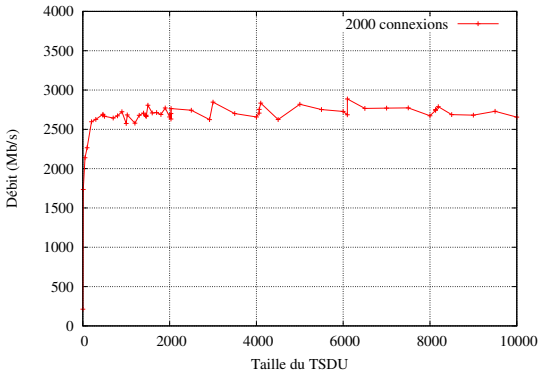
(b) Débit entre deux machines avec plusieurs instances bidirectionnelles de Click

FIGURE 4.5

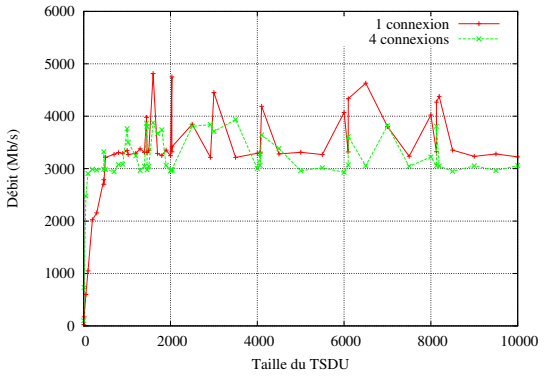
Pour pallier à cette perte de débit, nous avons donc étudié la parallélisation de Click. Une première évolution se situe au niveau de la direction des flux ; jusqu'alors chacune de nos instances de Click traitait les flux de manière bi–directionnelle. Aussi nous avons séparé nos instances en deux sous–instances qui ne traitent plus qu'une direction du flux.

Puis dans un second temps, grâce à la parallélisation dès les queues de la carte réseau, nous avons évalué l'impact de l'utilisation de plusieurs instances de Click, chacune étant liée à un ensemble de queues. La Figure 4.5b présente les résultats pour 16 connexions en fonction du nombre d'instances de Click (soit le double de processus vue la séparation effectuée).

Finalement, nous souhaitons tester notre implantation sous une charge conséquente, c'est-à-dire un très grand nombre de connexions utilisateurs. La Figure 4.6a présente le débit atteint lorsque l'on utilise 2000 connexions d'utilisateurs à travers la passerelle.



(a) Débit d'une passerelle sous fort trafic



(b) Débit du pont Ethernet, avec 1 ou 4 connexions

FIGURE 4.6

Avec un débit moyen d'utilisation de l'ordre de 2,75 Gb/s au niveau applicatif (ce qui est satisfaisant jusqu'à un TSDU de 2000 octets vue la référence, Figure 4.4a), nous allons finalement évaluer le surcoût lié à l'utilisation de Click.

4.3.1.2 Impact de Click

Finalement, pour mesurer l'influence de Click sur les performances obtenues, nous avons développé un pont Ethernet minimaliste, en réactivant les optimisations LRO et LSO. Les résultats du pont entre deux machines sont présentés sur la Figure 4.6b. Les pics de performances sont corrélés avec une taille IP maximale (64 Kio, appelée aussi trame "Super Jumbo") alors que la taille de la grande majorité des paquets est 9 ko (trame "Jumbo"). Nous ne sommes pas parvenus à reproduire systématiquement le passage de trame "Jumbo" à "Super Jumbo" dans les tests.

Conclusion

Ce chapitre termine la première partie de cette thèse qui s'est concentrée sur la présentation et les contributions aux deux architectures de passerelles envisagées dans le projet SHIVA. Le chapitre 3 a présenté notre évaluation des capacités des serveurs génériques dans le cas

du chiffrement de communications réseau. Dans le chapitre 4, nous avons exposé notre implémentation de la pile réseau avec Click dans le cadre de l'architecture en rupture.

Une des conclusions majeures est la difficulté à atteindre l'objectif de traitements réseau avec un débit de 10 Gb/s en respectant la norme Ethernet. Une autre conclusion réside dans l'importance majeure des optimisations d'agrégation/segmentation comme LRO et LSO.

La prochaine partie va se concentrer sur l'environnement réseau dans lequel opéreront les passerelles du projet SHIVA (et toute machine connectée à un réseau type Internet). On y étudiera en particulier les attaques et optimisations des performances liées aux messages ICMP.

Deuxième partie

Message de contrôle IP (ICMP) et sécurité

Chapitre 5

Messages de contrôle réseau et performance

Sommaire

5.1 Le protocole ICMP 46

5.1.1 Format des paquets ICMP 46

5.2 Les algorithmes d'optimisation de la taille des paquets 47

5.2.1 Taille maximale d'un paquet 48

5.2.2 Découverte du PMTU (PMTUd) 48

5.2.3 Découverte du PMTU via la couche de paquetisation (PLPMTUd) 48

5.2.4 PMTUd vs PLPMTUd 49

5.3 Des attaques à base de messages ICMP 50

5.3.1 L'attaque Smurf 50

5.3.2 L'attaque par TFN 51

5.3.3 Le "Ping of Death" 52

Dans ce chapitre nous présentons le protocole de messages de contrôle utilisé dans les réseaux IP, les deux principaux algorithmes de recherche de la taille optimale de paquet (PMTUd et PLPMTUd), ainsi que des attaques classiques utilisant ICMP.

ICMP (*Internet Control Message Protocol*) est un protocole de signalisation pour les réseaux basés sur IP, ce n'est pas un protocole permettant le transfert d'informations mais il reste essentiel au fonctionnement des réseaux IP. Il est en effet en charge de l'échange des informations de l'état du réseau et des messages de contrôle ou d'erreurs entre les différents nœuds.

Après une présentation générale d'ICMP, nous présenterons son usage dans les algorithmes de recherche de taille optimale de paquets afin d'optimiser l'utilisation des liens physiques.

5.1 Le protocole ICMP

Existant en deux versions, pour IPv4 [98, 55] et IPv6 [42], le protocole a connu de nombreuses évolutions au cours du temps - essentiellement dans le traitement des différents types de paquets par les nœuds du réseau. Cette section récapitule le fonctionnement général d'ICMP et les principaux types de messages du protocole.

5.1.1 Format des paquets ICMP

Il est à noter que bien qu'étant un protocole de niveau réseau (tout comme IP), les paquets ICMP sont encapsulés et acheminés via le protocole IP.

La structure générale des paquets ICMP est précisée sur la Figure 5.1. On notera que les champs de l'en-tête peuvent être suivis d'une partie des données du paquet IP ayant déclenché la génération du paquet ICMP par exemple (en fonction du type et code ICMP du paquet).

0-7	8-15	16-23	24-31
Type	Code	Code de contrôle d'erreur	
Suite de l'en-tête			
Données (optionnelles et de taille variable)			

FIGURE 5.1 – Format général des paquets ICMP

Nous allons détailler pour chaque type de message sa signification, son importance et ses différents codes. Ces messages peuvent être classés en deux catégories [54] : les messages d'erreurs et d'informations.

5.1.1.1 Messages ICMP d'erreur

Destination injoignable. Ces paquets sont envoyés en cas d'impossibilité d'acheminer un paquet (hors congestion du routeur et taille de paquet trop grande), de filtrage ou de problème d'adresse ou de port.

Paquet trop gros ("Packet Too Big" - PTB). Ce type de paquet ICMP est extrêmement important car il sert à un des algorithmes de maximisation de la taille des paquets qui sera présenté à la section 7.3.3.1. Il est généré lorsqu'un routeur ne peut pas propager un paquet à cause d'une taille trop grande par rapport à l'interface de sortie. Dans le cas d'IPv6 c'est assez simple puisque les routeurs n'ont pas la possibilité de fragmenter les paquets en transit ; pour IPv4, cela va dépendre des options IP mises en place par l'émetteur (bit d'interdiction de fragmentation - dit DF - mis à 1). Ces paquets ICMP utilisent les 16 bits de poids faible de la suite de l'en-tête pour préciser la taille maximale supportée (MTU, *Maximum Transmission Unit*).

Dépassement du temps. Si le champ durée de vie (TTL de l'en-tête IPv4 ou HopLimit d'IPv6) expire lors de l'acheminement du paquet (c'est-à-dire que le paquet a traversé plus de routeurs qu'autorisé par l'émetteur), le routeur intermédiaire renvoie ce type de paquet ICMP.

Problème de paramètre. Un paquet ICMP de ce type est émis en cas d'erreur dans l'en-tête IP, comme une option ou un champ inconnu.

5.1.1.2 Messages ICMP d'information

Requête et réponse d'écho. Ces messages sont échangés entre machines afin de tester leur connectivité ; ce type est en particulier utilisé par l'utilitaire `ping`.

Découverte et annonce de routeur. Ces paquets permettent à des machines d'un réseau d'annoncer l'existence d'une route plus directe pour atteindre d'autres réseaux.

Redirection. Ce type sert essentiellement entre routeurs afin de préciser qu'une meilleure route existe pour la destination du paquet.

Nous voyons que le protocole ICMP a des usages multiples. Dans la suite, nous nous concentrons sur l'utilisation d'ICMP dans les algorithmes d'optimisation de la taille des paquets IP, ce qui influence directement les performances en terme de débit atteignable.

5.2 Les algorithmes d'optimisation de la taille des paquets

La Figure 5.2 présente le débit utile pour un protocole transport utilisant IP (sans option) sur un lien Ethernet à 10 Gb/s en fonction de la taille des trames Ethernet, dont voici la formule de calcul du pourcentage :

$$\frac{taille_{données\ \mathrm{IP}}}{taille_{données\ \mathrm{IP}} + taille_{en-tête\ \mathrm{IP}} + taille_{encapsulation\ Ethernet}}$$

$$\text{avec } taille_{en-tête\ \mathrm{IP}} = \begin{cases} 20 \text{ octets pour IPv4} \\ 40 \text{ octets pour IPv6} \end{cases}$$

$$taille_{encapsulation\ Ethernet} = 38 \text{ octets}$$

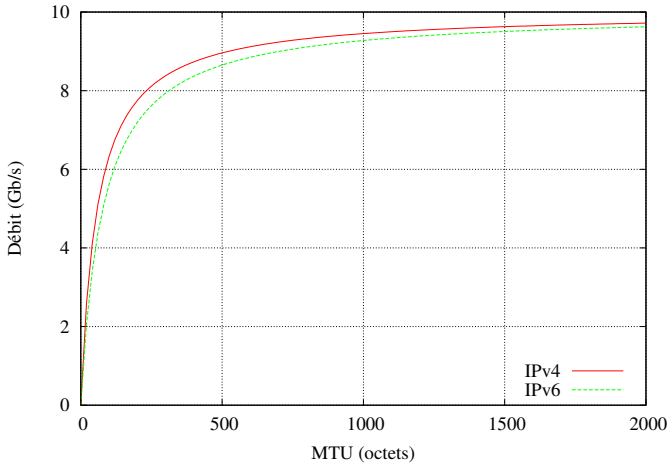


FIGURE 5.2 – Débit IP selon le MTU d'un lien Ethernet 10 Gb/s.

La différence de débit est uniquement due aux en-têtes Ethernet et IP. Bien qu'IP impose la taille minimale qu'un lien physique doit accepter, à savoir 576 octets en IPv4 et 1280 octets pour IPv6, il reste une marge d'optimisation (environ 10 %) à exploiter. Ce gain prend uniquement en compte l'utilisation optimale du lien ; d'un point de vue plus global, il faudra prendre en compte les traitements sur les routeurs du réseau.

Avant d'introduire les algorithmes d'optimisation, nous allons préciser le lexique.

5.2.1 Taille maximale d'un paquet

Comme illustré sur la Figure 2.9 de la section 2.4.1, la taille des données est variable (de 42 à 1500 octets pour Ethernet), mais est bridée par la valeur MTU qui reflète la taille maximale en octets d'un paquet sur le lien physique.

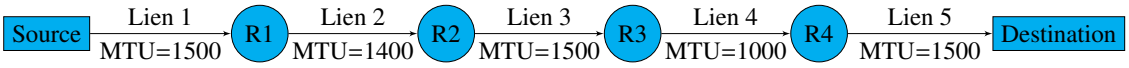


FIGURE 5.3 – Route avec plusieurs MTU

Lorsque l'on considère une route constituée de plusieurs liens, on appelle PMTU le plus petit MTU des liens (sur la Figure 5.3, le lien 4 est le lien limitant et impose un PMTU égal à 1000 octets). Tout paquet de taille plus petite que le PMTU pourra donc être acheminé sur chacun des liens sans erreurs.

5.2.2 Découverte du PMTU (PMTUd)

Le premier algorithme, appelé PMTUd [88], utilise fortement les paquets ICMP PTB et fonctionne sous forme d'essais consécutifs.

Lors de l'émission d'un paquet IP (pour IPv4 il faut mettre le bit "ne pas fragmenter" sinon les routeurs intermédiaires vont fragmenter le paquet), la source est forcément limitée par le MTU du lien physique sur lequel elle émet. Ensuite chaque routeur sur le chemin soit propage ce paquet, soit retourne un paquet ICMP PTB si le paquet IP est trop gros pour le lien suivant. L'émetteur pourra alors ré–émettre le paquet avec une taille corrigée. Lorsque l'on arrivera au destinataire, on obtiendra alors le PMTU, c'est-à-dire le minimum des MTUs de chaque lien entre la source et la destination. Les futurs paquets IP n'auront plus de problèmes de taille lors de l'acheminement.

5.2.3 Découverte du PMTU via la couche de paquetisation (PLPMTUd)

L'autre algorithme, dit PLPMTUd [84], peut reposer sur plusieurs protocoles dont certains de niveau transport ou "supérieur".

Pour ce faire, PLPMTUd utilise des "sondes" pour valider l'acheminement d'un paquet. La principale différence entre PMTUd et PLPMTUd vient des acteurs impliqués : PMTUd repose sur les routeurs intermédiaires en cas de problème de taille de paquet alors que PLPMTUd est prévu pour ne faire intervenir que la source et la destination dans le protocole.

5.2.3.1 Sondage du chemin

Pour valider l'utilisation d'une taille de MTU, PLPMTUd envoie une sonde de la taille souhaitée puis attend l'acquittement de sa bonne réception par la destination. Si cet acquittement arrive, la source considère que le chemin supporte cette taille de paquet IP. Le processus de sondage PLPMTUd continue, et de nouveaux paquets de taille supérieure sont envoyés afin d'ajuster le MTU au PMTU réel ; si l'acquittement n'est pas reçu, la source considère que le chemin ne supporte pas cette taille. On notera que PLPMTUd doit être configuré avec deux bornes (l'une inférieure et l'autre supérieure) que l'algorithme ne dépasse jamais.

5.2.3.2 Type d'acquittement

C'est ici que réside l'aspect multi–protocoles, PLPMTUd peut reposer sur tout protocole permettant l'acquittement des sondes quel que soit son niveau : transport, applications, etc. Le premier protocole utilisé est naturellement TCP dont les paquets avec le bit ACK peuvent servir d'acquittement aux sondes (qui seront dans ce cas précis aussi des paquets transportant des données).

5.2.4 PMTUd vs PLPMTUd

Nous allons désormais comparer les caractéristiques de chacun des deux algorithmes de recherche du PMTU, résumées par la Table 5.1.

5.2.4.1 Centralisation des algorithmes

PMTUd adopte une approche décentralisée, laissant le soin à chaque routeur d'annoncer (si besoin) le MTU de ses liens. Il fonctionne en considérant le PMTU comme le plus petit des MTU du chemin.

A contrario, PLPMTUd cherche à définir directement le PMTU du chemin, sans recourir à la connaissance de tous les MTU. Son approche est centralisée, s'appuyant uniquement sur la source et la destination.

PMTUd nécessite une configuration ICMP fonctionnelle sur tous les routeurs du chemin et garantit le PMTU "réel". PLPMTUd repose uniquement sur la source et la destination pour fournir une évaluation du PMTU.

5.2.4.2 Réactivité en cas de modification de la route source–destination

PMTUd utilisant les routeurs directement connectés au lien pour connaître son MTU, en cas de modification du lien ou de la route, le(s) routeur(s) concerné(s) informeront immédiatement la source en cas de réduction du PMTU. Un compte à rebours (10 minutes dans Linux par exemple) est généralement utilisé pour tester si le PMTU a augmenté.

PLPMTUd reposant sur un autre protocole pour gérer les sondes (envoi, acquittement, etc), il détectera un changement de PMTU via ce protocole ne validant pas la réception d'une sonde. Dans le cas de l'implantation en utilisant TCP, c'est le compte à rebours pour réception de l'ACK qui sera utilisé ; il est généralement fixé à deux fois le temps nécessaire pour un échange et acquittement entre la source et la destination (RTT) - soit d'un ordre inférieur à 1 seconde sur Internet normalement.

PMTUd réagit immédiatement en cas de réduction du MTU alors que PLPMTUd nécessite l'expiration d'un compte à rebours.

5.2.4.3 Intégration dans la pile protocolaire

PMTUd repose uniquement sur ICMP (et IP) pour fonctionner ; il est donc particulièrement sensible à la gestion d'ICMP par les routeurs sur la route.

PLPMTUd s'appuie sur un protocole contenant un mécanisme d'envoi de paquet et d'acquittement, dont l'exemple le plus courant est TCP. Néanmoins, des effets de bords peuvent apparaître car le mécanisme d'acquittement peut être utilisé à d'autres fins, en particulier pour le contrôle de congestion dans TCP.

5.2.4.4 Utilisation dans un VPN

Malgré la séparation du réseau en deux parties distinctes (les réseaux "locaux" et le(s) réseau(x) d'interconnexion), la route entre source et destination traverse les deux parties. En conséquence, PMTUd a besoin que l'implantation du VPN propage les paquets ICMP PTB du réseau d'interconnexion au réseau local de la source.

PLPMTUd n'utilisant que la source et la destination dans son fonctionnement, il ne requiert pas de traitement particulier par un VPN, hormis le transfert des paquets.

	PMTUd	PLPMTUd
Acteurs impliqués	source et routeurs intermédiaires	source et destination
Réactivité au changement	immédiate	dépendante de l'implantation
Protocole(s) utilisé(s)	ICMP	dépendant de l'implantation
Intégration dans un VPN	dépendante de l'implantation	transparente

TABLE 5.1 – Comparatif des caractéristiques de PMTUd et PLPMTUd

Nous étudions maintenant l'aspect sécurité d'ICMP, en particulier quelques attaques classiques.

5.3 Des attaques à base de messages ICMP

Dans cette section nous décrivons trois attaques par messages ICMP : l'attaque Smurf [31], l'attaque par TFN [32] et le "Ping of Death" [50]. Les deux premières visent du déni de service (DoS) alors que la dernière attaque l'implantation d'ICMP sur une machine.

5.3.1 L'attaque Smurf

Cette attaque est une version plus évoluée du classique "*ping* flood" [94]. Plutôt que de simplement noyer la cible de message ICMP "echo request", l'attaquant utilise les machines d'un réseau pour amplifier l'attaque : au lieu d'avoir une seule machine (celle de l'attaquant) envoyant des messages ICMP "echo request" toutes les machines du réseau envoient des messages ICMP "echo reply". On parle alors de déni de service distribué (DDoS).

Dans le détail (illustré par la Figure 5.4), l'attaquant doit tout d'abord identifier un réseau (au sens IP) pour lequel un routeur d'accès autorise les messages ICMP "echo

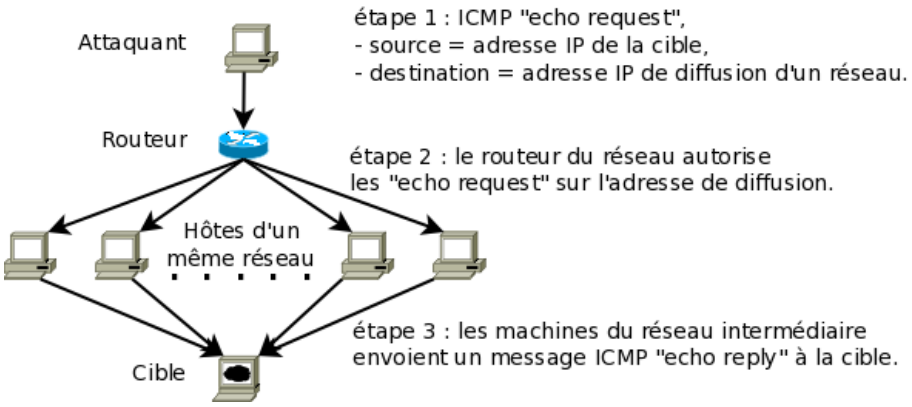


FIGURE 5.4 – Attaque Smurf.

request" sur l'adresse de diffusion. Puis, il envoie sur cette adresse de diffusion un message ICMP "echo request" (qui sera donc transmis à toutes les machines du réseau) en changeant l'adresse IP source du paquet par l'adresse IP de la cible (étape 1). Le routeur transmet le message aux machines du réseau qui répondent un message ICMP "echo reply" à destination de la source du paquet reçu : donc à l'adresse IP de la machine cible qui se retrouve submergée de messages ICMP.

5.3.2 L'attaque par TFN

Un "Tribe Flood Network" est un ensemble de machines, dont le but est de mettre en place des attaques DDoS. Un TFN est généralement composé de deux niveaux hiérarchiques : un chef (ou centre de commande) et des hôtes TFN. Comme l'illustre la Figure 5.5, le chef du TFN commande les hôtes (par exemple via des messages ICMP "echo reply") qui attaquent alors la cible. Les hôtes ont plusieurs moyens d'attaque pour parvenir au DDoS dont les messages ICMP "echo reply" ou "echo request" par exemple. Les hôtes du TFN sont généralement des machines infectées par un botnet qui agissent à l'insu de leur propriétaire.

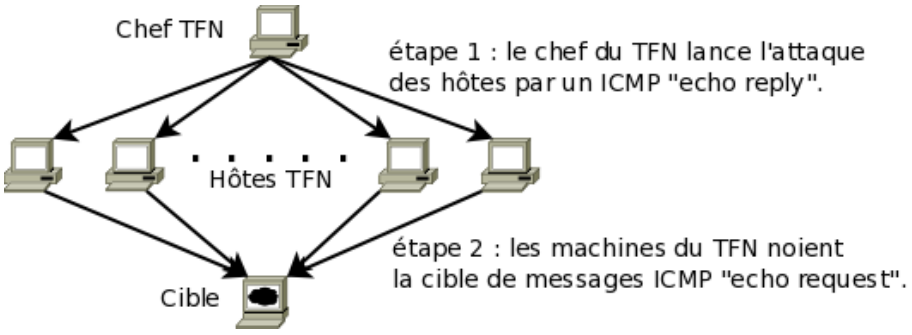


FIGURE 5.5 – Attaque par TFN.

L'architecture des TFN a depuis été largement reprise, en particulier pour les attaques plus ciblées : désormais pour voler des informations les attaquants utilisent un (ou plusieurs) centre de commande qui gère à distance des machines infectées. Ces machines, qui ont accès aux données qui intéressent l'attaquant, récupèrent les informations puis les envoient à l'attaquant.

5.3.3 Le "Ping of Death"

Le "Ping of Death" (PoD) illustre une autre classe d'attaques possibles par messages ICMP : l'attaque de l'implantation de la pile réseau. Cette attaque, assez simple, consiste à envoyer, sous forme fragmentée, un message ICMP (historiquement "echo request") dont la taille est supérieure à la taille IP maximale (2^{16} soit 64 Kio). Après réception des paquets IP, une mauvaise implantation de la pile réseau (c'est-à-dire qui gère mal les trop grandes tailles dans ce cas) peut mener à une défaillance du système ; dans certains cas, la défaillance se manisfestait par un arrêt de la machine.

Ce chapitre a présenté le protocole ICMP et deux applications antagonistes : l'utilisation d'ICMP dans l'algorithme standard d'optimisation de la taille des paquets (PMTUd) et quelques attaques classiques qui utilisent ICMP.

La suite de cette partie montre comment utiliser ICMP afin d'améliorer le débit de communication et son importance dans la sécurité des VPN, en particulier ceux basés sur IPsec.

Chapitre 6

IBTrack : ICMP Black holes Tracker

Sommaire

6.1 Modélisation du réseau 54
6.1.1 Terminologie d'un chemin 54
6.1.2 Terminologie des propriétés d'un routeur 55
6.2 Algorithmes 56
6.2.1 Caractérisation du comportement d'un routeur pour un protocole de test 56
6.2.2 Algorithme d'affinage 58
6.3 Techniques de mise en évidence d'équivalence de chemins 60
6.3.1 Les routeurs "anonymes" 61
6.3.2 Les tunnels 61
6.4 Étude expérimentale 62
6.4.1 Méthodologie 62
6.4.2 Résultats 63

Ce chapitre détaille IBTrack, un logiciel développé pendant cette thèse afin de caractériser le comportement des routeurs d'un chemin par rapport à ICMP. Nous allons présenter les modèles et algorithmes utilisés par IBTrack, puis les mesures faites à grande échelle sur Internet pour donner une vue d'ensemble du comportement des routeurs.

IBTrack a pour but d'assister un utilisateur en cas de problème de connectivité. Par exemple, afin d'obtenir le débit le plus élevé, un utilisateur va fortement se reposer sur les paquets ICMP PTB [79, 78]. IBTrack veut répondre à la question suivante des utilisateurs : "est-ce que je peux faire confiance aux routeurs sur les routes que j'utilise pour m'informer correctement (via des messages ICMP) de l'état du réseau ?". Son but est donc d'être facilement utilisable localement sur une machine connectée à Internet et d'estimer, pour une destination donnée, l'état des routeurs intermédiaires.

Comparé à "Reverse traceroute" [68] et Hubble [69], des travaux proches, IBTrack se différencie en n'ayant pas besoin de recourir à des nœuds de sondage autre que la machine de l'utilisateur (Reverse traceroute) et en ne sondant pas périodiquement les réseaux étudiés (Hubble).

IBTrack est complémentaire aux outils de découverte de la topologie d'un réseau tel que MERLIN [82, 87] : IBTrack caractérise le comportement des routeurs sur un chemin donné alors que MERLIN étudie le réseau dans sa globalité.

Nous allons donc proposer une modélisation du réseau adaptée à cette problématique, des algorithmes de caractérisation du comportement des routeurs et finalement mener une campagne de test à large échelle sur Internet pour donner une vue globale des routeurs d'Internet.

6.1 Modélisation du réseau

Afin de préciser le fonctionnement d'IBTrack, nous commençons par détailler le modèle formel que nous avons utilisé.

6.1.1 Terminologie d'un chemin

Dans un réseau IPv4 ou IPv6, considérons un chemin entre une source *S* et une destination *D*. Ce chemin est constitué de liens successifs entre les routeurs que traverse un paquet envoyé par *S* à destination de *D*. Du fait de la dynamique de la topologie d'un réseau IP, un chemin n'est pas défini statiquement par sa source et sa destination, mais varie dans le temps. De plus, les politiques de routage ne sont pas uniquement basées sur l'adresse de destination d'un paquet IP mais aussi potentiellement sur le champ protocole de l'entête IP, si bien que le chemin dépend du protocole utilisé. Un chemin est donc défini par le quadruplet : (Source, Destination, Temps – i.e. date –, Protocole) et est composé de l'ensemble ordonné des routeurs que parcourt un paquet de protocole P envoyé à l'instant *T* depuis *S* à destination de *D*.
Pour un chemin donné, considérons un routeur *R* le composant. Trois parties du réseau (et pas uniquement du chemin) sont définies (Figure 6.1) :

- le *chemin aller initial* est la sous–partie entre *S* et *R* parcourue par les paquets de protocole P.
- Le *chemin aller final* est la sous–partie entre *R* et *D* parcourue par les paquets de protocole P.
- Le *chemin ICMP retour* est le chemin entre *R* et *S* utilisé par les paquets de protocole ICMP que *R* (en tant que source) envoie à *S* (en tant que destination).

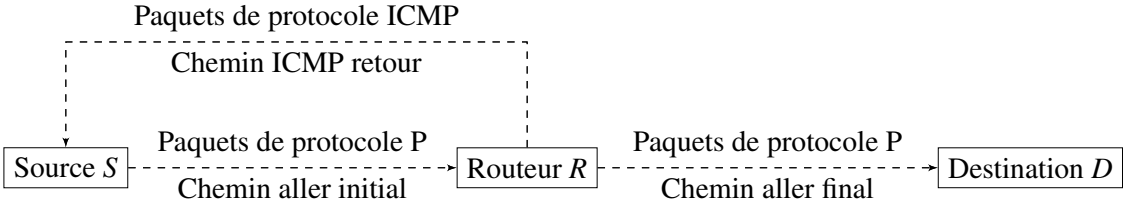


FIGURE 6.1 – Terminologie d'un chemin réseau

Concernant le chemin ICMP retour, il faut noter deux points importants (Figure 6.2) :

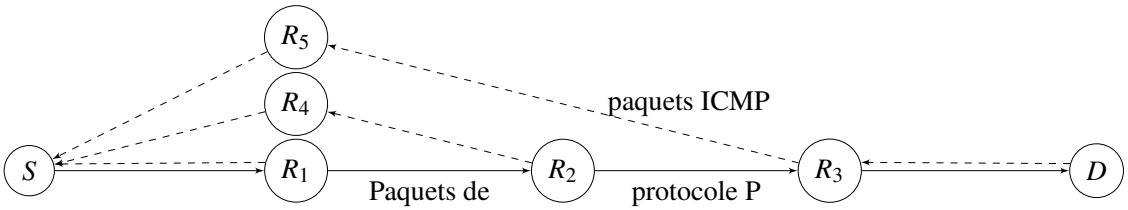


FIGURE 6.2 – Chemins ICMP de retour

– il n'est pas nécessairement identique à l'inverse du chemin aller initial à cause des politiques de routage qui peuvent dépendre du protocole et du fait que le routage n'est pas symétrique en général ;

– les chemins ICMP retour associés à deux routeurs (éventuellement adjacents sur le chemin) peuvent être complètement disjoints.

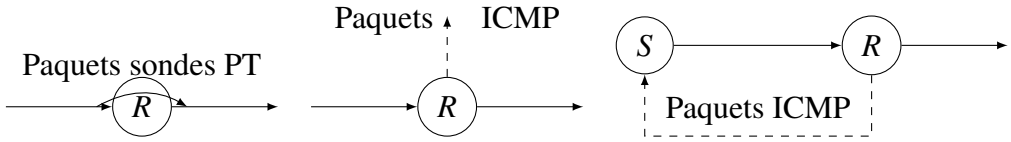
6.1.2 Terminologie des propriétés d'un routeur

Considérons un routeur *R* sur le chemin associé à un protocole de test PT aller. Nous définissons trois propriétés afin de caractériser le comportement du routeur :

Propriété P1 : *R* transmet les paquets de protocole PT.
Tout paquet reçu par *R* depuis le chemin aller initial est émis sur le chemin aller final (nous supposons que le routage est possible).

Propriété P2 : *R* est "coopératif" pour les paquets de protocole PT.
Pour tout paquet reçu par *R* depuis le chemin aller initial dont le traitement doit engendrer l'envoi d'un paquet ICMP sur le chemin ICMP retour, *R* génère et envoie correctement le paquet ICMP.

Propriété P3 : les paquets ICMP ne sont pas filtrés sur le chemin ICMP retour.
Les paquets ICMP émis par *R* à destination de *S* sont bien reçus par *S*. Cela implique que chaque routeur sur le chemin ICMP retour transmet ces paquets ICMP. Cette propriété dépasse le seul routeur *R* ; mais étant limité uniquement à *S*, le chemin ICMP retour est un des moyens d'étudier le routeur *R* depuis *S*.



(a) P1 : *R* transmet les paquets sondes

(b) P2 : *R* crée les paquets ICMP

(c) P3 : *R* crée les paquets ICMP

FIGURE 6.3 – Propriétés pour la caractérisation d'un routeur

Le comportement d'un routeur *R* sur un chemin peut être exprimé comme une expression logique fonction de ces trois propriétés.

Par exemple si le comportement d'un routeur est défini pour un protocole P par `P1.(P2.!P3 +!P2)`, cela signifie qu'il transmet les paquets de protocole PT et soit (i) il

est coopératif mais les paquets ICMP sont filtrés sur le chemin ICMP retour, soit (ii) il n'est pas coopératif et on ne peut rien dire sur le chemin ICMP retour.

L'objectif d'IBTrack est de préciser autant que possible les propriétés `P1`, `P2` et `P3` pour chacun des routeurs du chemin entre S et D pour un protocole donné PT au moment du sondage. Nous allons maintenant détailler les algorithmes utilisés par IBTrack.

6.2 Algorithmes

Dans un premier temps nous présentons l'algorithme qu'utilise IBTrack afin de déterminer le comportement des routeurs sur un chemin lors de l'utilisation d'un protocole donné. Puis, nous étudierons comment deux tests avec des protocoles différents peuvent être recoupés afin de préciser le cas échéant ces propriétés.

6.2.1 Caractérisation du comportement d'un routeur pour un protocole de test

Le premier algorithme utilisé par IBTrack permet de déterminer les trois propriétés associées à un routeur R donné pour un protocole de test PT donné. Pour cela l'algorithme va chercher à déterminer quels paquets parcourent les différents éléments du chemin considéré. Plus particulièrement l'algorithme se compose de deux phases :

- test du chemin aller à travers R,
- test du chemin ICMP retour depuis R.

6.2.1.1 Chemin aller à travers R

Cette phase doit servir à vérifier le comportement de R par rapport à la propriété `P1`. Pour cela il faut regarder si les paquets de protocole PT reçus par R depuis le chemin aller initial sont bien émis sur le chemin aller final (sauf en cas de paquet incorrect bien évidement). N'ayant pas accès au réseau en dehors de la source, il faut se baser sur :

- une preuve de bonne réception d'un paquet du chemin aller initial par un routeur du chemin aller final ; dans ce cas IBTrack se base sur la réception d'un paquet ICMP, déclenché par un paquet sonde, émis depuis un routeur du chemin aller final.
- Une preuve de bonne réception par la destination D ; IBTrack se base alors sur la réception d'un paquet émis par la destination et qui confirme la réception du paquet sonde (typiquement un paquet TCP d'acquittement lors de l'ouverture d'une connexion TCP).

6.2.1.2 Chemin ICMP retour depuis R

Cette deuxième phase sert à évaluer les propriétés `P2` et `P3`. Pour tester la propriété `P2`, il faudrait pouvoir monitorer la sortie de *R* sur le chemin ICMP retour. N'ayant pas d'accès au réseau, IBTrack s'en remet donc à la réception d'un paquet ICMP, ce qui valide aussi la propriété `P3`. Toutefois en cas de non–réception par S du paquet ICMP, IBTrack ne peut pas déterminer quelle propriété n'est pas vérifiée, sachant que cela peut-être les deux.

6.2.1.3 Description de la catégorisation des comportements des routeurs

Cette fonction d'IBTrack est obtenue en suivant l'arbre des possibilités de la Figure 6.4, illustrant l'application des deux étapes.

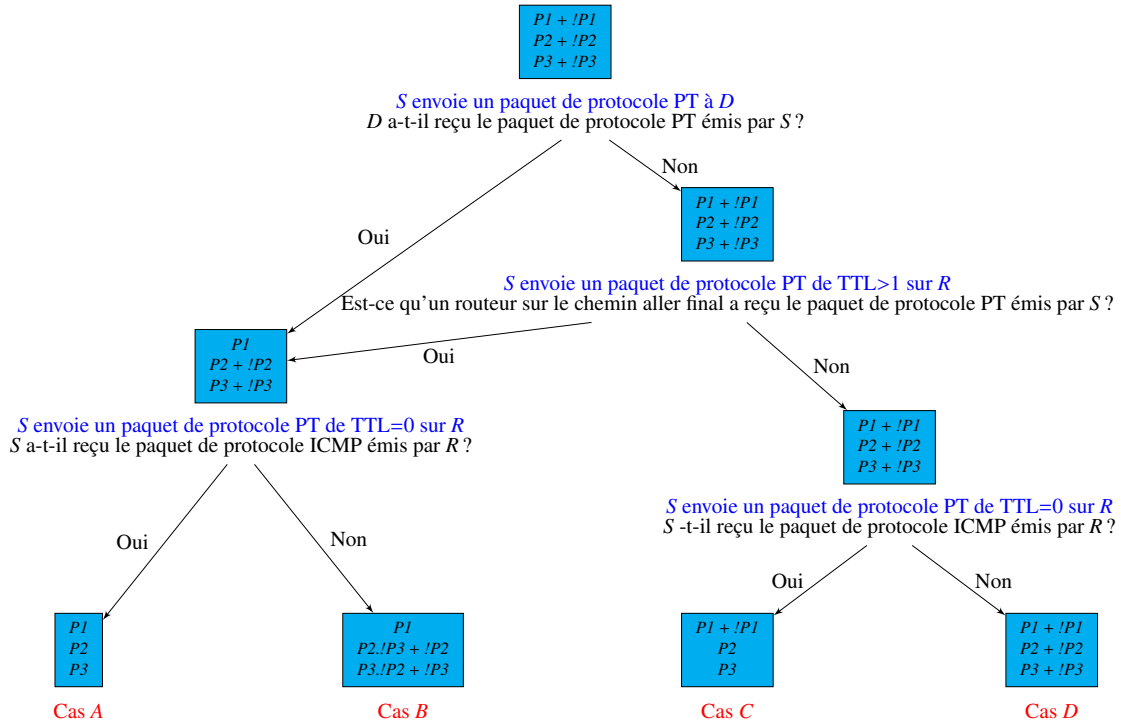


FIGURE 6.4 – Algorithme de caractérisation du comportement d'un routeur R pour un protocole de test PT donné et à un instant donné

Cas A : `P1.P2.P3`
Cela représente le cas d'un routeur qui se comporte "idéalement" pour un utilisateur bienveillant, c'est-à-dire qu'il transmet les paquets de protocole PT, émet les paquets ICMP si besoin et ceux-ci ne sont pas filtrés sur le chemin ICMP retour.

Nombre de sauts	IP du routeur	RTT	Classification IBTrack
1	11.11.11.11	0.3 ms	**A**
2	22.22.22.22	0.4 ms	**A**
3	33.33.33.33	1 ms	**A**

TABLE 6.1 – Exemple de routeurs classifiés **A**

Cas B : `P1.(P2.!P3+!P2).(P3.!P2+!P3)`, plus simplement `P1.!(P2.P3)`
La propriété `P1` est vérifiée, mais ce qui est le plus intéressant pour ce cas est en rapport avec les deux autres propriétés. En effet, IBTrack n'est pas en mesure de détecter sur *S* la réception de paquet *ICMP* émis par *R*. Il y a deux causes possibles : soit *R* n'émet pas de paquets ICMP, soit les paquets ICMP sont filtrés sur le chemin ICMP retour. On peut donc en déduire les implications suivantes :

– `P2` ⇒ `!P3` : si un paquet *ICMP* est émis par *R*, alors il est filtré sur le chemin *ICMP* retour.

– `P3` ⇒ `!P2` : si le chemin *ICMP* retour ne filtre pas les paquets *ICMP*, cela implique que *R* n'a pas émis de paquet *ICMP*.

Nombre de saut	IP du routeur	RTT	Classification IBTrack
3	33.33.33.33	1 ms	A
4	* * *		**B**
5	55.55.55.55	14 ms	

TABLE 6.2 – Exemple de routeur classifié **B**

Cas C : `(P1+!P1).P2.P3`
IBTrack a pu valider l'émission de paquet *ICMP* par *R* et leur routage correct sur le chemin *ICMP* retour ; toutefois aucun paquet, en rapport avec le trafic de test, n'a été reçu depuis la destination ou un routeur du chemin aller final : IBTrack ne peut donc pas définir l'état de *R* pour la propriété `P1`.

Nombre de saut	IP du routeur	RTT	Classification IBTrack
8	88.88.88.88	40 ms	**C**
9	* * *		
etc.	* * *		

TABLE 6.3 – Exemple de routeur classifié **C**

Cas D : `(P1+!P1).(P2+!P2).(P3+!P3)`
Aucun paquet de test émis par IBTrack ne provoque de réaction mesurable à la source. Les propriétés de *R* ne sont donc pas définies dans ce cas.

Nombre de saut	IP du routeur	RTT	Classification IBTrack
8	88.88.88.88	40 ms	C
9	* * *		**D**
etc.	* * *		**D**

TABLE 6.4 – Exemple de routeurs classifiés **D**

6.2.2 Algorithme d'affinage

Afin d'affiner la classification issue de l'algorithme précédent, IBTrack utilise un second algorithme qui est dépendant du protocole de test PT. Pour cela, il faut noter que la propriété `P3`, qui ne concerne que le chemin ICMP retour, est indépendante du protocole de test PT. Si deux chemins (Source *S*, Destination *D*, Temps T, Protocole de test PT1) et (Source *S*, Destination *D*, Temps T, Protocole de test PT2) sont identiques (mêmes routeurs dans le même ordre), alors pour un routeur *R* donné la propriété `P3` doit être identique pour ces

deux tests. Ceci est l'idée principale de l'affinage utilisé par IBTrack. Cependant on est dépendant de l'identité des chemins, question qui sera adressée dans la section 6.3. Pour le moment nous considérons qu'IBTrack a vérifié que les chemins sont identiques.

L'algorithme de catégorisation de base d'IBTrack est capable de déterminer parfaitement l'état de la propriété `P3` pour les cas A et C. IBTrack utilisera en complément l'algorithme d'affinage sur les cas B et D. Regardons chacun de ces deux cas.

6.2.2.1 Affinage du cas B (Figure 6.5)

C'est pour ce cas que l'affinage prend le plus de sens. En effet si le second protocole de test PT2 permet d'affirmer que le routeur *R* considéré respecte la propriété `P3`, alors l'implication `P3` $\Rightarrow$ `!P2` est applicable pour le protocole PT1. Donc si la catégorisation pour le protocole PT2 classe *R* dans le cas A ou C, IBTrack pourra conclure que *R* ne respecte pas la propriété `P2` pour le protocole PT1, comme présenté dans la Figure 6.6. Cela crée donc un sous–cas de B, que l'on nomme B', et qui peut être exprimé comme suit : `P1.!P2.P3`

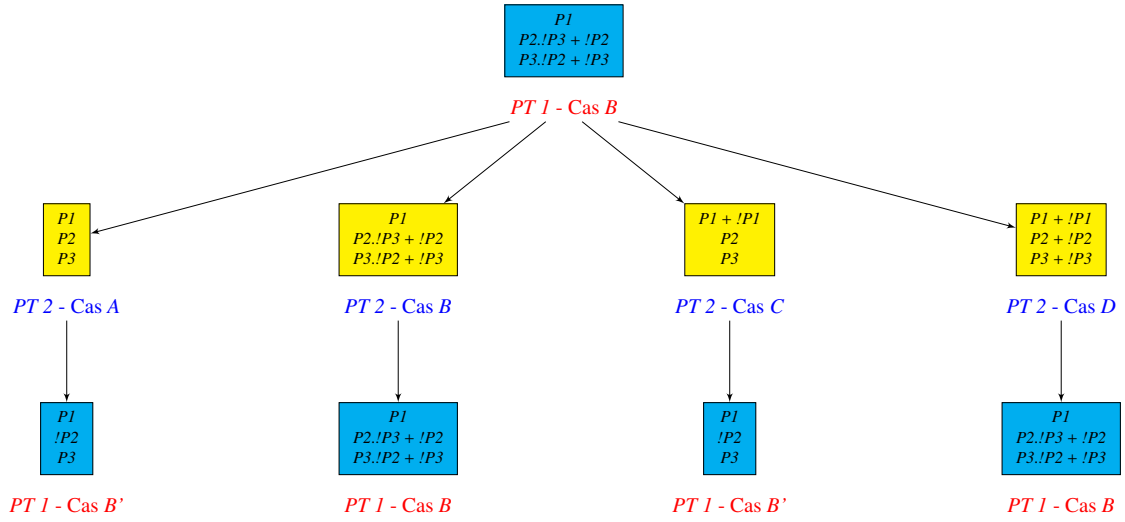


FIGURE 6.5 – Affinage d'un cas *B* avec un second protocole de test

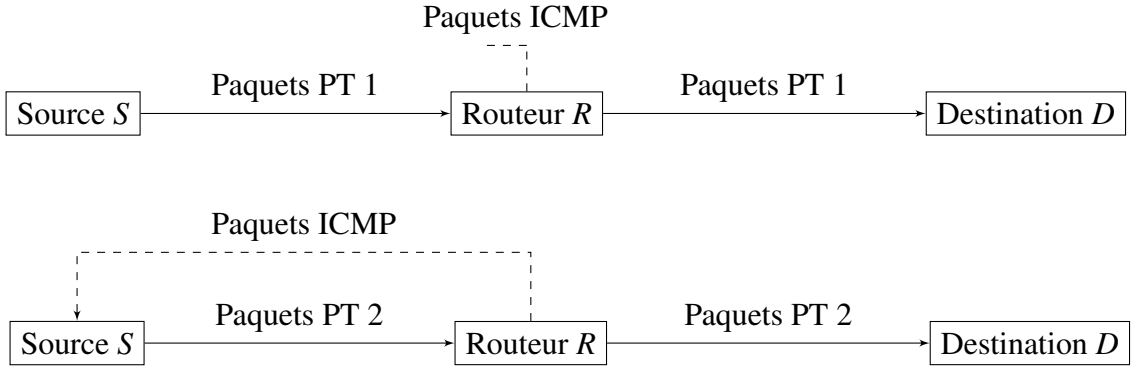


FIGURE 6.6 – Cas de deux tests permettant l'affinage de la propriété P3

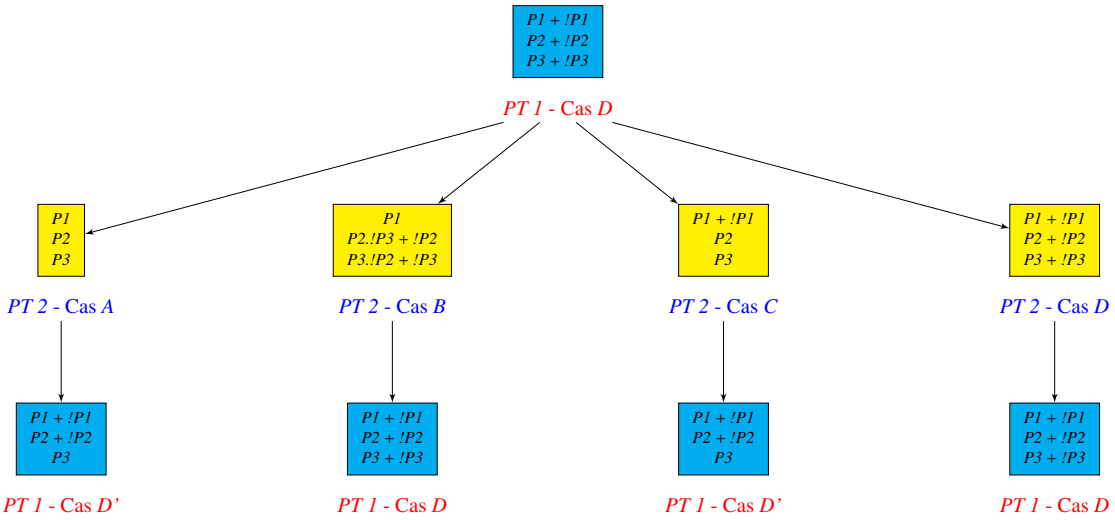


FIGURE 6.7 – Affinage d'un cas *D* avec un second protocole de test

6.2.2.2 Affinage du cas D (Figure 6.7)

L'affinage de ce cas n'est possible que pour la propriété `P3` car il n'y a plus d'implication comme précédemment. IBTrack peut affiner le cas D pour un protocole PT1 uniquement lorsque, pour un autre protocole PT2, la catégorisation donne le cas A ou C. Le cas D' exprime le fait que malgré le fait que le chemin ICMP retour ne filtre pas les paquets ICMP, on ne dispose d'aucune information sur le traitement des paquets PT1 ou sur la création de paquet ICMP déclenchée par les paquets PT1 ; une des explications possibles est le fait que le routeur R n'a pas reçu de paquet PT1, ce qui empêche de faire des déductions fortes de ces observations.

6.2.2.3 Extension de l'affinage avec plus de deux protocoles.

Il est évidement possible d'étendre cet affinage avec plusieurs protocoles, mais il faut toujours s'assurer que l'égalité des chemins pour chaque comparaison deux à deux soit valide.

Les algorithmes qu'utilise IBTrack étant définis, nous allons maintenant présenter la méthodologie utilisée pour évaluer si deux chemins (ou plus) sont identiques.

6.3 Techniques de mise en évidence d'équivalence de chemins

L'application de l'algorithme d'affinage en utilisant plusieurs protocoles de test requièrent de considérer des chemins équivalents ; c'est la difficulté principale pour IBTrack. En effet, il faut être capable de comparer des chemins dont certains routeurs sont "anonymes", c'est-à-dire qui n'apparaissent pas explicitement en sortie de `traceroute` car représentés par "*" (si il n'y a pas de routeurs "anonymes" alors la comparaison des chemins est triviale).

De plus, il faut aussi être capable de gérer les tunnels qui sont de plus en plus présents au sein de la topologie d'Internet (par exemple les tunnels MPLS[46]). Enfin, il faut noter que nous stoppons les tests une fois cinq routeurs "anonymes" consécutifs rencontrés tel que présenté Figure 6.8a.

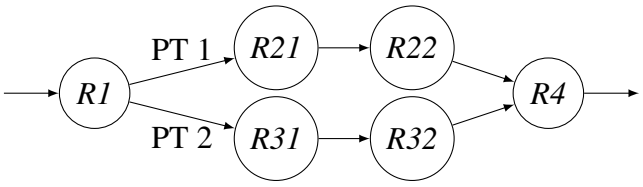
Protocole de test
12 *R12*
13 *
14 *
15 *
16 *
17 *

(a) Fin de test par cinq routeurs "anonymes".

PT 1	PT 2
1 *R1*	1 *R1*
2 *	2 *R2*
3 *	3 *R3*
4 *	4 *R4*
5 *	5 *R5*
6 *R6*	6 *R6*

(b) Deux chemins considérés identiques.

PT 1	PT 2
1 *R1*	1 *R1*
2 *R21*	2 *R22*
3 *R31*	3 *R32*
4 *R4*	4 *R4*



(c) Deux chemins considérés identiques lorsque deux routeurs différents sont acceptés.

FIGURE 6.8 – Exemples de comparaison de chemins

6.3.1 Les routeurs "anonymes"

N'utilisant pas d'autres machines sur le réseau que la source des tests, IBTrack n'as pas de moyen d'expliciter les routeurs "anonymes". À la place, il utilise un algorithme dont le but est de vérifier si deux (ou plusieurs) chemins peuvent être identiques.

Pour ce faire, IBTrack vérifie dans un premier temps que les routeurs identifiés sur tous les chemins considérés (deux ou plus) se situent à la même place par rapport à la source *S*. Puis, il vérifie qu'entre les routeurs identifiés il y a le même nombre de routeurs "intermédiaires" sur chaque chemin. Deux exemples sont présentés dans la Figure 6.8b.

Ainsi deux chemins sont considérés équivalents si et seulement si il est possible de substituer chacun des routeurs anonymes qu'ils comportent par des routeurs intermédiaires pour rendre les deux chemins identiques.

6.3.2 Les tunnels

Les tunnels étant de plus en plus présents sur Internet (30% des chemins contiennent un tunnel MPLS d'après l'étude de Donnet et al. [46]), IBTrack est aussi équipé d'un algorithme permettant de considérer égaux des chemins localement différents à cause de la présence de tunnels.

Pour ce faire, IBTrack permet de configurer le "nombre maximal de routeurs consécutifs différents" qu'il accepte, afin de pouvoir effacer les différences de chemin dues par exemple

à un routage en fonction du protocole IP, comme illustré par la Figure 6.8c. De plus, il faut noter que cette fonctionnalité est cumulable avec la précédente qui concerne les routeurs "anonymes". Dans l'ordre, IBTrack va donc ignorer les routeurs "anonymes" puis appliquer l'algorithme de gestion des tunnels (s'il est actif).

Les mécanismes d'IBTrack présentés, nous exposons maintenant la méthodologie de mesure utilisée pour la campagne de tests.

6.4 Étude expérimentale

Cette section présente dans un premier temps la méthodologie utilisée par IBTrack pour mettre en œuvre les algorithmes définis à la section 6.2, puis les résultats de la campagne de test sur Internet.

6.4.1 Méthodologie

Nous présentons ici la politique de choix des adresses IP utilisées soit comme source soit comme destination, l'ensemble des outils sur lesquels IBTrack se base et les méthodes de détermination de l'équivalence de plusieurs chemins.

6.4.1.1 Adresses IP source

Pour tester à grande échelle IBTrack sur le réseau Internet, nous avons utilisé les machines du réseau Planet-Lab[25] comme source pour nos mesures. Planet-Lab est un réseau de machines connectées à Internet dont le but est l'aide au développement de nouveaux services réseau. Planet-Lab contient plus de 1000 nœuds répartis sur plus de 500 sites à travers le monde, majoritairement hébergés par des instituts académiques et des laboratoires industriels.

Pour notre campagne de test, nous avons déployé IBTrack sur de nombreux serveurs distribués dans le monde. Néanmoins seules trente machines étaient utilisables pour nos mesures, les autres ne fonctionnant pas ou ne permettant pas l'utilisation des outils nécessaires au bon fonctionnement d'IBTrack. Cela nous permet d'avoir une bonne couverture des chemins possibles pour atteindre une destination précise, et donc d'obtenir une bonne vue d'ensemble du réseau Internet.

6.4.1.2 Adresses IP destination

Afin de tester un sous-ensemble représentatif du réseau Internet, nous avons utilisé une liste des sous-réseaux `IPv4 /24` routables d'Internet. Nous nous sommes servis d'un instantané des sous-réseaux `IPv4 /24` routables fourni par CAIDA[62], composé d'environ 10 000 sous-réseaux. Pour les mesures, il faut noter deux choses à propos de l'instantané de CAIDA : (i) il est en fait constitué d'adresse IPv4+ dont l'adresse machine est aléatoire pour chaque sous-réseau (il n'y a donc pas nécessairement d'hôte connecté et/ou fonctionnant), et (ii) il est fortement dépendant du moment de sa création vu que la topologie d'un réseau IP est dynamique (à la fois au niveau du routage, mais aussi de ses sous-réseaux).

En conséquence, les adresses IP destinations que nous avons utilisées sont au nombre de 10 000, et seul un petit sous ensemble correspond à une machine physique susceptible de répondre.

6.4.1.3 Outils utilisés par IBTrack

Pour la partie récupération des mesures réseau, IBTrack se base sur `scamper`[80] qui implante en particulier les fonctionnalités de `paris-traceroute`[33].

Paris-traceroute Cette version améliorée de l'outil `traceroute` classique permet de révéler plus de liens et routeurs, tout en supprimant des faux liens déduits par `traceroute`.

Scamper Cet outil implante la plupart des outils de diagnostic réseau (tel que `ping`, `paris-traceroute`, etc) avec la possibilité de paralléliser leurs exécutions.

Une des fonctionnalités que nous utiliserons est la capacité pour `scamper` d'arrêter les mesures après un certain nombre d'échecs dans le mode de fonctionnement `traceroute`. Plus particulièrement, nous avons réglé le mode `paris-traceroute` de `scamper` pour qu'il s'arrête après cinq essais qui échouent (c'est-à-dire, pour cinq valeurs consécutives du champ TTL, les trois tests échouent). Toutefois, lors de l'interprétation des tests nous ne considérerons qu'un seul des cinq essais afin de ne pas trop bruiter les résultats. En effet, prendre en compte ces cinq routeurs représenterait une modification du nombre de routeurs sur un chemin de l'ordre de 10 % à 30 % ; or un seul routeur qui filtrant le trafic peut créer un blocage pour toute la suite de la route. Lors de l'interprétation des résultats il faudra donc garder cette "simplification".

La section suivante présente les résultats des tests à large échelle sur Internet ainsi qu'une analyse fine sur les résultats des différents algorithme d'IBTrack.

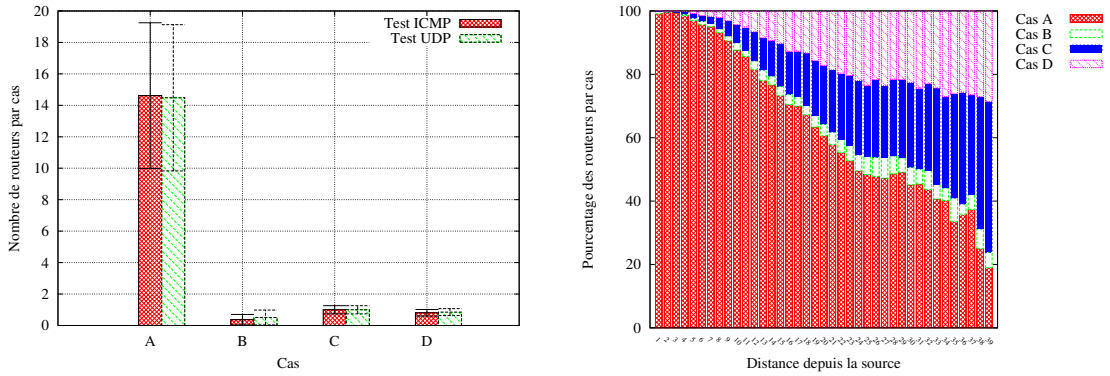
6.4.2 Résultats

Cette section présente en détail les résultats des mesures faites sur Internet depuis trente nœuds Planet-Lab ainsi qu'un accès via un fournisseur d'accès Internet (FAI) de type "particulier". Comme précisé à la section 6.4.1.2, plus de 10 000 adresses IPv4 sont utilisées comme destination de cette campagne de test.

6.4.2.1 Étude à large échelle d'Internet

Dans un premier temps, nous allons utiliser les résultats obtenus par IBTrack pour définir les comportements (i) des chemins et (ii) des routeurs qu'un utilisateur peut obtenir sur Internet. Sondant depuis les nœuds Planet-Lab, nous ne pouvons pas utiliser le protocole TCP, qui sera utilisé ultérieurement pour l'étude fine.

Caractérisation des chemins. Nous commençons par caractériser les chemins sur Internet avec `traceroute` en normalisant le nombre de routeurs par cas classifié précédemment comme le montre la Figure 6.9a. En résumé, un chemin moyen sur Internet est composé de 16,78 routeurs qui sont majoritairement (14,5 sur les 16,78) dans le cas "idéal" A pour les utilisateurs.



(a) Nombre de routeurs par cas (normalisé par test, avec l'écart type)

(b) Répartition des routeurs par cas en fonction de la distance à la source S

FIGURE 6.9

Puis nous caractérisons la répartition normalisée des routeurs par cas en fonction de leur distance avec la source du test (Figure 6.9b). Cela met en évidence le fait que plus un routeur est "loin" (en terme de nombre de routeurs traversés) moins "bien" il se comporte vis-à-vis du protocole ICMP. Pour mémoire, lorsque le test de type `traceroute` se termine prématurément (suite à cinq routeurs consécutifs ne répondant pas, identifiés par "*"), seul un des cinq routeurs sera comptabilisé dans les résultats, comme expliqué à la section 6.4.1.3.

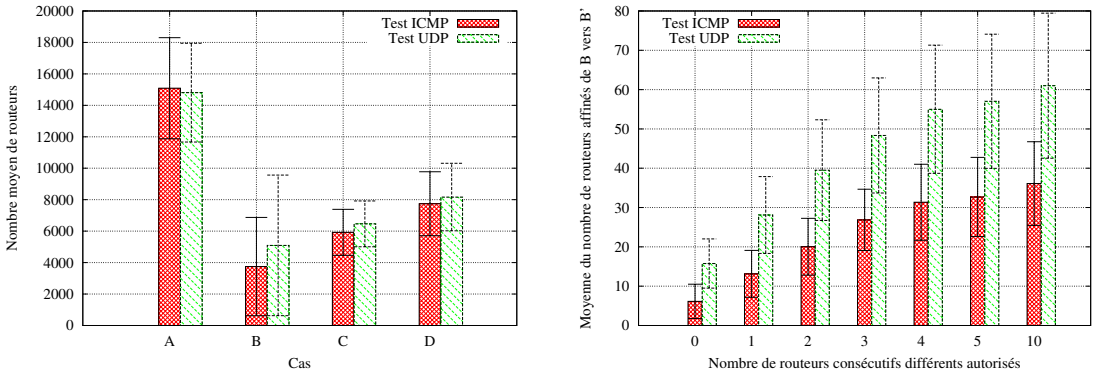
Le corollaire des Figures 6.9a et 6.9b est donc que la majorité des chemins sur Internet sont courts mais qu'avec la longueur ils deviennent composés de routeurs traitant moins bien ICMP.

Nous allons maintenant étudier le comportement des routeurs rencontrés afin de définir un profil de routeur "moyen" sur Internet tel que vu par un utilisateur.

Caractérisation des routeurs Dans un premier temps, nous cherchons à caractériser l'état moyen des routeurs d'Internet pour un utilisateur. Pour ce faire nous avons, pour chaque nœud Planet-Lab, regroupé l'ensemble des routeurs atteints (en retirant les occurrences multiples - grâce à l'algorithme d'identification des routeurs "anonymes" en particulier) ; puis les résultats pour chaque source sont agrégés sur la Figure 6.10a qui les présente de façon normalisé pour une source (avec l'écart type). Attention au cas D dont on ne représente qu'un essai sur cinq (cf. section 6.4.1.3). Contrairement à la Figure 6.9a qui montrait qu'un chemin était principalement (14,5/16,78) constitué de routeurs A, ils ne sont pas majoritaires à l'échelle d'Internet même si A reste le cas le plus répandu.

Ainsi nous pouvons conclure que la majeure partie des chemins sur Internet utilise essentiellement des routeurs de type A, mais ces routeurs A ne représentent qu'un peu moins de 50% des routeurs atteints, et ce que le protocole de test soit ICMP ou UDP.

Pour finir l'étude à large échelle sur Internet, nous allons regarder un peu plus précisément l'algorithme d'affinage sur les routeurs du cas B ainsi que les aspects tunnels via notre algorithme de comparaison des chemins. Nous allons donc chercher à évaluer pour chaque nœud Planet-Lab combien de routeurs du cas B peuvent être affinés en B' grâce au second



(a) Nombre de routeurs différents par cas (normalisé par source, avec l'écart type)

(b) Nombre de routeurs affinés du cas B à B' en fonction de la longueur des "tunnels" autorisée (avec l'écart type).

FIGURE 6.10

protocole de test en vérifiant que le même chemin soit utilisé. De plus, nous allons faire varier le nombre maximal de routeurs consécutifs différents autorisés (aussi dénommée la longueur autorisée des "tunnels") de l'algorithme d'identification des chemins identiques.

Les résultats sont présentés sous forme normalisée pour l'ensemble des nœuds Planet-Lab sur la Figure 6.10b. L'application de l'algorithme d'affinage sans autoriser de divergence de chemin permet de préciser en moyenne le comportement de 0,16% (ICMP) et 0,31% (UDP) des routeurs catégorisés B. Il est plus facile d'affiner dans le cas d'UDP car plus de routeurs sont initialement dans le cas B (30% plus nombreux que pour ICMP). Une explication possible est l'interdiction explicite de créer des paquets ICMP dans le cas de trafic UDP, basée sur le fait qu'UDP ne garantie pas le transport des données.

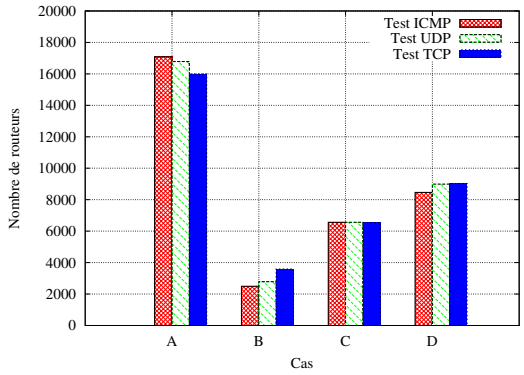
Si on augmente la longueurs des tunnels autorisés à 5 (respectivement 10), le nombre moyen de routeur affiné passe de 6 à 32 (resp. 36). Il est donc important d'autoriser une longueur moyenne d'au moins cinq routeurs différents. Cela corrèle le fait que 90% des tunnels MPLS (présents sur au moins 30% des chemins sur Internet) sont au plus d'une longueur de cinq sauts [46].

La section suivante se concentre sur l'étude d'un cas spécifique, en particulier sur l'application de l'algorithme d'affinage.

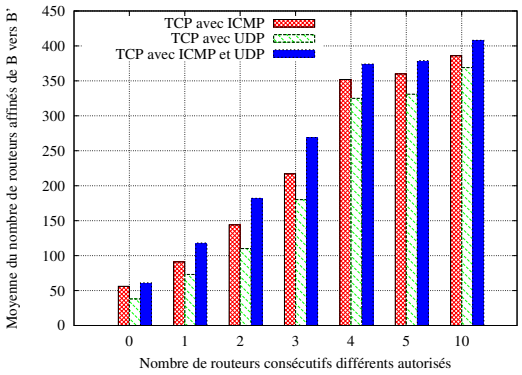
6.4.2.2 IBTrack en action chez un utilisateur d'Internet

Nous avons effectué la même campagne de test depuis un accès Internet via un FAI pour particulier. Cela nous a permis de faire le test en utilisant le protocole TCP. La répartition des routeurs est présentée sur la Figure 6.11a. Les chiffres sont globalement similaires aux mesures depuis Planet-Lab (cf. Figure 6.10a), avec toutefois des routeurs plus coopératifs (plus de cas A et moins de cas B).

La Figure 6.11b détaille l'affinage des routeurs du cas B lors du test TCP. Pour l'affinage on a utilisé ICMP et UDP seuls, puis les deux protocoles ensemble. Il s'agit de déterminer si, pour l'un des deux protocoles utilisés pour l'affinage, on est capable de classer le routeur dans le cas B' pour TCP. La principale conclusion de cette expérience est le fait que l'utilisation de plusieurs protocoles lors de l'affinage ne modifie que faiblement le résultat



(a) Répartition des routeurs depuis un accès type "particulier"



(b) Affinage du cas *B* pour le protocole TCP.

FIGURE 6.11

par rapport à l'utilisation d'un seul protocole. Si l'on compare l'affinage en utilisant ICMP et ICMP+UDP, l'amélioration n'est que de 5,2% ; elle monte jusqu'à 14,5% lorsque l'on compare l'affinage avec UDP et avec `ICMP+UDP`. Nous interprétons ceci au niveau des routeurs par la présence d'une règle explicite qui interdit la création des paquets ICMP "Time Exceeded" lors du traitement de trafic TCP, probablement pour limiter les risques d'attaque.

Conclusion

Nous avons donc présenté IBTrack, un utilitaire développé pour les utilisateurs finaux d'Internet, afin d'évaluer le comportement des routeurs sur un chemin par rapport à ICMP.

Dans le cas de connexion TCP chez un particulier, il permet une caractérisation plus poussée de 10% des routeurs ne répondant pas en ICMP mais qui traitent correctement le trafic TCP pour la partie routage, ce qui représente 1% du nombre total de routeurs atteints par l'utilisateur.

Le but d'IBTrack est de permettre l'utilisation optimale des réseaux en prenant en compte au mieux le traitement d'ICMP par les routeurs.

Les prochains chapitres vont désormais étudier l'utilisation d'ICMP comme vecteur d'attaque contre les passerelles de sécurité.

Chapitre 7

Attaque d'une implantation IPsec intégrée par exploitation des messages ICMP

Sommaire

7.1 Gestion d'ICMP par IPsec 68
7.2 ICMP comme vecteur d'attaque d'IPsec 69
7.2.1 Modèle de l'attaquant 69
7.2.2 Les attaques potentielles via ICMP 70
7.2.3 Contre-mesures 71
7.3 Déni de service sur les connexions TCP d'un utilisateur 72
7.3.1 Description de l'attaque 73
7.3.2 Attaque d'une passerelle IPsec générique 74
7.3.3 Comportement logiciel de la machine utilisateur 75
7.4 Effet de l'attaque sur un flux UDP 78
7.5 Surcoût des tunnels vs taille minimale des paquets IP 80
7.6 Recherche de PMTU dans IPsec 80

Dans ce chapitre nous présentons l'impact du protocole ICMP sur les passerelles VPN IPsec intégrées. Nous proposons une analyse de sécurité dans un premier temps, puis mettons en place une attaque à base de paquets ICMP PTB qui peut mener au déni de service. Nous montrons en particulier que la configuration par défaut de Linux est très facilement attaquable, et qu'un problème fondamental existe quant au support des tailles minimum de paquets en présence de tunnels. Finalement, nous proposons une solution pour contourner le problème lorsqu'une passerelle en rupture est utilisée.

Nous considérons pour ce chapitre un modèle de sécurité soit par des passerelles VPN soit par une implantation locale d'IPsec (utile dans le cas d'appareils nomades, cf. Figure 7.1). De plus, nous nous limitons aux implantations d'IPsec en mode tunnel avec le protocole ESP (section 2.2) car ce sont celles qui peuvent garantir le plus de sécurité.

Puisque les attaques que nous considérons sont basées sur ICMP, nous allons tout d'abord préciser la gestion des paquets ICMP par la norme IPsec.

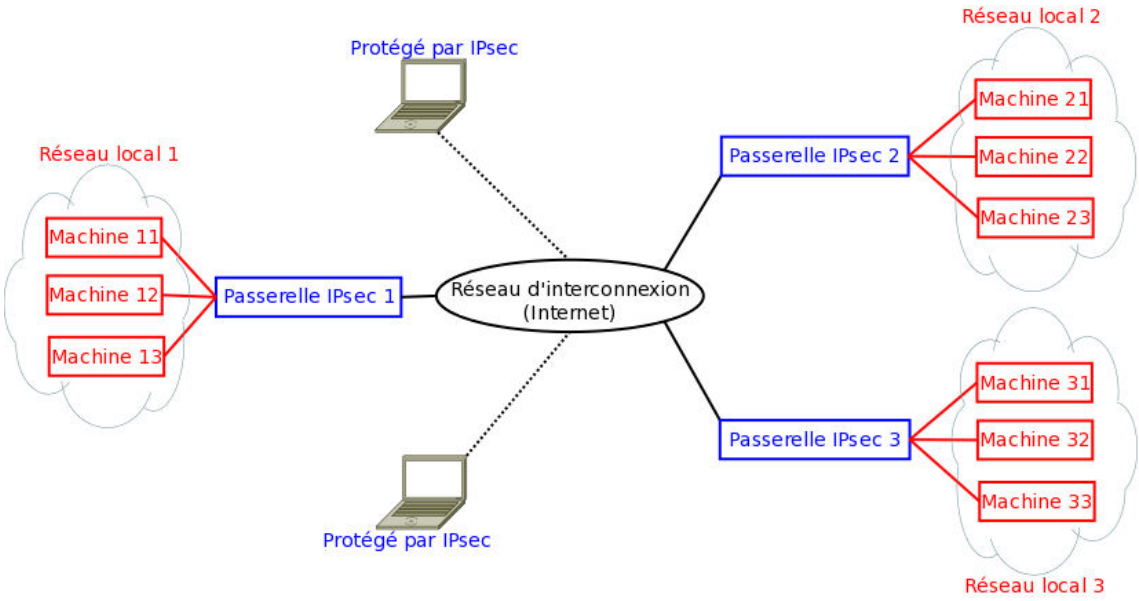


FIGURE 7.1 – Modèle réseau d'un VPN réalisé avec des passerelles et appareils nomades.

7.1 Gestion d'ICMP par IPsec

Premièrement nous rappelons que les types ICMP sont séparés en deux catégories : les messages d'erreur et ceux d'information (section 5.1.1). Voici donc les recommandations de traitement des paquets ICMP par la norme IPsec [73] :

- tous les types ICMP d'information (quelle que soit leur provenance, réseaux rouge ou noir) doivent faire l'objet d'une règle explicite de traitement dans la SPD ;

- pour les messages ICMP d'erreur en provenance d'un réseau rouge, la passerelle IPsec doit d'abord vérifier que le paquet est légitime, c'est-à-dire que sa source et sa destination correspondent à un tunnel ouvert dans la SAD. Plus précisément, cela veut dire qu'il existe une clé de session dans la SAD dans le sens destination vers source ; si ce n'était pas le cas, le message ICMP ne serait pas déclenché par un paquet préalable et n'aurait donc pas de raison d'exister ;

- les messages ICMP d'erreur en provenance du réseau d'interconnexion doivent faire l'objet d'une règle de traitement qui est de la responsabilité de l'administrateur de la passerelle IPsec. L'implantation pourra en particulier vérifier les données du paquet ICMP. En effet, pour certains types de messages d'erreurs, le paquet ICMP doit contenir des données du paquet ayant déclenché le message ICMP. Dans le cas d'un paquet "déclencheur" appartenant à un tunnel IPsec, la passerelle sera donc en mesure de déchiffrer, au moins partiellement, les données du paquet "déclencheur" : grâce à l'index du paramètre de sécurité qui est contenu au début du de l'en–tête AH ou ESP la passerelle peut retrouver les informations nécessaires au déchiffrement dans la SAD. Une fois les données contenues dans le paquet ICMP déchiffrées, la passerelle pourra vérifier la légitimité du paquet "déclencheur", par exemple en vérifiant que les adresses déchiffrées source et destination du paquet "déclencheur" font partie d'un tunnel autorisé.

Pour le cas particulier des paquets ICMP PTB, utilisés par PMTUd, si l'administrateur les accepte, alors le PMTU sera stocké directement dans la SAD. Ultérieurement, en cas de réception par la passerelle d'un paquet d'un réseau rouge excédant le PMTU, la passerelle peut soit le fragmenter (dans le cas d'IPv4 avec le bit DF à 0) soit le jeter et émettre un paquet ICMP PTB (dans le cas d'IPv4 avec le bit DF à 1 ou pour IPv6) en précisant la taille maximale, calculée à partir du PMTU annoncé par le réseau d'interconnexion et de l'éventuelle encapsulation due au tunnel.

Ces règles sont résumées dans la Table 7.1.

Type ICMP	**Réseau d'origine**	**Traitement recommandé**
Message d'information	Tous	Politique de l'administrateur
Message d'erreur	Réseau rouge	Vérification de la légitimité, puis traitement
Message d'erreur	Réseau noir	Politique de l'administrateur

TABLE 7.1 – Recommandations de traitement d'ICMP dans IPsec

La prochaine section étudie comment ICMP, malgré une configuration correcte, peut être un vecteur d'attaque pour une passerelle IPsec.

7.2 ICMP comme vecteur d'attaque d'IPsec

Avant de discuter des attaques de passerelle IPsec avec des paquets ICMP bien choisis et des contre-mesures possibles, nous allons d'abord définir le modèle de l'attaquant.

7.2.1 Modèle de l'attaquant

Nous définissons les différents attaquants par rapport aux deux capacités suivantes :

– premièrement, nous considérons qu'un attaquant peut **écouter sur le réseau noir**. Cette hypothèse est raisonnable dans le cas d'appareils nomades utilisant une connexion de type WiFi. En effet, dans ce cas n'importe quel appareil connecté au réseau WiFi peut écouter le trafic.

– Deuxièmement, un attaquant peut **émettre sur le réseau noir**. Cette transmission peut être "sur le chemin", c'est-à-dire qu'elle est émise par un routeur ou sous-réseau sur la route entre la source et la destination du tunnel IPsec attaqué, sinon elle peut être "hors du chemin". De plus, nous considérons que l'attaquant peut **usurper les adresses**, c'est-à-dire utiliser une adresse IP source qui n'est pas la sienne. En reprenant l'exemple d'un utilisateur nomade sur un réseau WiFi, l'injection d'un paquet à destination de l'utilisateur est immédiate.

Toutefois, l'attaquant ne peut pas chiffrer ou déchiffrer un paquet ; les algorithmes cryptographiques de chiffrement ou d'échange de clés sont considérés robustes.

Nous décrivons maintenant les attaques permises par le modèle de l'attaquant.

7.2.2 Les attaques potentielles via ICMP

Considérant la couche cryptographique robuste, nous ne cherchons pas à attaquer la confidentialité des échanges, mais plutôt :

- épuisement des ressources et donc déni de service (DoS) ;
- réduction du débit des communications ;
- blocage des communications.

De plus, contrairement aux attaques présentées au paragraphe 2.2.6, nous concentrons notre travail sur les attaques utilisant les spécificités réseau plutôt que cryptographiques. Nous retirons aussi du champ de l'adversaire les attaques ICMP uniquement réseau telles que celles présentées au paragraphe 5.3,

Nous présentons dans cette section, en fonction des types ICMP, les attaques possibles sur une implantation IPsec.

7.2.2.1 Déni de service (DoS)

Typiquement, les attaques de type DoS se basent sur l'épuisement des ressources de calcul d'une entité responsable d'un traitement particulier. Si IPsec est configuré pour rejeter tout paquet ICMP, le risque est très faible.

Par contre, si l'implantation d'IPsec autorise certains types de messages, issus du réseau noir, alors le risque de DoS n'est pas nul car la vérification de l'appartenance à un tunnel actif des données du paquet ICMP coûtera alors à IPsec des calculs supplémentaires (cryptographiques et donc coûteux). L'attaque DoS en sera alors facilitée.

7.2.2.2 Réduction du débit

Un attaquant a deux mécanismes ICMP pour réduire le débit :

- dans le cas de flux TCP, le type "source quench" a pour but d'informer un client ou un serveur de réduire son débit afin d'éviter la congestion de l'autre participant (RFC 5927 [53]). Un attaquant peut utiliser ce type à condition de pouvoir atteindre le client ou le serveur (et donc de passer à travers la passerelle IPsec) ;
- en émettant un paquet ICMP PTB, l'attaquant peut forcer la passerelle IPsec à réduire la taille autorisée en sortie sur le réseau noir. Pour IPv6 et IPv4 avec le bit DF à 1, les paquets ICMP PTB (ou du moins l'information) doivent être transmis jusqu'à la machine qui a émis le paquet "déclencheur" car c'est la seule qui a le droit de le fragmenter. En cas d'IPv4 avec le bit DF à 0, le routeur précédent (potentiellement implantant IPsec) doit faire lui-même la fragmentation, pouvant en plus mener à un DoS par épuisement des ressources de calcul du routeur.

Pour ces deux attaques, il est nécessaire de pouvoir écouter et émettre sur le réseau noir.

7.2.2.3 Blocage des communications

Pour les attaques visant à empêcher toute communication, deux types de messages ICMP sont directement utilisables : le type *destination injoignable* (sauf pour le code PTB) et le type *redirection*, mais ces deux attaques concernent la partie routage d'IP et sont donc hors de notre champ d'analyse.

Nous allons maintenant présenter nos recommandations pour palier le plus possible à ces attaques.

7.2.3 Contre-mesures

Les contre-mesures à ces attaques proviennent essentiellement de la politique de sécurité que l'administrateur met en place. Nous rappelons que nous ne traitons pas les attaques ICMP directes, telles que les attaques DoS par surcharge (sauf si l'attaque utilise IPsec pour épuiser les ressources) et que les machines des réseaux rouges sont supposées de confiance (et donc ne peuvent être à l'origine d'attaques). Voici nos recommandations de traitement d'ICMP dans une implantation d'IPsec. Nous attirons l'attention du lecteur sur le fait que nos recommandations ne sont destinées qu'aux implantations d'IPsec, mais qu'une autre partie de la pile réseau (routage par exemple) peut devoir prendre en compte un paquet ignoré par IPsec (à condition de traiter le paquet avant IPsec).

7.2.3.1 Messages ICMP d'information

Lorsqu'ils proviennent du réseau noir, IPsec doit les supprimer sans condition (c'est-à-dire sans vérification afin de ne pas risquer une attaque DoS par épuisement à cause des calculs cryptographiques). Cela n'est pas problématique de supprimer ce type de messages car ils sont issus de routeurs intermédiaires sans avoir été déclenché par un paquet d'un tunnel. Par exemple un *ping* d'un routeur du réseau noir doit être supprimé par IPsec, mais ce n'est pas problématique car ICMP peut répondre "avant".

7.2.3.2 Messages ICMP d'erreur

Le réseau rouge étant de confiance, nous allons nous concentrer sur les messages ICMP d'erreur provenant du réseau noir, dont les types peuvent être subdivisés en deux sous–catégories : (i) les erreurs dues à des problèmes de connexion et (ii) les erreurs des réseaux. Avant de présenter les politiques recommandées (par type ICMP), nous rappelons que dans le cadre d'IPsec en mode tunnel il y a deux "niveaux" dans un paquet (Figure 2.3) :

- le paquet IP intérieur est le paquet IP tel qu'émis par la source (et reçu par la destination) sur les réseaux rouges. Les adresses IP de l'en-tête sont celles de la source et de la destination ;
- le paquet IP extérieur est le paquet IP qui transite sur le réseau noir. Son en-tête est composé des adresses IP des passerelles (ou d'un appareil nomade) et ses données contiennent le paquet IPsec (dont le paquet intérieur chiffré).

Les messages d'erreurs qui nous intéressent sont :

Dépassement du temps : lorsque le mode tunnel est utilisé dans IPsec, le lien entre deux implantations d'IPsec compte comme un "saut" du point de vue du lien IP intérieur. Lors de la réception d'un paquet depuis le réseau rouge, une passerelle doit donc d'abord décrémenter le champ TTL avant de procéder aux traitements IPsec. Si un paquet ICMP de ce type arrive depuis le réseau noir, IPsec doit donc le supprimer, après traitement dans IP le cas échéant. Certes IPsec est de niveau réseau, mais il se situe au dessus d'IP. Ce type de messages ICMP doit donc être traité par IP et n'a plus de sens au niveau IPsec.

Source Quench : ce type ICMP est utilisé entre acteurs d'une connexion TCP pour réduire le débit. Ils sont donc légitimes au sein d'une connexion intérieure, mais doivent être supprimés s'ils arrivent du réseau noir.

Problème de paramètre : ces messages ICMP font suite à des paquets dont l'en-tête contient une erreur. S'ils sont émis depuis le réseau noir, ils ne concernent donc que l'en-tête IP extérieur. IPsec peut donc enregistrer l'erreur et doit ensuite supprimer le paquet.

Destination injoignable : ce type regroupe plusieurs sous–types. Nous écartons d'abor le cas particulier d'ICMP PTB qui sera traité au paragraphe suivant. Les autres sous–types indiquent un problème lors de l'acheminement du paquet. Les erreurs ne concernent donc que les informations de l'en-tête extérieur. IPsec peut donc enregistrer l'erreur et doit ensuite supprimer le paquet.

Paquet trop gros (PTB) : ce couple (type, code) ICMP est particulièrement important à cause de son implication dans le PMTUd. Dans le cadre d'IPsec, les informations sur le PMTU ainsi remontées doivent être stockées dans la SAD, potentiellement être contraintes par un minimum (comme prévu par les normes IP par exemple), et accessibles et modifiables par l'administrateur.

Ce mécanisme de stockage dans la SAD permet à un attaquant de contourner la frontière qu'IPsec est censé faire entre les réseaux rouges et noirs. Pour cela, il suffit d'envoyer un paquet ICMP PTB depuis le réseau noir, avec le PMTU désiré : un nouveau paquet ICMP PTB sera alors envoyé par la passerelle IPsec sur le réseau rouge si un paquet intérieur dépasse ce PMTU. Un administrateur peut figer les valeurs de PMTU dans la SAD pour se prémunir de ce genre d'attaque, par exemple à 576 octets en IPv4 et à 1280 octets pour IPv6 qui sont les valeurs minimum garanties, mais c'est alors au détriment des performances puisque l'on interdit alors toute taille de paquet supérieure.

Nous montrons maintenant que la situation est plus complexe qu'il n'y parait.

7.3 Déni de service sur les connexions TCP d'un utilisateur

Dans cette section nous détaillons la mise en place d'une attaque de type DoS basée sur des messages ICMP PTB. Nous l'illustrons en considérant une connexion TCP d'un utilisateur protégé par IPsec, mais le problème est identique avec d'autres protocoles (nous décrivons aussi l'attaque sur du trafic UDP).

7.3.1 Description de l'attaque

Cette attaque se déroule dans le cas d'un tunnel IPsec impliquant au moins une passerelle (on ne prend donc pas en compte les tunnels entre deux appareils nomades, le regroupement sur un seul appareil de la pile réseau de l'utilisateur et de l'implantation d'IPsec pouvant protéger de l'attaque). Nous visons dans un premier temps la passerelle impliquée dans le tunnel afin d'atteindre la connexion TCP dans un second temps. Notre attaque fonctionne aussi bien sur le client que sur le serveur de la connexion TCP.

7.3.1.1 Réduction du PMTU du tunnel

La première étape de l'attaque utilise un paquet ICMP PTB. Afin de contourner les mécanismes de protection d'IPsec contre les attaques "aveugles" (c'est-à-dire par l'envoi de paquet ICMP PTB simple), nous ajoutons en données ICMP une partie d'un paquet IP extérieur du tunnel IPsec. L'attaquant va donc écouter sur le réseau noir, récupérer un paquet IP extérieur d'un tunnel IPsec actif, puis créer un paquet ICMP PTB avec une valeur de MTU faible et copier le début du paquet IP extérieur en tant que données.

À ce point, le trafic de la connexion TCP n'est pas bloqué, mais le PMTU du tunnel a été mis à une valeur choisie (modulo le minimum accepté par la passerelle) par l'attaquant sur la passerelle visée, provoquant une réduction du débit due à la fragmentation désormais nécessaire que doit opérer la source.

7.3.1.2 Déni de service des connexions TCP

La seconde étape n'implique plus l'attaquant mais repose uniquement sur l'interaction entre IPsec et IP. En effet, une connexion TCP ouverte par un utilisateur sera alors sujette au PMTU injecté dans la passerelle pour la direction sortante. L'ouverture de connexion ne sera pas impactée car les trois messages de la poignée de main sont de petite taille, mais les paquets de données suivants pourront être limités par le PMTU de la passerelle. Nous montrons que le PMTU injecté dans la passerelle peut entrer en conflit avec les valeurs minimales imposées par IP, créant ainsi un DoS.

Réduction du PMTU en présence d'un tunnel : nous illustrons la fin de l'attaque dans le cas d'IPv4, pour lequel la taille minimale des paquets que doit supporter toute implantation est de 576 octets. À cause du mode tunnel utilisé par IPsec, le paquet IP intérieur est encapsulé dans un paquet ESP puis dans le paquet IP extérieur. La taille ajoutée par ces deux nouvelles encapsulations est d'au moins 34 octets, (elle est dépendante du chiffre et du mode utilisés, dans notre configuration 38 octets sont ajoutés). Si de plus, l'utilisateur met le bit DF à 1 et que l'attaque a mis le PMTU du tunnel à 576 (une valeur inférieure est peu probable à cause de la taille minimale imposée par IPv4), alors la passerelle IPsec va envoyer un paquet ICMP PTB à l'utilisateur pour réduire la taille des paquets qu'il envoie à une valeur de 538 octets ($576 - 38$). Ce message ICMP sera alors ignoré par la source, cette dernière considérant que des paquets IP de 576 octets doivent être supportés par la passerelle. La situation est alors bloquée entre l'utilisateur (qui n'envoie que des paquets de 576 octets) et la passerelle (qui demande une réduction de la taille des paquets), provoquant un déni de service sur la connexion TCP.

7.3.2 Attaque d'une passerelle IPsec générique

Nous mettons en place l'attaque précédente sur une passerelle et une machine utilisateur utilisant la distribution Linux Debian [8] ("Squeeze" avec un noyau 3.2.1) dans sa configuration de base. Nous commençons par valider que la connexion à un serveur appartenant à un réseau rouge distant en utilisant SSH fonctionne correctement.
Nous attirons l'attention du lecteur sur le fait que pour la fin du chapitre nous n'utilisons plus que IPv4 et confondons IP et IPv4. L'investigation a été uniquement faite sur IPv4, mais les mécanismes tels que la fragmentation ou la taille minimale sont communs à IPv6.

Notre première observation est que l'attaque "aveugle" ne fonctionne pas, comme prévu par les traitements d'ICMP dans IPsec. Désormais, tous les paquets ICMP PTB "d'attaque" auront en données le début d'un paquet IP extérieur du tunnel IPsec.

Une fois les messages ICMP PTB correctement créés par l'attaquant, nous observons que l'utilisateur ne peut plus utiliser le tunnel pour se connecter au serveur distant. Nous décrivons maintenant les deux étapes de l'attaque en IPv4 (Figure 7.2), du point de vue de la passerelle. La Figure 7.3 montre une capture du trafic du réseau rouge entre une passerelle IPsec et un serveur SSH lors de la connexion d'un client via un tunnel attaqué.

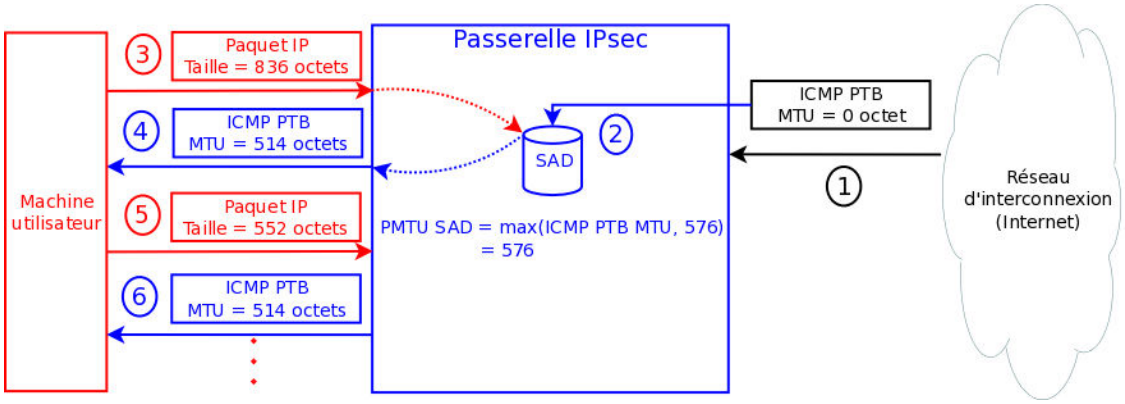


FIGURE 7.2 – Déroulement d'une attaque par ICMP PTB sur passerelle IPsec, cas PMTUd

Attaque ICMP de la passerelle : lors de l'envoi depuis le réseau noir d'un paquet ICMP PTB (étape 1), IPsec stocke dans la SAD (après les vérifications d'usage) la valeur :

$$PMTU_{SAD} = max(MTU_{ICMP\ PTB}, 576) = 576, \text{ (étape 2.)}$$

Propagation du PMTU noir sur le réseau rouge : lors de la réception d'un paquet IP intérieur dépassant le PMTU (étape 3, paquet horodaté `0.004933`), la passerelle crée un paquet ICMP PTB annonçant un MTU à respecter de :

$$MTU = PMTU_{SAD} - taille_{encapsulation\ IP/IPsec/ESP} \text{ (étape 4.)}$$

Dans notre trace (horodatage `0.004953`), la valeur du MTU annoncée est de 514 octets. La taille annoncée est sous–optimale : comme montré au paragraphe 7.3.1.2 un paquet IP de 538 octets n'est pas affecté par cette première phase de l'attaque.

```
0.000000 a.b.10.7.48058 > a.b.11.5.ssh:   S *:*(0) win 17920 <mss 8960,sackOK,timestamp 1245892 0,nop,wscale 7> (DF)
0.000146 a.b.11.5.ssh   > a.b.10.7.48058: S *:*(0) ack * win 17896 <mss 8960,sackOK,timestamp 1319280 1245892,nop,wscale 7> (DF)
0.000304 a.b.10.7.48058 > a.b.11.5.ssh: . ack 1 win 140 <nop,nop,timestamp 1245892 1319280> (DF)
0.004561 a.b.11.5.ssh   > a.b.10.7.48058: P 1:33(32) ack 1 win 140 <nop,nop,timestamp 1319281 1245892> (DF)
0.004698 a.b.10.7.48058 > a.b.11.5.ssh:   . ack 33 win 140 <nop,nop,timestamp 1245893 1319281> (DF)
0.004773 a.b.10.7.48058 > a.b.11.5.ssh:   P 1:33(32) ack 33 win 140 <nop,nop,timestamp 1245893 1319281> (DF)
0.004858 a.b.11.5.ssh   > a.b.10.7.48058: . ack 33 win 140 <nop,nop,timestamp 1319281 1245893> (DF)
0.004933 a.b.11.5.ssh   > a.b.10.7.48058: P 33:817(784) ack 33 win 140 <nop,nop,timestamp 1319281 1245893> (DF)
0.004953 a.b.11.4       > a.b.11.5:       ICMP ERROR: a.b.10.7 unreachable - need to frag (mtu 514) [tos 0xc0]
0.004998 a.b.10.7.48058 > a.b.11.5.ssh:   P 33:881(848) ack 33 win 140 <nop,nop,timestamp 1245893 1319281> (DF)
0.005084 a.b.11.5.ssh   > a.b.10.7.48058: . 33:533(500) ack 33 win 140 <nop,nop,timestamp 1319281 1245893> (DF)
0.005092 a.b.11.4       > a.b.11.5:       ICMP ERROR: a.b.10.7 unreachable - need to frag (mtu 514) [tos 0xc0]
0.005095 a.b.11.5.ssh   > a.b.10.7.48058: P 533:817(284) ack 33 win 140 <nop,nop,timestamp 1319281 1245893> (DF)
0.005228 a.b.10.7.48058 > a.b.11.5.ssh:   . ack 33 win 140 <nop,nop,timestamp 1245893 1319281,nop,nop,sack 1 {533:817} > (DF)
0.043580 a.b.11.5.ssh   > a.b.10.7.48058: . ack 881 win 154 <nop,nop,timestamp 1319291 1245893> (DF)
0.215586 a.b.11.5.ssh   > a.b.10.7.48058: . 33:533(500) ack 881 win 154 <nop,nop,timestamp 1319334 1245893> (DF)
0.215594 a.b.11.4       > a.b.11.5:       ICMP ERROR: a.b.10.7 unreachable - need to frag (mtu 514) [tos 0xc0]
0.639580 a.b.11.5.ssh   > a.b.10.7.48058: . 33:533(500) ack 881 win 154 <nop,nop,timestamp 1319440 1245893> (DF)
0.639586 a.b.11.4       > a.b.11.5:       ICMP ERROR: a.b.10.7 unreachable - need to frag (mtu 514) [tos 0xc0]
```

FIGURE 7.3 – Trace *tcpdump* du trafic sur le réseau rouge lors de l'attaque, cas PMTUd. Ici, la machine du client distant (adresse IP a.b.10.7) essaie de se connecter au serveur SSH local d'adresse a.b.11.5, situé derrière une passerelle IPsec d'adresse locale a.b.11.4 (NB : les informations annexes ont été retirées).

Boucle IP/ICMP sur le réseau rouge : nous observons que la machine utilisateur renvoie un paquet IP de taille moindre (étape 5, horodatage `0.005084`), mais il est toujours trop gros (552 octets) comparé à la taille acceptée par la passerelle. Il est donc rejeté et donne naissance à un nouveau message ICMP PTB annonçant un MTU de 514.

L'attaque DoS d'une passerelle IPsec sous Debian est donc un succès, nous sommes en mesure d'interdire l'envoi de gros paquets de données par l'utilisateur. Par exemple, un utilisateur n'est alors plus capable de se connecter à un serveur SSH via la passerelle attaquée.

La prochaine section étudie donc plus en détail le comportement des protocoles et algorithmes sur la machine utilisateur afin de trouver une parade à cette attaque.

7.3.3 Comportement logiciel de la machine utilisateur

Nous cherchons ici à comprendre le comportement de la machine utilisateur, en particulier la taille du paquet IP à l'étape 5, de 552 octets, alors que la passerelle a annoncé un MTU de 514 octets.

7.3.3.1 Fonctionnement du PMTUd

De manière similaire au calcul du PMTU par IPsec lors de la mise à jour de la SAD, PMTUd calcule la valeur du PMTU, lors de la réception d'un paquet ICMP PTB, de la manière suivante :

$PMTU = max(MTU_{configuration}, MTU_{ICMP\ PTB})$, avec $MTU_{configuration} = 552$ par défaut.

Le problème provient donc de la valeur configurée de 552 octets qui empêche la prise en compte du MTU annoncé par le message ICMP. Cette valeur est arbitraire et propre à Debian (Squeeze), Linux 3.2.1 utilisant pour sa part 562 octets, ce qui là aussi permet à l'attaque DOS de réussir.

PMTUd n'étant pas le seul algorithme de recherche de PMTU, nous configurons la machine utilisateur pour utiliser PLPMTUd qui fonctionne sans utiliser les messages ICMP PTB afin de déterminer si cela peut constituer une parade à l'attaque.

7.3.3.2 Fonctionnement du PLPMTUd

Nous détaillons les échanges réseau qui ont lieu dans cette attaque (Figure 7.4) et la trace de notre mise en œuvre (Figure 7.5). Les deux premières étapes (mise à 576 du PMTU dans la SAD) sont identiques au cas avec PMTUd.

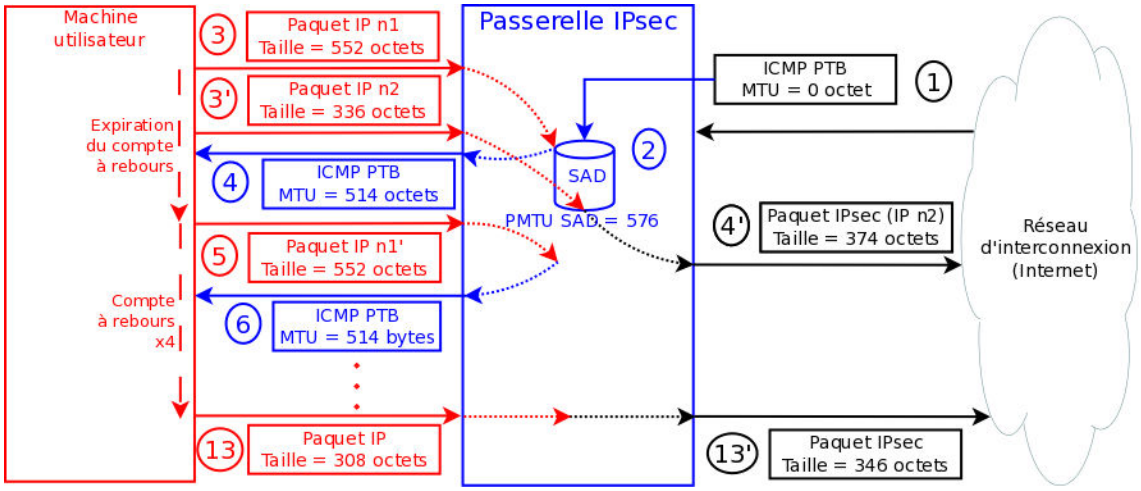


FIGURE 7.4 – Attaque par ICMP PTB sur passerelle IPsec, cas PLPMTUd

Gestion des fragments par la passerelle : l'envoi des données utilisateur doit désormais se faire en deux paquets IP (étapes 3 et 3', horodatage `0.004711` et `0.004721`) puisque l'implantation de PLPMTUd par Linux se base sur TCP et utilise la taille maximale des segments (ou MSS, Maximum Segment Size [99]), initialisée à 512, comme valeur initiale pour la taille des sondes. Le paquet de 836 octets est devenu deux paquets de 552 et 336 octets, car les en-têtes IP et TCP sont de 52 octets dans notre cas (à cause de l'option "timestamp" deTCP) et que l'application au dessus de TCP cherche à envoyer 784 octets, coupés en 500 et 284 octets.

Le premier paquet, après ajout des en–têtes, étant plus grand que le PMTU stocké dans la SAD, la passerelle génère un message ICMP PTB (l'étape 4, horodatage `0.004719`). Le second passera la vérification de taille et sera traité par IPsec à l'étape 4' (mais non délivré par TCP à l'application distante car il manquera des données dans le flux TCP).

ICMP PTB et PLPMTUd : finalement, à l'étape 5, après expiration du compte à rebours TCP (initialisé à deux fois le RTT de la poignée de main) la machine utilisateur renvoie le premier paquet sans baisser pour autant la taille.

Puis, après l'expiration du cinquième compte à rebours, PLPMTUd réduit la taille TCP des sondes à 256 (étape 13, horodatage `6.586634`). Les paquets sont désormais suffisamment petits pour être traités par la passerelle (étape 13'). Après plus de six secondes d'attente, l'utilisateur peut enfin poursuivre sa connexion au serveur SSH. Toutefois, tous les paquets IP envoyés par le serveur font au maximum 308 octets. Pour comparaison, en 2004, un utilisateur tolérait une attente de deux secondes sur le Web [91].

Selon le mécanisme de recherche du PMTU utilisé par la machine utilisateur du réseau rouge de la passerelle attaquée, l'attaque permet soit d'interdire toute communication

```
0.000000 a.b.10.7.48063 > a.b.11.5.ssh:   S *:*(0) win 17920 <mss 8960,sackOK,timestamp 1572549 0,nop,wscale 7> (DF)
0.000142 a.b.11.5.ssh   > a.b.10.7.48063: S *:*(0) ack * win 17896 <mss 8960,sackOK,timestamp 1645937 1572549,nop,wscale 7> (DF)
0.000417 a.b.10.7.48063 > a.b.11.5.ssh:   . ack 1 win 140 <nop,nop,timestamp 1572550 1645937> (DF)
0.004208 a.b.11.5.ssh   > a.b.10.7.48063: P 1:33(32) ack 1 win 140 <nop,nop,timestamp 1645938 1572550> (DF)
0.004535 a.b.10.7.48063 > a.b.11.5.ssh:   . ack 33 win 140 <nop,nop,timestamp 1572551 1645938> (DF)
0.004538 a.b.10.7.48063 > a.b.11.5.ssh:   P 1:33(32) ack 33 win 140 <nop,nop,timestamp 1572551 1645938> (DF)
0.004676 a.b.11.5.ssh   > a.b.10.7.48063: . ack 33 win 140 <nop,nop,timestamp 1645938 1572551> (DF)
0.004688 a.b.10.7.48063 > a.b.11.5.ssh:   . 33:545(512) ack 33 win 140 <nop,nop,timestamp 1572551 1645938> (DF)
0.004711 a.b.11.5.ssh   > a.b.10.7.48063: . 33:533(500) ack 33 win 140 <nop,nop,timestamp 1645938 1572551> (DF)
0.004719 a.b.11.4       > a.b.11.5:       ICMP ERROR: a.b.10.7 unreachable - need to frag (mtu 514) [tos 0xc0]
0.004721 a.b.11.5.ssh   > a.b.10.7.48063: P 533:817(284) ack 33 win 140 <nop,nop,timestamp 1645938 1572551> (DF)
0.004960 a.b.10.7.48063 > a.b.11.5.ssh:   P 545:881(336) ack 33 win 140 <nop,nop,timestamp 1572551 1645938> (DF)
0.005006 a.b.10.7.48063 > a.b.11.5.ssh:   . ack 33 win 140 <nop,nop,timestamp 1572551 1645938,nop,nop,sack 1 {533:817} > (DF)
0.005046 a.b.11.5.ssh   > a.b.10.7.48063: . ack 881 win 156 <nop,nop,timestamp 1645938 1572551> (DF)
0.214634 a.b.11.5.ssh   > a.b.10.7.48063: . 33:533(500) ack 881 win 156 <nop,nop,timestamp 1645991 1572551> (DF)
0.214643 a.b.11.4       > a.b.11.5:       ICMP ERROR: a.b.10.7 unreachable - need to frag (mtu 514) [tos 0xc0]
0.638636 a.b.11.5.ssh   > a.b.10.7.48063: . 33:533(500) ack 881 win 156 <nop,nop,timestamp 1646097 1572551> (DF)
0.638646 a.b.11.4       > a.b.11.5:       ICMP ERROR: a.b.10.7 unreachable - need to frag (mtu 514) [tos 0xc0]
1.486639 a.b.11.5.ssh   > a.b.10.7.48063: . 33:533(500) ack 881 win 156 <nop,nop,timestamp 1646309 1572551> (DF)
1.486645 a.b.11.4       > a.b.11.5:       ICMP ERROR: a.b.10.7 unreachable - need to frag (mtu 514) [tos 0xc0]
3.186646 a.b.11.5.ssh   > a.b.10.7.48063: . 33:533(500) ack 881 win 156 <nop,nop,timestamp 1646734 1572551> (DF)
3.186655 a.b.11.4       > a.b.11.5:       ICMP ERROR: a.b.10.7 unreachable - need to frag (mtu 514) [tos 0xc0]
6.586634 a.b.11.5.ssh   > a.b.10.7.48063: . 33:289(256) ack 881 win 156 <nop,nop,timestamp 1647584 1572551> (DF)
6.586831 a.b.10.7.48063 > a.b.11.5.ssh:   . ack 289 win 148 <nop,nop,timestamp 1574196 1647584,nop,nop,sack 1 {533:817} > (DF)
6.586941 a.b.11.5.ssh   > a.b.10.7.48063: . 289:533(244) ack 881 win 156 <nop,nop,timestamp 1647584 1574196> (DF)
6.587143 a.b.10.7.48063 > a.b.11.5.ssh:   . ack 817 win 156 <nop,nop,timestamp 1574196 1647584> (DF)
6.587147 a.b.10.7.48063 > a.b.11.5.ssh:   P 881:905(24) ack 817 win 156 <nop,nop,timestamp 1574196 1647584> (DF)
6.588458 a.b.11.5.ssh   > a.b.10.7.48063: P 817:969(152) ack 905 win 156 <nop,nop,timestamp 1647584 1574196> (DF)
6.589189 a.b.10.7.48063 > a.b.11.5.ssh:   P 905:1049(144) ack 969 win 164 <nop,nop,timestamp 1574197 1647584> (DF)
6.593662 a.b.11.5.ssh   > a.b.10.7.48063: . 969:1225(256) ack 1049 win 164 <nop,nop,timestamp 1647585 1574197> (DF)
6.593739 a.b.11.5.ssh   > a.b.10.7.48063: . 1225:1481(256) ack 1049 win 164 <nop,nop,timestamp 1647585 1574197> (DF)
6.593750 a.b.11.5.ssh   > a.b.10.7.48063: P 1481:1689(208) ack 1049 win 164 <nop,nop,timestamp 1647585 1574197> (DF)
6.593946 a.b.10.7.48063 > a.b.11.5.ssh:   . ack 1481 win 176 <nop,nop,timestamp 1574198 1647585> (DF)
```

FIGURE 7.5 – Trace *tcpdump* du trafic sur le réseau rouge lors de l'attaque, cas PLPMTUd. Les notations sont identiques à celles de la Figure 7.3

.

(grand paquet IP avec le bit DF à 1) soit de réduire le débit de la connexion. Le déroulement et la prise en compte des paquets échangés dans chacun des deux cas n'est pas identique et des valeurs par défaut différentes sont utilisées.

Nous regardons plus en détail l'importance de la configuration des algorithmes de découverte du PMTU.

7.3.3.3 Effets de la configuration locale de l'utilisateur sur l'attaque

Tout d'abord, nous rappelons que la présence de notre tunnel IPsec ajoute 38 octets de plus à un paquet IP intérieur à cause des différentes encapsulations (en-têtes IP extérieur, IPsec et ESP). De plus, l'attaquant est capable de mettre la valeur du PMTU de la SAD à 576. Un utilisateur doit donc envoyer des paquets IP de 538 octets ou moins pour éviter d'être impacté par l'attaque.

La configuration du PMTUd est directement la valeur qu'IP va utiliser pour le calcul du PMTU, comme précisé à la section 7.3.3.1. Si l'utilisateur modifie le $MTU_{configuration}$ pour permettre la prise en compte du MTU annoncé par la passerelle, alors il pourra s'affranchir de l'attaque.

La configuration du PLPMTUd passe par la valeur de base du MSS. L'utilisateur doit donc retrancher la taille des en-têtes IP intérieur et TCP par rapport à la taille du MTU (dans notre cas, 52 octets), soit un MSS de 486 octets.

Les résultats de l'attaque pour les deux algorithmes et leur configuration sont résumés dans la Table 7.2. Nous considérons que malgré une configuration correcte de la machine

Algorithme	Valeur minimale	Résultat de l'attaque
PMTUd	538 ou inférieur	Réduction de débit
PMTUd	539 ou supérieur	Déni de service
PLPMTUd	486 ou inférieur	Réduction de débit
PLPMTUd	487 ou supérieur	Réduction débit plus importante (paquets de 308 octets)

TABLE 7.2 – Effet de l'attaque via ICMP PTB selon la configuration Linux

utilisateur, l'attaque a au moins permis une réduction de débit en forçant l'utilisation de paquets IP de faible taille.

Cette solution n'est pas satisfaisante, car même si elle permet à l'utilisateur de retrouver sa connexion avec la machine distante, elle nécessite une action sur les machines des réseaux rouges. De plus les utilisateurs peuvent ne pas être au courant de la présence de tunnel ou d'autres tunnels peuvent être utilisés ce qui réduirait encore les valeurs minimales requises. Le problème reste donc entier.

Avant de continuer l'analyse de ce problème, nous détaillons les effets de l'attaque sur un flux UDP.

7.4 Effet de l'attaque sur un flux UDP

Dans les cas précédemment étudiés, l'utilisation du protocole TCP forçait la mise à 1 du bit DF. Ce paragraphe met en avant notre attaque dans le cas d'un protocole ne forçant pas le bit DF.

La Figure 7.7 illustre le déroulement dans le cas du protocole UDP. Les deux premières étapes de mise à sa valeur minimale du PMTU dans la SAD sont identiques. La Figure 7.6 (respectivement 7.8) montre une trace du trafic sur le réseau rouge (resp. noir) une fois l'attaque effectuée.

Perte du premier paquet IP

L'application cliente cherche à envoyer 1100 octets de données en UDP sur le port destination `21513`. Pour information, la pile réseau ajoute 28 octets d'en–têtes IP et UDP. La machine utilisateur envoie un premier datagramme UDP contenant toutes les données. Ce datagramme est encapsulé dans un paquet IP avec le bit DF mis à 1 (étape 3, horodatage `0.000000`). La passerelle rejette le paquet à cause du bit DF à 1 et envoie un message ICMP PTB (étape 4, horodatage `0.000026`). Ce datagramme est définitivement perdu car UDP ne stocke pas les données transmises.

```
0.000000 a.b.11.5.58921 > a.b.10.7.21513: udp 1100 (DF)
0.000026 a.b.11.4 > a.b.11.5: ICMP ERROR: a.b.10.7 unreachable - need to frag (mtu 514) [tos 0xc0]
[...]
3.000000 a.b.11.5.43117 > a.b.10.7.21513: udp 1100 (frag 9719:528@0+)
3.000067 a.b.11.5 > a.b.10.7: (frag 9719:528@528+)
3.000083 a.b.11.5 > a.b.10.7: (frag 9719:52@1056)
```

FIGURE 7.6 – Trace *tcpdump* du trafic sur le réseau rouge lors de l'attaque, cas UDP. Les notations sont identiques à celles de la Figure 7.3

.

Fragmentation par la machine utilisateur

Lors de l'émission de nouvelles données par l'application (en direction de la même adresse IP et du même port UDP destination), la pile réseau va fragmenter les 1100 octets (plus l'en–tête UDP) dans trois paquets IP (étapes 5, 6 et 7, horodatage `3.000000, 3.000067, 3.000083`). De plus, le bit DF est à 0.

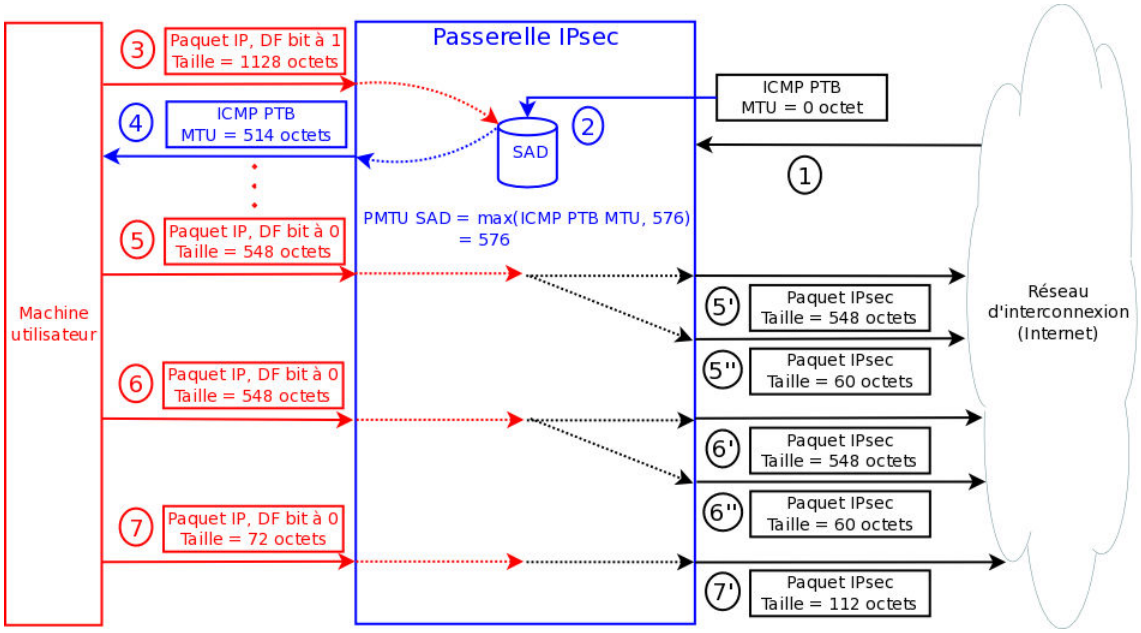


FIGURE 7.7 – Attaque par ICMP PTB sur passerelle IPsec, cas UDP

Fragmentation par la passerelle IPsec

Une fois le chiffrement effectué par la passerelle IPsec, les deux premiers paquets IP s'avèrent plus gros que le PMTU stocké dans la SAD. Toutefois, contrairement au cas du protocole TCP, le bit DF étant à 0, la passerelle peut fragmenter (à nouveau) pour tenir la contrainte de taille. Comme montré par la Figure 7.8, on obtient cinq paquets IPsec sur le réseau noir, dont deux de très petite taille (étapes 5" et 6", horodatage `3.000037, 3.000081`).

```
3.000033 esp a.b.20.4 > a.b.20.8 spi 0x00005fb5 seq 131 len 528 (frag 42480:528@0+)
3.000037 a.b.20.4 > a.b.20.8: (frag 42480:40@528)
3.000078 esp a.b.20.4 > a.b.20.8 spi 0x00005fb5 seq 132 len 528 (frag 42481:528@0+)
3.000081 a.b.20.4 > a.b.20.8: (frag 42481:40@528)
3.000085 esp a.b.20.4 > a.b.20.8 spi 0x00005fb5 seq 133 len 92
```

FIGURE 7.8 – Trace *tcpdump* du trafic sur le réseau noir lors de l'attaque, cas UDP. Les notations sont identiques à celles de la Figure 7.3

.

Dans le cas d'un trafic qui laisse le bit DF à 0, l'attaque permet donc de fortement réduire le débit en augmentant le nombre de paquets de manière exagérée : alors qu'après notre attaque seulement trois paquets IPsec seraient suffisants, la passerelle a besoin d'émettre cinq paquets pour transmettre les données.

Nous analysons donc plus en détail la limite mise en avant par notre attaque et montrons qu'elle est intrinsèque à IP, avant de proposer une solution plus globale que l'aspect configuration des machines utilisatrices de la passerelle.

7.5 Surcoût des tunnels vs taille minimale des paquets IP

Au-delà de l'attaque précédente, un blocage similaire peut avoir lieu dans le cas légitime de la Figure 7.9. Alors que la norme IP impose de pouvoir gérer une taille minimale de paquet, la présence d'un tunnel fais grossir le paquet IP intérieur au dessus de cette taille minimale. Dès lors, même si chaque lien respecte la norme "localement", le surcoût de l'encapsulation bloquera le paquet à l'échelle du chemin.

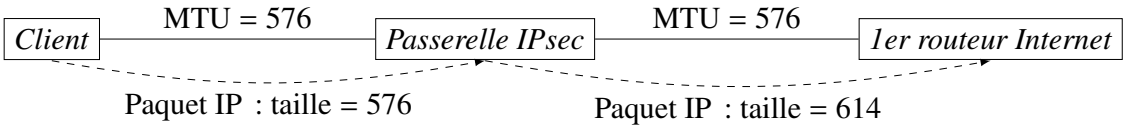


FIGURE 7.9 – Cas légitime de blocage dû à la présence d'un tunnel

Le problème fondamental provient donc de la gestion locale de la limitation de la taille des paquets (i.e. un appareil doit pouvoir émettre un paquet d'au moins 576 octets sur les liens physiques auxquels il est connecté), alors qu'un tunnel peut augmenter la taille d'un paquet IP sans que son émetteur en soit informé.

En respectant les normes de la suite IP, il n'est pas possible de résoudre ce problème. En effet, IP impose le fait de pouvoir utiliser une taille de paquet de 576 octets, mais peut interdire (via le bit DF) à toute machine intermédiaire de fragmenter le paquet et les connexions IPsec en mode tunnel nécessitent de nouveaux en-têtes et donc un surcoût de taille.

Certes ce blocage fondamental des normes est insoluble, néanmoins nous analysons l'architecture en rupture pour trouver une solution à l'attaque par ICMP PTB.

7.6 Recherche de PMTU dans IPsec

Le souci fondamental permettant le succès de l'attaque est la mauvaise gestion du PMTU sur le réseau noir lorsque l'on se repose sur les seuls messagesICMP PTB issus du réseau noir. Ceci reste indépendant de l'utilisation de l'algorithme de recherche de taille maximum de chemin utilisé sur les machines utilisateur (PMTUd ou PLPMTUd).

Nous proposons donc d'utiliser un mécanisme similaire au PLPMTUd sur une passerelle pour évaluer le PMTU "noir", plutôt que de recourir aux messages ICMP PTB comme fait actuellement. Dans le cas d'une architecture en rupture, le module de traitement réseau rouge se situe avant le tunnel, et peut donc estimer le PMTU de manière équivalente à une machine cliente, c'est-à-dire en prenant en compte le surcoût dû au tunnel. Les deux modules de traitement rouge des passerelles (ou la partie de la pile réseau correspondant au module de traitement rouge pour les appareils nomades et les passerelles intégrées) échangeront un ensemble de messages permettant d'évaluer le PMTU, qu'ils pourront ensuite transmettre aux clients si besoin. Idéalement, une passerelle IPsec aura alors à

évaluer autant de PMTU qu'il y a de politiques de sécurité IPsec. Cette solution souffre toutefois de faiblesses :

- le module de traitement rouge n'effectue que les traitements protocolaires avant IPsec, qui eux s'effectuent sur le module matériel. C'est donc sur le HSM que se trouvent les bases de données d'IPsec. Il faudra donc évaluer le PMTU pour chaque destinataire "rouge", c'est-à-dire machine d'un réseau local rouge ou appareil nomade. Le module de traitement rouge ne connaît ni les adresses de destination de chacun des réseaux rouges ni les politiques à appliquer (qui peuvent être multiples pour un même réseau rouge distant) dont dépend le surcoût des en-têtes ;

- les échanges d'évaluation du PMTU noir se font en sus des communications des clients et d'IPsec, et de manière systématique : pour chaque valeur possible dans IPsec du n-uplet des sélecteurs (adresses IP, protocole transport et ses identifiants - numéros de ports par exemple, et nom de l'entrée dans la SAD), il faut évaluer au préalable le PMTU pour le chemin. Il faut de plus prévoir un mécanisme de maintien en vie (ou "keepalive") pour vérifier que le PMTU initialement découvert est toujours valide.

  Ce mécanisme est donc plus coûteux que lorsqu'IPsec se base sur les messages ICMP PTB pour lesquels les paquets supplémentaires ne sont générés qu'en cas de besoin par les routeurs. De plus, il ne permet pas de recouvrir la taille des sondes en les utilisant pour transporter des données comme dans le cas du PLPMTUd sur une machine utilisateur.

Cette solution permet de s'affranchir totalement de l'attaque (plus de réduction de débit puisque la taille réelle supportée par le réseau est utilisable) grâce au fait de ne reposer que sur le routage IP et plus du tout sur les paquets ICMP noirs. De plus, elle est transposable sur les appareils nomades et passerelles intégrées (afin d'être interopérable avec les passerelles en rupture). Toutefois, elle est d'une part plus coûteuse en nombre de paquets / utilisation du débit et d'autre part moins réactive au changement de topologie du réseau.

Conclusion

Nous avons montré dans ce chapitre qu'une passerelle IPsec présente une faille exploitable dans sa gestion des messages ICMP PTB. De plus, nous avons mis en place cette attaque qui aboutit à un déni de service dans le cas d'une connexion SSH par exemple. Plus précisement, selon l'utilisation d'IP (bit DF à 0 ou 1) et la configuration du PMTUd ou du PLPMTUd sur les machines des utilisateurs l'attaque mène à un déni de service ou une réduction du débit utilisable car la réduction de la taille des paquets va augmenter le nombre de paquets à traiter.

Nous avons proposé une contre–mesure à cette attaque dans le cas d'une passerelle en rupture, extensible aux clients nomades et passerelles intégrées, mais elle ne résout pas le problème fondamental : les normes IP et IPsec n'ont pas prévu la gestion de la taille minimale acceptée lors de la présence d'un tunnel, qui augmente la taille des paquets à cause des en-têtes supplémentaires.

La troisième partie de cette thèse expose le choix entre les deux architectures de passerelle possible dans le cadre du projet SHIVA et présente les conclusions et perspectives de ce travail.

Troisième partie

Conclusions et persectives

Chapitre 8

Implantation d'un prototype de passerelle SHIVA

Sommaire

8.1 Choix de l'architecture 85

8.2 Prototypes de passerelle SHIVA 86

8.2.1 Prototype avec émulateur de chiffrement 86

8.2.2 Prototype avec carte de chiffrement 87

Ce chapitre présente le choix de l'architecture fait par les partenaires du projet SHIVA ainsi que les prototypes développés.

8.1 Choix de l'architecture

Nous avons proposé deux architectures (Figure 8.1) pour le projet SHIVA, dont le but est de concevoir et développer un prototype de passerelle VPN IPsec sécurisé à très haut débit. Ces propositions ont été évaluées sur deux principaux critères :

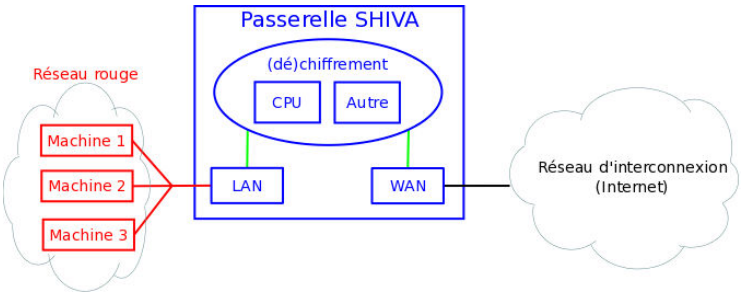
La capacité de chiffrement

La passerelle doit pouvoir traiter des flux de paquets Ethernet/IP à un débit de 10 Gb/s (à la fois en chiffrement et en communication réseau).

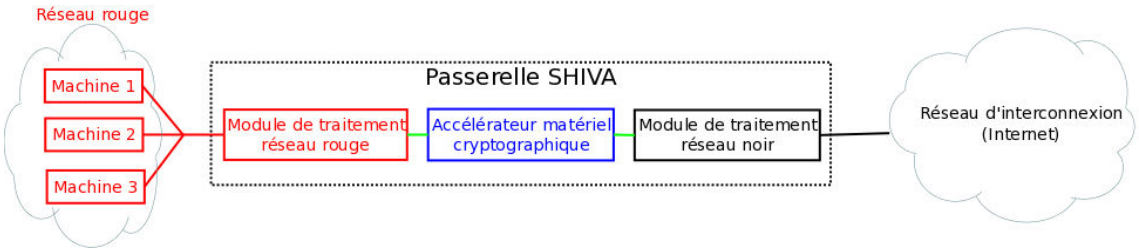
Nous avons montré que l'architecture intégrée mise en œuvre sur un serveur générique n'a pas les capacités de calcul nécessaires pour soutenir le débit visé. L'architecture en rupture permet d'envisager de meilleures performances (sous réserve que le module matériel tienne le débit) même si son implantation n'est pas immédiate (en particulier du fait du besoin de paralléliser les traitements sur les modules de traitement réseau).

La sécurité de la passerelle

Le projet SHIVA est ambitieux en terme de sécurisation des passerelles de chiffrement. Il considère en effet des attaquants ayant accès à des ressources illimitées, autant de temps que nécessaire pour réaliser ses attaques et l'accès physique au matériel (l'annexe A.1 résume quelques attaques considérées dans SHIVA). Nous



(a) Architecture d'une passerelle intégrée.



(b) Architecture d'une passerelle en rupture.

FIGURE 8.1 – Architectures proposées pour la passerelle SHIVA

nous sommes particulièrement intéressés aux attaques réseau sur les passerelles, spécialement par rapport à la gestion d'ICMP.

Nous avons démontré qu'une attaque basée sur les messages ICMP PTB est possible sur une passerelle intégrée. Nous avons proposé une contre-mesure pour les architectures en rupture, qui toutefois ne permet pas de résoudre le problème fondamental du surcoût des tunnels.

Vu les points précédents, les partenaires du projet SHIVA ont choisi l'architecture en rupture pour les passerelles IPsec qui présente de meilleures performances et sécurité.

Nous présentons maintenant deux versions du prototype que nous avons implanté pendant le projet SHIVA.

8.2 Prototypes de passerelle SHIVA

Le prototype final du projet SHIVA ne sera pas présenté dans cette thèse. Les deux versions intermédiaires sont le prototype de test fonctionnel avec un émulateur du HSM et le prototype avec un module de chiffrement matériel intégré dans un serveur (passerelle hybride entre les deux architectures).

8.2.1 Prototype avec émulateur de chiffrement

Afin de tester l'architecture globale d'une passerelle SHIVA, nous avons remplacé l'HSM de SHIVA par un serveur (indépendant des modules de traitements) émulant logiciellement les traitements du module, en particulier l'encapsulation protocolaire ESP. Ce protocole a permis de valider fonctionnellement les traitements des protocoles réseau par une passerelle en rupture.

Une fois ce prototype validé, nous avons donc intégré une version intermédiaire du module développé pendant le projet SHIVA.

8.2.2 Prototype avec carte de chiffrement

Le module utilisé pour ce prototype a été développé sur la base d'une carte FPGA existante afin de valider un certain nombre de briques matérielles. Ces développements, qui ont été effectués par un partenaire du projet SHIVA, ne sont pas détaillés dans cette thèse.

Cette carte intermédiaire ne propose pas toutes les fonctionnalités requises. En particulier les interfaces d'entrée/sortie à 10 Gb/s sont constituées du seul bus PCIe. Lors de la validation de ce prototype, les débits atteints ont été faibles (de l'ordre de 400 Mb/s) alors que les modules de traitement développés sous Click supportent une charge au moins six fois supérieure.

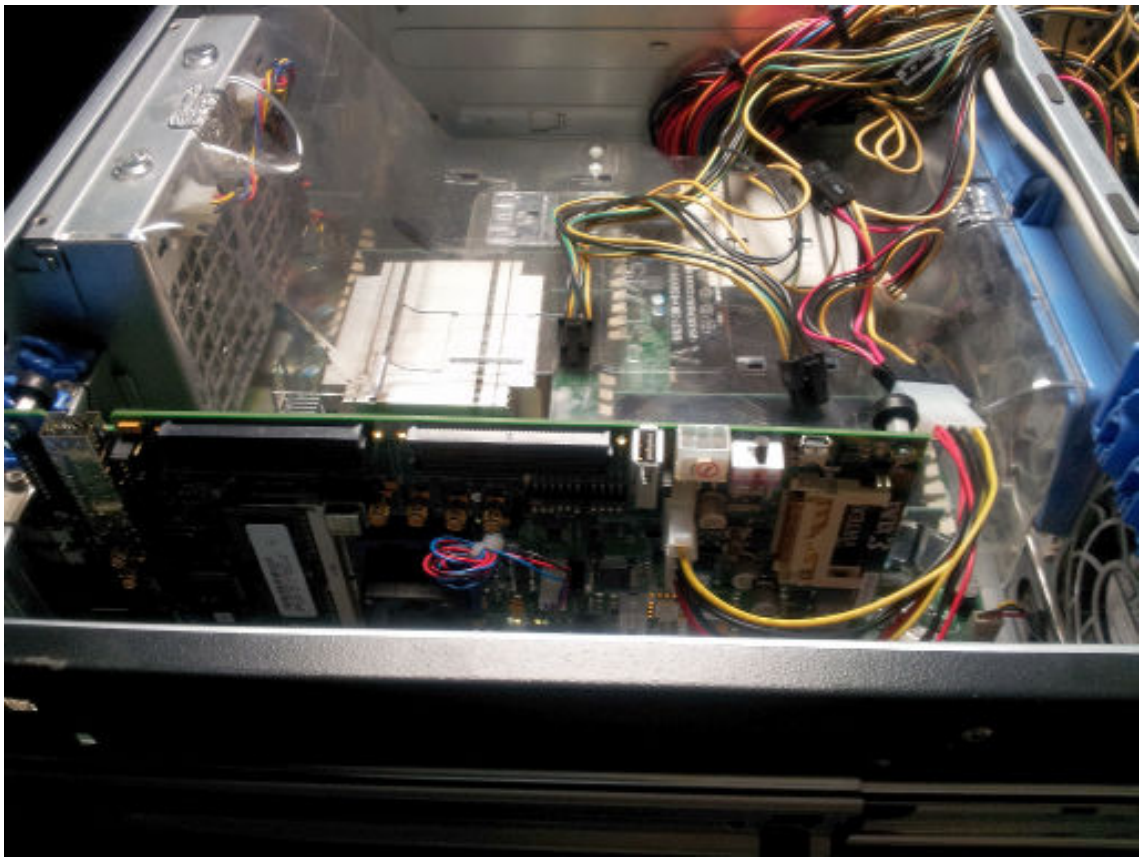


FIGURE 8.2 – Intégration du prototype de module SHIVA dans une passerelle

Deux explications à ces résultats :

– l'intégration de la carte au sein même d'une des machines de traitement pose le problème des interruptions. En effet, le transfert de données via le bus PCIe requiert des actions logicielles qui ne seront plus présentes dans la version finale.

– Au moment des mesures, la carte supportait inefficacement les données de petite taille, telles qu'imposées par les paquets réseau (soit 1500 octets, alors que la carte fonctionnait de manière optimale pour une taille de 4 Kio).

Toutefois, ce prototype a permis de valider plusieurs points. L'utilisation d'une carte (ici générique dans sa construction et son intégration dans la machine mais programmée via les briques "IP") permet de réduire l'utilisation des processeurs laissant de la puissance de calcul pour des services réseau complémentaires, comme un pare-feu par exemple. Pour comparaison, en effectuant le chiffrement de manière logicielle, les performances sont 20 % plus faibles.

La seconde version du HSM n'ayant pas été livrée, elle n'a pas pu être intégré dans notre plateforme de validation. Nous ne pouvons donc pas donner de résultat avec une version théoriquement plus performante du module matériel.

Nous présentons maintenant les conclusions de cette thèse ainsi que ses perspectives.

Chapitre 9

Enseignements et perspectives

Sommaire

9.1 Chiffrement de communications réseau à très haut débit 89

9.1.1 Nécessité d'une accélération matérielle pour le chiffrement . . . 89

9.1.2 Sécurisation des passerelles de chiffrement par architecture en rupture 90

9.1.3 Parallélisation des traitements réseau 91

9.2 Interactions entre ICMP et IPsec 92

9.2.1 ICMP nécessaire pour des communications réseau performantes ? 92

9.2.2 Vulnérabilité des passerelles IPsec à ICMP 93

Nous discutons dans ce chapitre des différents enseignements apportés par cette thèse ainsi que les perspectives pour des travaux futurs.

9.1 Chiffrement de communications réseau à très haut débit

Dans un premier temps, nous discutons les points que nous avons adressés par rapport aux capacités de calcul nécessaires pour le chiffrement de trafic réseau à très haut débit.

9.1.1 Nécessité d'une accélération matérielle pour le chiffrement

Les serveurs génériques supportent sans souci les communications à très haut débit (sans modification matérielle), grâce à différentes avancées technologiques désormais communes : mode "polling", cartes réseau multi-queues, augmentation du nombre d'unités de calcul. Les cartes réseau multi-queues sont particulièrement intéressantes car elles permettent de trier grossièrement par flux le trafic réseau en amont de tout traitement logiciel.

Par contre, comme démontré par notre étude de 2009, l'ajout du chiffrement pour sécuriser les données échangées lors de communications à 10 Gb/s se heurte à l'impossibilité de traiter un très grand nombre de petits paquets (a contrario traiter un petit nombre de grands paquets est facile). Plus récemment, les travaux de Salamatian et al. [60, 81] sur la

recherche de route IP à très haut débit (40 Gb/s et 100 Gb/s) ont montré que ces traitements nécessitent des développements (matériels et logiciels) spécifiques.

Si une solution avec chiffrement logiciel est utilisable pour le grand public dont l'ordre de grandeur du débit réseau est moindre (1 Gb/s actuellement), elle n'est pas adaptée au chiffrement réseau à très haut débit, en particulier sur des passerelles IPsec qui agrègent un grand nombre de flux.

Aussi, malgré les dernières avancées technologiques (intégration d'instructions spécifique à AES dans les processeurs par exemple), nous pensons que la solution de chiffrement logiciel sur un unique serveur sera toujours en retrait comparée aux évolutions des cartes réseau vers le très haut débit (les normes Ethernet visent désormais 100 Gb/s et plus).

En effet, dans le prolongement des architectures à unités de calcul hautement parallélisées introduites par le GPGPU, les fabricants de processeurs proposent désormais, sur le marché des serveurs, des architectures à grand nombre de cœurs, comme l'architecture Intel Larrabee. Ces solutions devraient présenter l'avantage d'être moins coûteuses au niveau des communications que celles basées sur des cartes graphiques. Toutefois, ces unités de calcul restent encore fortement spécialisées pour les calculs en mode flottants plutôt qu'entiers comme requis par le chiffrement type AES. Le gain n'est donc pas garanti. Une alternative possible est d'exploiter des processeurs hautement parallèles spécialisés pour les traitements par flux sur des données entières (par exemple l'architecture "MPPA" de Kalray).

D'ailleurs des solutions mixtes, processeurs génériques couplés à un co–processeur spécialisé dans le chiffrement, sont présentes sur le marché : Cisco et Cavium proposent par exemple des solutions avec des cartes filles à intégrer dans un routeur pour effectuer le chiffrement. Ces solutions, qui ne sont pas purement logicielles, n'ont cependant pas les propriétés de sécurité d'un module en rupture. Notre solution exploite un module spécialisé pour le chiffrement et qui est en rupture, que nous résumons ci-après.

9.1.2 Sécurisation des passerelles de chiffrement par architecture en rupture

Pour atteindre un chiffrement à 10 Gb/s il est actuellement nécessaire de se reposer sur un module matériel d'accélération des opérations cryptographiques. De plus, il faut ajouter du matériel supplémentaire afin d'exécuter les traitements afférents aux protocoles réseau utilisés (dans le cadre du projet SHIVA ou des produits existants), ce qui crée des opportunités supplémentaires pour les attaquants.

Nous avons conçu une architecture dite en rupture, tirant avantage de ces besoins en intégrant un module de sécurité matériel (HSM) dans un environnement avec isolation rouge–noir. En effet, ce type d'architecture permet d'améliorer la sécurité par conception à condition de valider chaque élément. Toutefois, cette architecture impose de répartir les traitements des différentes couches protocolaires sur plusieurs entités physiques et de les optimiser pour atteindre l'objectif de 10 Gb/s.

Spécialisation des traitements réseau

Sur cette nouvelle architecture, les traitements protocolaires (dans notre cas encapsulation–décapsulation Ethernet, IP, IPsec, ESP et chiffrement–déchiffrement AES) sont répartis

sur trois éléments matériels distincts en rupture : module réseau rouge, module matériel cryptographique et module réseau noir. Il faut donc trois unités de traitement spécialisées.

L'utilisation d'un routeur logiciel configurable permet de restreindre les traitements protocolaires effectués au minimum requis par l'architecture : le flux de traitement de chaque paquet est spécifié, en fonction de son type, et un paquet qui sortirait de ces flux sera immédiatement détruit. La spécialisation limite donc la surface pour les attaquants tout en facilitant la preuve de bon fonctionnement. De plus, les solutions logicielles présentent l'avantage de pouvoir suivre les évolutions matérielles et de permettre la mise en place des dernières évolutions des algorithmes (de parallélisation par exemple).

Notre solution est basée sur le routeur logiciel Click. Un inconvénient de ce choix est l'impossibilité d'analyser, par des outils de preuve, la sécurité d'un tel logiciel à cause de sa taille. Une perspective, pour avoir une preuve formelle, serait de développer les piles réseau nécessaires sans logiciel non–prouvé.

9.1.3 Parallélisation des traitements réseau

L'approche historique de traitement des paquets réseau par interruptions n'est plus viable à très haut débit (à moins de recouvrir à des optimisations telles que les trames Ethernet "jumbo" par exemple, qui peuvent avoir une taille jusqu'à 64 Kio, rentabilisant alors une interruption. Ceci reste cependant une exception).

Il est donc nécessaire de tirer pleinement parti des cartes réseau multi-queues et du mode "polling" au niveau logiciel. Les cartes multi-queues permettent, grâce à leur tri préalable dans les queues en fonction d'un n-uplet basé sur les en-têtes des paquets, de faire une première séparation du trafic réseau qui respecte les flux : les paquets d'un même flux iront toujours dans la même queue. Il faut alors utiliser une politique parallélisée de traitement de ces paquets entrants, où chaque thread récupère dans une queue spécifique les paquets à traiter, via une technique de "polling". Il est à noter qu'il existe une alternative, consistant à utiliser les affinités des interruptions avec un cœur donné, mais à très haut débit elles consomment trop de temps sur les cœurs associés [102].

Parallélisme et équilibrage de charge par vol de paquets

L'architecture proposée fonctionne bien dans le cas d'une bonne répartition du débit sur les différents flux. Dans le cas extrême d'un unique flux, un seul processeur devrait faire tous les traitements. Une perspective est d'étendre notre étude de la parallélisation des traitements réseau et prenant en compte le problème de l'équilibrage de charge interne à un flux.

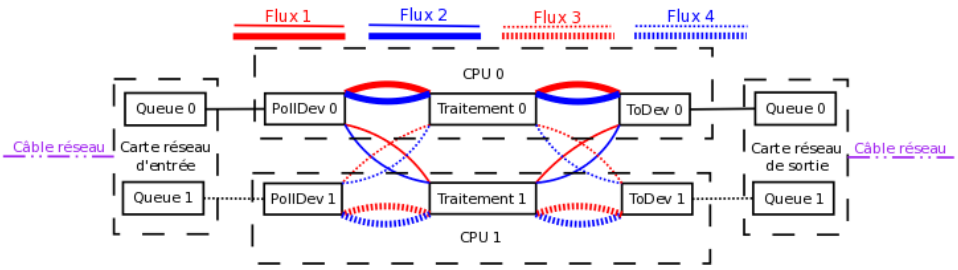


FIGURE 9.1 – Équilibrage des charges du traitement de deux flux réseau.

L'idée générale est de continuer à tirer parti des multiples unités de calcul lorsqu'il y a un nombre limité de flux réseau. Par exemple, dans le cas de la Figure 9.1, quatre flux réseau arrivent sur une passerelle et sont équitablement répartis dans deux queues. S'ils se partagent également le débit réseau, chacun des processeurs peut alors traiter de bout en bout deux flux sur la passerelle (cas des quatre flux en gras). Toutefois, si les flux ont des débits largement différents, et si un des processeurs n'est plus utilisé par ses flux, l'idée est alors de "voler" des paquets (flux "fins") à traiter sur la queue de l'autre processeur et lui rendre après les traitements. Il se pose alors le problème de réordonnancer après leur traitement les paquets dans les flux qui ont fait l'objet de vols.

9.2 Interactions entre ICMP et IPsec

ICMP est un des protocoles essentiels de la suite IP car il permet de transmettre les informations de contrôle sur l'état du réseau. Nous avons démontré qu'il est primordial de considérer correctement ICMP pour les aspects performance et sécurité des passerelles, alors qu'il est habituellement négligé (ainsi il n'est pas rare de figer le PMTU sur une passerelle IPsec).

9.2.1 ICMP nécessaire pour des communications réseau performantes ?

Afin d'utiliser la taille de paquet optimale, IP et IPsec utilisent intensément ICMP à travers l'algorithme de découverte du PMTU. Toutefois, certaines parties des réseaux IP filtrent les paquets ICMP, créant des "trous noirs" ICMP dommageables pour les performances des communications.

Nous avons donc développé une méthodologie d'analyse et un logiciel, IBTrack, dont le but est de préciser le comportement, par rapport à ICMP, des routeurs et chemins entre un utilisateur et une destination donnée. Pour ce faire, nous avons proposé une modélisation du réseau adaptée à notre problématique ainsi que des algorithmes pour caractériser le comportement des routeurs et affiner dans un second temps cette catégorisation.

Une étude à large échelle sur Internet a montré qu'IBTrack était capable de préciser la nature de 10 % des trous noirs ICMP pour un utilisateur de TCP sur Internet. Cela semble peu, mais il faut souligner que la méthodologie proposée ne repose que sur une observation à la source, sans aucune modification du cœur de réseau. Fort de cet outil, un administrateur peut plus facilement identifier la nature précise des problèmes liés à ICMP, voir les corriger.

PLPMTUd plutôt que PMTUd ?

Pour le cas de la découverte du PMTU, une alternative à PMTUd, indépendante d'ICMP, existe : PLPMTUd. Basé sur un système de sondage du chemin (alors que PMTUd attend que les routeurs remontent les erreurs), PLPMTUd nécessite un protocole de paquetisation pour fonctionner (le premier candidat étant TCP naturellement). Il envoie une sonde et si la destination valide sa réception, il juge la taille de la sonde utilisable.

Toutefois, la nature de l'approche par sondage de PLPMTUd le rend intrinsèquement moins réactif aux changements de chemin et plus coûteux en nombre de paquets et débit utilisé que PMTUd.

Il faut donc un compromis entre : d'une part l'utilisation d'ICMP qui permet de ne pas surcharger le réseau et d'avoir un retour rapide d'information sur l'état du réseau, mais qui se heurte aux problèmes du filtrage par certains routeurs ; et d'autre part des algorithmes indépendants d'ICMP, et donc non affectés par ces filtrages, mais aux performances (débit et latence) moindres.

Une perspective est l'étude des compromis possibles entre ces deux solutions.

9.2.2 Vulnérabilité des passerelles IPsec à ICMP

Nous avons aussi analysé les vulnérabilités liées à ICMP sur la sécurité d'une passerelle de chiffrement IPsec. Un problème fondamental sur le minimum de la taille maximale d'un paquet en présence de tunnels (IPsec dans notre cas) a été identifié. Il permet la mise en place d'une attaque de déni de service sur des passerelles IPsec en utilisant des messages ICMP PTB.

L'analyse des détails de l'attaque a permis de trouver des contre-mesures partielles : une configuration bridant la taille maximale des paquets IP plus que demandé par la norme, ou l'utilisation d'un mécanisme de découverte du PMTU noir similaire à PLPMTUd dans IPsec.

Le principe est de faire effectuer à l'implantation IPsec un sondage du PMTU du réseau d'accès (noir) sans tenir compte des messages ICMP, de manière similaire à PLPMTUd, afin de s'affranchir de la remise à zéro du PMTU stocké, et donc de l'attaque. Une perspective est de développer cet algorithme qui pourra ensuite être soumis à l'IETF pour faire avancer la norme IPsec.

Plus généralement, nous avons découvert que cette attaque est possible grâce à un défaut du traitement d'ICMP dans IPsec. Elle repose en fait sur un blocage fondamental présent dans les normes IP : les valeurs minimales des tailles maximales de paquets, que doit supporter tout lien physique, ne tiennent pas compte de la présence éventuelle de tunnels qui, du fait de l'augmentation des tailles de paquets (nouveau en-tête ajouté), conduit le paquet à excéder cette taille maximale. Ceci, couplé à l'impossibilité de redécouper un paquet chiffré, mène à un déni de service.

Il faut donc trouver comment gérer les surcoûts liés aux tunnels lorsque l'on approche de la limite inférieure des tailles maximales IP. Cette discussion est une opportunité d'apporter une contribution à l'IETF.

Annexe A

Attaques et contre-mesures physiques

Sommaire

A.1 État de l'art des principales attaques sur du matériel cryptographique I

A.1.1 Attaques physiques I

A.1.2 Attaques logiques III

A.2 Considérations matérielles pour l'implantation du HSM IV

A.2.1 Gestion des clés et sessions IV

A.2.2 Contrôle et gestion d'une passerelle V

A.2.3 Initialisation cryptographique V

A.2.4 Rotation des aires de stockage V

Cet annexe présente les différentes attaques matérielles considérées lors du projet SHIVA ainsi que les solutions envisagées.

A.1 État de l'art des principales attaques sur du matériel cryptographique

A.1.1 Attaques physiques

De plus en plus d'opérations cryptographiques étant déléguées désormais à du matériel dédié, les appareils cryptographiques deviennent les cibles des attaques physiques. Il y a deux grandes familles d'attaques, celles visant à rendre l'appareil inutilisable - dites déni de service - et celles dont le but est d'extraire du matériel des éléments qui permettront ensuite de déchiffrer les communications par exemple. Ces attaques peuvent être classées en fonction du niveau d'intrusion qu'elles nécessitent, ce qui impactera directement les contre-mesures à mettre en place (les attaques peu intrusives étant difficiles à détecter).

– Les attaques par canal auxiliaire passives ; ces attaques reposent sur l'étude de l'appareil en fonctionnement. Pour gêner ces attaques il faut mettre en œuvre des méthodes qui vont rendre les variations du fonctionnement matériel indépendantes des données utilisées.

– La violation physique au contraire est la plus intrusive des méthodes puisque ce type d'attaque implique des modifications du matériel afin soit d'en extraire des données soit d'altérer son fonctionnement.

– L'injection de faute est un mélange des deux types d'attaques précédentes dans lequel l'attaquant injecte des données dans l'appareil cryptographique et étudie parallèlement les modifications de comportement ; la corrélation des deux peut permettre l'extraction d'informations.

– Des modifications de l'environnement de l'appareil peuvent permettre du déni de service (température trop haute pour permettre un fonctionnement correct) ou des récupérations d'information - il a été montré récemment que de basses températures peuvent permettre un allongement de la durée de rémanence dans une mémoire [57].

Voici des exemples plus précis d'attaques par canal auxiliaire et par violation physique.

A.1.1.1 Attaques par canal auxiliaire

La majorité des attaques par canal auxiliaire implique une observation des différentes caractéristiques de l'appareil qui sont mises en relation avec les opérations effectuées.

– La consommation d'énergie est souvent corrélée au besoin en énergie nécessaire pour changer l'état d'un bit matériel. Une première catégorie d'attaque par canal auxiliaire consiste en l'étude de la consommation d'un appareil pendant son fonctionnement [105] ; puis un rapprochement des différentes empreintes énergétiques avec les clés potentiellement utilisées est fait. En effet, pour un même algorithme la consommation ne sera pas identique en fonction de la clé de chiffrement utilisée.

– De façon très similaire à l'attaque précédente, un autre moyen d'attaquer du matériel cryptographique utilise les émissions électromagnétiques : les variations d'émissions en fonction des paramètres peut mener à des informations sur les clés [90].
Mais il est aussi possible d'utiliser ce type de mesures pour cartographier le matériel (généralement combiné avec une autre attaque pour récupérer les clés en mémoire par exemple). Les dernières évolutions de ces attaques les ont placées dans la catégorie des attaques par canal auxiliaire depuis qu'un attaquant n'a plus besoin d'accéder directement à la carte mais uniquement à proximité.

– Les attaques temporelles se basent sur l'observation du temps nécessaire au matériel pour exécuter certaines actions (génération de clé, signature ou chiffrement d'un paquet, etc). Ce type d'attaque est particulièrement utilisé couplé avec des injections de fautes [97] (car l'attaquant accélère alors son attaque en diminuant la quantité de donnée nécessaire).

Toutefois si l'attaquant a accès au matériel cryptographique, alors les attaques par violation physique lui permettent des méthodes beaucoup plus directes et efficaces.

A.1.1.2 Attaques par violation physique

Bien que plus puissantes, cette gamme d'attaque est beaucoup plus difficile à exécuter. Toutefois le projet SHIVA visant une utilisation gouvernementale, de telles attaques font parties de l'objectif de sécurité.

- L'écoute des canaux de communication internes et externes peut révéler des transferts d'informations critiques sur l'appareil sur les bus internes par exemple.
- L'extraction ou la modification de données directement en mémoire. Cette attaque peut-être associée à une modification d'environnement (refroidissement pour allonger la rémanence) ou de l'injection de code afin de modifier le code exécuté par exemple.
- L'injection de fautes dans les données traitées, en utilisant des lasers [40] ou par modification des ressources externes (l'alimentation par exemple) peut amener à deux types d'attaques ; soit essayer de mettre l'appareil dans un état instable et/ou non prévu, soit en corrélation avec l'observation du comportement à la récupération d'information.
- Des modifications au niveau du circuit logique implanté sur le matériel peut amener à une modification du flux des données ou du déroulement de l'exécution.

Mais les appareils cryptographiques peuvent aussi être vulnérables à travers les logiciels qu'ils exécutent ou dans leur gestion des protocoles utilisés.

A.1.2 Attaques logiques

Ces attaques sont dites "logiques" dans la mesure où elles ne requièrent pas nécessairement un accès physique au matériel.

- La première catégorie à prendre en compte est celle des logiciels malveillants tels que les virus, chevaux de Troie et de manière plus générale les logiciels utilisant des vulnérabilités d'implantation pour prendre le contrôle du système.
- L'utilisation de données fausses ou mal–formées peut révéler des erreurs dans le traitement des protocoles donnant ainsi accès à des privilèges usurpés ou provoquant un état instable voir inconnu d'un sous-système.
- Certaines attaques vont chercher à rendre un appareil ou une information présente indisponible afin de procéder à un déni de service pour les utilisateurs. On peut citer par exemple l'épuisement des ressources de calcul afin de créer une famine.

Les attaques dites logiques sont donc difficilement caractérisables car elles visent souvent un système particulier ; elles requièrent donc une attention particulière aux vecteurs d'attaque lors de la conception.

Étant donné cet inventaire (non exhaustif) des attaques à prendre en considération lors de la conception de la passerelle SHIVA, la prochaine section présente les solutions possibles pour la conception du HSM de l'architecture en rupture.

A.2 Considérations matérielles pour l'implantation du HSM

Devant considérer les attaques physiques précédentes, le choix de la réalisation matérielle n'est pas trivial.

A.2.0.1 Les circuits intégrés spécifiques à une application (ASIC)

Ils ont été historiquement développés par les nations souhaitant protéger leurs algorithmes cryptographiques. Toutefois les circuits sont maintenant aisément extractibles afin d'en induire la conception et donc potentiellement les algorithmes. De plus l'utilisateur doit prendre en compte sa dépendance envers les constructeurs et concepteurs du circuit lors de l'évaluation de la sécurité du circuit [47].

A.2.0.2 Les circuits logiques programmables (FPGA)

Les FPGA offrent des caractéristiques natives pour protéger la conception du circuit. En effet la mémoire étant volatile, un arrêt "efface" le circuit du matériel. Cela limite l'effort de sécurisation à la zone mémoire contenant la configuration du FPGA ; effort qui peut être redondant avec la sécurisation des zones contenant les clés. De plus les contraintes de personnalisation de l'algorithme de chiffrement peuvent être adressées par une configuration par l'utilisateur de l'algorithme plutôt que par le constructeur ou concepteur.

Les partenaires ont donc choisi de développer le HSM à base de FPGA, choix qui permet de combiner le potentiel de sécurisation de l'implantation du système avec sa modularité par le client.

En plus de la partie cryptographique au cœur de l'HSM, le module devra aussi gérer la surveillance et la gestion en fonction des différents flux qui y seront traités.

A.2.1 Gestion des clés et sessions

Il faut bien séparer la gestion de l'échange des clés (et l'état des sessions) de la gestion du stockage sur le HSM des clés.

A.2.1.1 Protocole d'échange de clés

L'utilisation d'un FPGA comme base du module de sécurité permet si besoin est d'implanter dessus un processeur afin d'exécuter des logiciels génériques. Une gestion de cette famille de protocole via une implantation purement matériel présente le désavantage de nécessiter une "grande surface" sur le circuit. Au contraire, une implantation logicielle est beaucoup plus facile à mettre en œuvre mais doit se prémunir des attaques logiques sur l'implantation.

A.2.1.2 Stockage des clés de chiffrement

C'est typiquement un problème qui doit être adressé au niveau matériel car une solution de stockage logicielle (chiffrement logiciel puis stockage en mémoire par exemple) serait plus vulnérable. L'HSM SHIVA devra donc mettre en œuvre des mécanismes de stockage sécurisé.

A.2.2 Contrôle et gestion d'une passerelle

La surveillance d'un système et sa gestion sont des fonctionnalités plus facilement développables sur des processeurs génériques et en utilisant des langages de programmation de "haut niveau" (comparé à de l'assembleur par exemple). Et cela présente l'avantage de séparer les flux associés des flux cryptographiques éliminant les interférences au niveau de sous-systèmes qui devraient traiter les deux types de flux.

L'architecture retenue pour gérer le protocole d'échange de clé, leur stockage et les aspects gestion/contrôle sera mixte ; le traitement des aspects protocolaires et gestion/contrôle se fera de manière logicielle alors que le stockage de clés sera matériel. Toutefois la partie logicielle ne devra jamais travailler sur une clé non chiffrée. De plus, elle devra être exécutée sur un processeur implanté sur le module de sécurité.

Nous allons maintenant voir les différentes techniques qui seront utilisées lors du développement du module de sécurité de l'architecture en rupture en plus des protections standard (protection des bus, techniques de réduction des attaques par étude de la consommation de ressources, supervision constante, etc) qui seront bien évidement mises en place sur le module de sécurité.

A.2.3 Initialisation cryptographique

La phase d'initialisation du moteur cryptographique d'un appareil est l'une des phases les plus critiques ; en effet des attaques de type "écoute clandestine" peuvent compromettre l'architecture du circuit et des clés stockées. La solution envisagée repose sur l'implantation d'une initialisation par étapes où chaque étape vérifie l'intégrité, l'authenticité et le niveau d'autorisation avant de passer à l'étape suivante.

Un procédé à trois étapes, illustré sur la Figure A.1 permet d'envisager un niveau de sécurité suffisant :

1. La première étape (après la mise sous tension) charge la configuration basique du moteur de sécurité. Cela permet alors une vérification basique du matériel ainsi que l'installation de l'interface pour l'entrée des paramètres de sécurité nécessaires au passage à la seconde phase.

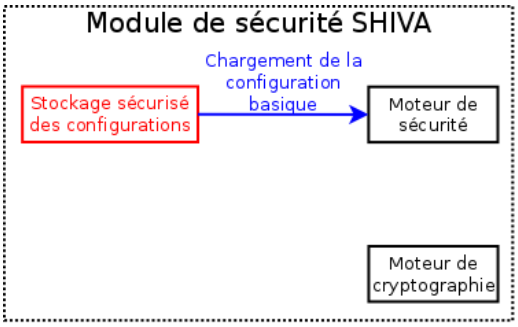
2. La seconde étape doit permettre de charger la configuration sécurisée du moteur de sécurité ainsi que l'initialisation basique du moteur cryptographique.

3. Finalement, le moteur de sécurité charge et installe la version sécurisée du moteur cryptographique.

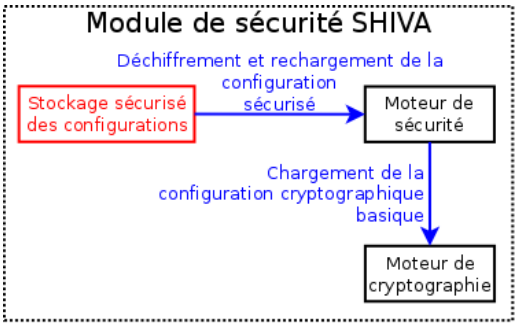
Cette initialisation en trois phases devra vérifier que le module est sécurisé avant de procéder au chargement d'information de plus haut niveau de confidentialité. Si une étape échoue, on ne doit pas procéder ni au déchiffrement ni au traitement d'information des phases suivantes. De plus en cas d'erreur, le module devra passer dans un état bloqué qui enclenchera l'effacement des informations critiques afin de rendre le module inutilisable.

A.2.4 Rotation des aires de stockage

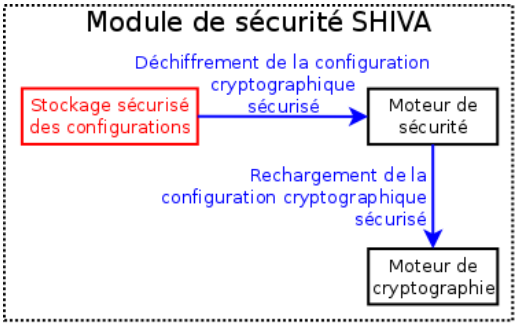
La rémanence des données est un risque largement connu qui peut alors compromettre le stockage de clés de chiffrement et autres informations critiques, particulièrement sur le



(a) 1ère étape : initialisation du moteur de sécurité



(b) 2ème étape : sécurisation du moteur de sécurité, initialisation du moteur cryptographique



(c) 3ème étape : sécurisation du moteur cryptographique

FIGURE A.1 – Initialisation en trois phases du module matériel

matériel "grand public" (mémoire, stockage type flash, etc). En effet, la rémanence peut se produire après quelques heures de stockage d'une information via électromigration [56] (déplacement d'atomes métalliques dans un support soumis à une très forte densité de courant) ou contamination ionique [58] (dégradation d'une puce de silicium par un ion extérieur, provenant du circuit intégré par exemple).

C'est pour ces raisons que le module matériel SHIVA va implanter une gestion des zones mémoire avec rotation des informations (les plus sensibles) qui déplacera physiquement sur le circuit les zones de stockage (mémoire et registres). Ce mécanisme sert à prévenir les attaques a priori sur les zones qui pourraient contenir des informations critiques.

Bibliographie

[1] Atelier B. http://www.atelierb.eu/.

[2] BrookGPU. http://graphics.stanford.edu/projects/brookgpu/.

[3] CEA Leti. http://www-leti.cea.fr/en.

[4] Certification Critères Communs. http://www.ssi.gouv.fr/fr/certification-qualification/cc/.

[5] Cisco 7600 Series/Catalyst 6500 Series Services SPA Carrier-400. http://www.cisco.com/en/US/prod/collateral/routers/ps368/product_data_sheet0900aecd8027c9ee_ps8768_Products_Data_Sheet.html.

[6] CS. http://www.c-s.fr/.

[7] CUDA Parallel Computing Platform. http://www.nvidia.com/object/cuda_home_new.html.

[8] Debian, the universal operating system. http://www.debian.org/.

[9] Direction générale de la compétitivité de l'industrie et des services. http://www.dgcis.redressement-productif.gouv.fr/.

[10] EASii-IC. http://www.easii-ic.com/.

[11] Ethernet working group. http://www.ieee802.org/3/.

[12] Improving Network Performance in Multi-Core Systems. http://www.intel.com/content/dam/doc/white-paper/improving-network-performance-in-multi-core-systems-paper.pdf.

[13] Inria - Inventeurs du monde numérique. http://www.inria.fr/.

[14] Institut Fourier. http://www-fourier.ujf-grenoble.fr/.

[15] Intel Many Integrated Core Architecture. http://www.intel.com/content/www/us/en/architecture-and-technology/many-integrated-core/intel-many-integrated-core-architecture.html.

[16] Intel Nehalem Architecture. http://www.intel.com/technology/architecture-silicon/next-gen/.

[17] iWall. http://www.iwall.fr/.

[18] Laboratoire Jean Kuntzmann : Mathématiques appliquées - Informatique. http://www-ljk.imag.fr/.

[19] Laboratoire TIMA - Techniques de l'Informatique et de la microélectronique pour l'Architecture des systèmes intégrés. http://tima.imag.fr/.

[20] LIG - Laboratoire d'Informatique de Grenoble. http://www.liglab.fr/.

[21] Linux. http://kernel.org/.

[22] MINALOGIC : Micro nanotechnologies et Logiciel. http://www.minalogic.com.

[23] MITRE corporation. http://www.mitre.org.

[24] Netheos : trust and privacy. http://www.netheos.net/.

[25] Planet-Lab. http://www.planet-lab.org.

[26] SHIVA : Secured Hardware Immune Versatile Architecture. http://shiva.minalogic.net.

[27] The Coq proof assitant. http://coq.inria.fr/.

[28] The Zettabyte Era. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/VNI_Hyperconnectivity_WP.html.

[29] Université Joseph Fourier. http://www.ujf-grenoble.fr/.

[30] Watch where you click : International cyber attacks on the rise. http://edition.cnn.com/2013/03/12/politics/international-cyber-crime.

[31] Smurf IP Denial–of–Service Attacks. http://www.cert.org/advisories/CA-1998-01.html, January 1998.

[32] Similar Attacks Using Various RPC Services. http://www.cert.org/incident_notes/IN-99-04.html, July 1999.

[33] Brice Augustin, Xavier Cuvellier, Benjamin Orgogozo, Fabien Viger, Timur Friedman, Matthieu Latapy, Clémence Magnien, and Renata Teixeira. Avoiding traceroute anomalies with Paris traceroute. In *Internet Measurement Conference*, 2006.

[34] Marcelo Bagnulo. RFC 6181 : Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses. http://datatracker.ietf.org/doc/rfc6181/, March 2011.

[35] Julien Bernard, Jean-Louis Roch, and Daouda Traore. Processor-oblivious parallel stream computations. In *16th Euromicro International Conference on Parallel, Distributed and network-based Processing*, Toulouse, France, Feb 2007.

[36] Alex Biryukov, Adi Shamir, and David Wagner. Real Time Cryptanalysis of A5/1 on a PC. In *Fast Software Encryption*, volume 1978 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2000.

[37] Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting mobile communications : the insecurity of 802.11. In *MOBICOM*, pages 180–189. ACM, 2001.

[38] Ian Buck, Tim Foley, Daniel Reiter Horn, Jeremy Sugerman, Kayvon Fatahalian, Mike Houston, and Pat Hanrahan. Brook for GPUs : stream computing on graphics hardware. *ACM Trans. Graph.*, 23(3) :777–786, 2004.

[39] Can Berk Guder. AES on CUDA. `http://github.com/cbguder/aes-on-cuda`, January 2009.

[40] Gaetan Canivet, Jessy Clédière, Jean Baptiste Ferron, Frédéric Valette, Marc Renaudin, and Régis Leveugle. Detailed Analyses of Single Laser Shot Effects in the Configuration of a Virtex-II FPGA. In *IOLTS*, pages 289–294, 2008.

[41] H.K. Jerry Chu and Vivek Kashyap. RFC 4391 : Transmission of IP over InfiniBand (IPoIB). `http://datatracker.ietf.org/doc/rfc4391/`, April 2006.

[42] Alex Conta, Stephen Deering, and Mukesh Gupta, Ed. RFC 4443 : Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. `http://datatracker.ietf.org/doc/rfc4443/`, March 2006.

[43] Joan Daemen and Vincent Rijmen. *The Design of Rijndael : AES - The Advanced Encryption Standard*. Springer Verlag, 2002.

[44] Daouda Traore and Jean-Louis Roch and Nicolas Maiilard and Thierry Gautier and Julien Bernard. Deque-free work-optimal parallel STL algorithms. In Springer-Verlag, editor, *EUROPAR 2008*, Las Palmas, Spain, August 2008.

[45] Mihai Dobrescu, Norbert Egi, Katerina Argyraki, Byung-Gon Chun, Kevin Fall, Gianluca Iannaccone, Allan Knies, Maziar Manesh, and Sylvia Ratnasamy. RouteBricks : Exploiting Parallelism to Scale Software Routers. In *Proceedings of the 22nd ACM Symposium on Operating Systems Principles (SOSP)*. ACM, 2009.

[46] Benoit Donnet, Matthew J. Luckie, Pascal Mérindol, and Jean-Jacques Pansiot. Revealing MPLS Tunnels Obscured from Traceroute. In *SIGCOMM Computer Communication Review*, volume 42, April 2012.

[47] Saar Drimer. Volatile FPGA design security – a survey (v0.96), April 2008.

[48] Morris Dworkin. Recommendation for Block Cipher Modes of Operation : Galois/Counter Mode (GCM) for Confidentiality and Authentication. `http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf`, 2007.

[49] Norber Egi, Adam Greenhalgh, Mark Handley, Gianluca Iannaccone, Maziar Manesh, Laurent Mathy, and Sylvia Ratnasamy. Improved Forwarding Architecture and Resource Management for Multi-Core Software Routers. In *Network and Parallel Computing, 2009. NPC '09. Sixth IFIP International Conference on*, pages 117–124, Oct. 2009.

[50] Jon Erickson. *Hacking : the art of exploitation, 2nd edition*. No Starch Press, San Francisco, CA, USA, second edition, 2008.

[51] Horst Feistel. Cryptography and Computer Privacy. *Scientific American 228*, pages 15–23, 1973.

[52] Flavio D. Garcia, Gerhard de Koning Gans, Ruben Muijrers, Peter van Rossum, Roel Verdult, Ronny Wichers Schreur, and Bart Jacobs. Dismantling MIFARE Classic. In *ESORICS*, volume 5283 of *Lecture Notes in Computer Science*, pages 97–114. Springer, 2008.

[53] Fernando Gont. RFC 5927 : ICMP Attacks against TCP. `http://datatracker.ietf.org/doc/rfc5927/`, July 2010.

[54] Fernando Gont, Guillermo Gont, and Carlos Pignataro. Recommendations for filtering ICMP messages. `http://datatracker.ietf.org/doc/draft-ietf-opsec-icmp-filtering/`, July 2013.

[55] Fernando Gont and Carlos Pignataro. RFC 6918 : Formally Deprecating Some ICMPv4 Message Types. `http://datatracker.ietf.org/doc/rfc6918/`, April 2013.

[56] Peter Gutmann. Data Remanence in Semiconductor Devices. In *USENIX Security Symposium*, 2001.

[57] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember : cold-boot attacks on encryption keys. *Commun. ACM*, 52(5) :91–98, 2009.

[58] Robert L. Hance, K. Erington, and Mark A. Chonko. Mobile ion contamination in CMOS circuits : a clear and present danger. In *Integrated Reliability Workshop, 1994. Final Report., 1994 International*, pages 3–, 1994.

[59] Omar Hasan, Jean-Marc Pierson, and Lionel Brunie. Access Control in Ubiquitous Environments Based on Subjectivity Eliminated Trust Propagation. In *EUC (2)*, pages 603–609. IEEE Computer Society, 2008.

[60] Peng He, Hongtao Guan, Gaogang Xie, and Kavé Salamatian. Evaluating and Optimizing IP Lookup on Many Core Processors. In *ICCCN*, pages 1–7. IEEE, 2012.

[61] Russell Housley. RFC 3686 : Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP). `http://datatracker.ietf.org/doc/rfc3686/`, January 2004.

[62] Young Hyun, Bradley Huffaker, Dan Andersen, Emile Aben, Colleen Shannon, Matthew Luckie, and KC Claffy. The CAIDA IPv4 Routed /24 Topology Dataset. `http://www.caida.org/data/active/ipv4_routed_24_topology_dataset.xml`, November 2011.

[63] Ludovic Jacquin. État de l'art sur les serveurs réseaux 10 Gbits/sec. Technical report, Inria, 2010.

[64] Ludovic Jacquin. Spécification des flux. Technical report, Inria, 2010.

[65] Ludovic Jacquin, Vincent Roca, Mohamed Ali Kaafar, Fabrice Schuler, and Jean-Louis Roch. IBTrack : an ICMP black holes tracker. In *Global Telecommunications Conference (GLOBECOM), 2012 IEEE*, pages 2827–2833, 2012.

[66] Ludovic Jacquin, Vincent Roca, Jean-Louis Roch, and Mohamed Al Ali. Parallel arithmetic encryption for high-bandwidth communications on multicore/GPGPU platforms. In *Proceedings of the 4th International Workshop on Parallel and Symbolic Computation*, PASCO '10, pages 73–79, New York, NY, USA, 2010. ACM.

[67] Ludovic Jacquin and Fabrice Schuler. Implantation sur plate-forme PC standard du traitement des flux avec chiffrement simulé sur l'émulateur logiciel restreint du module SHIVA. Technical report, Inria, 2011.

[68] Ethan Katz-Bassett, Harsha V. Madhyastha, Vijay Kumar Adhikari, Colin Scott, Justine Sherry, Peter van Wesep, Thomas E. Anderson, and Arvind Krishnamurthy. Reverse traceroute. In *NSDI*, pages 219–234, 2010.

[69] Ethan Katz-Bassett, Harsha V. Madhyastha, John P. John, Arvind Krishnamurthy, David Wetherall, and Thomas Anderson. Studying black holes in the internet with Hubble. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'08, pages 247–262, Berkeley, CA, USA, 2008. USENIX Association.

[70] Charlie Kaufman, Paul Hoffman, Yoav Nir, and Pasi Eronen. RFC 5996 : Internet Key Exchange Protocol Version 2 (IKEv2). `http://datatracker.ietf.org/doc/rfc5996/`, September 2010.

[71] Stephen Kent. RFC 4302 : IP Authentication Header. `http://datatracker.ietf.org/doc/rfc4302/`, December 2005.

[72] Stephen Kent. RFC 4303 : IP Encapsulating Security Payload (ESP). `http://datatracker.ietf.org/doc/rfc4303/`, December 2005.

[73] Stephen Kent and Karen Seo. RFC 4301 : Security Architecture for the Internet Protocol. `http://datatracker.ietf.org/doc/rfc4301/`, December 2005.

[74] Eddie Kohler. *The click modular router*. PhD thesis, 2001.

[75] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The click modular router. *ACM Trans. Comput. Syst.*, 18(3) :263–297, August 2000.

[76] Cédric Lauradoux. Throughput/code size tradeoff for stream ciphers. In *Proceedings of SASC 2007 - ECRYPT Workshop on stream ciphers*, Bochum, Allemagne, january 2007.

[77] Victor W. Lee, Changkyu Kim, Jatin Chhugani, Michael Deisher, Daehyun Kim, Anthon D. Nguyen, Nadathur Satish, Mikhail Smelyanskiy, Srinivas Chennupaty, Per Hammarlund, Ronak Singhal, and Pradeep Dubey. Debunking the 100X GPU vs. CPU myth : an evaluation of throughput computing on CPU and GPU. In *Proceedings of the 37th annual international symposium on Computer architecture*, ISCA '10, pages 451–460, New York, NY, USA, 2010. ACM.

[78] Matthew Luckie, Kenjiro Cho, and Bill Owens. Inferring and debugging path MTU discovery failures. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, IMC '05, pages 17–17, Berkeley, CA, USA, 2005. USENIX Association.

[79] Matthew Luckie and Ben Stasiewicz. Measuring path MTU discovery behaviour. In *Proceedings of the 10th annual conference on Internet measurement*, IMC '10, pages 102–108, New York, NY, USA, 2010. ACM.

[80] Matthew J. Luckie. Scamper : a scalable and extensible packet prober for active measurement of the internet. In *Internet Measurement Conference*, pages 239–245, 2010.

[81] Layong Luo, Gaogang Xie, Yingke Xie, Laurent Mathy, and Kavé Salamatian. A hybrid IP lookup architecture with fast updates. In *INFOCOM*, pages 2435–2443. IEEE, 2012.

[82] Pietro Marchetta, Pascal Mérindol, Benoit Donnet, Antonio Pescapè, and Jean-Jacques Pansiot. Topology Discovery at the Router Level : A New Hybrid Tool Targeting ISP Networks. *IEEE Journal on Selected Areas in Communications*, 29(9) :1776–1787, 2011.

[83] Edoardo D. Mastrovito. *VLSI Architectures for Computation in Galois Fields.* Linköping studies in science and technolog, Linköping University, Sweden, 1991.

[84] Matt Mathis and John W. Heffner. RFC 4821 : Packetization Layer Path MTU Discovery. `http://datatracker.ietf.org/doc/rfc4821/`, March 2007.

[85] John Michael McConnell. NSTISSAM TEMPEST/2-95. `http://web.archive.org/web/20070408221244/http://cryptome.org/tempest-2-95.htm`, December 1995.

[86] Christopher B. McCubbin, Ali Aydin Selçuk, and Deepinder P. Sidhu. Initialization Vector Attacks on the IPsec Protocol Suite. In *9th IEEE International Workshops on Enabling Technologies : Infrastructure for Collaborative Enterprises (WETICE 2000)*, pages 171–175. IEEE Computer Society, 2000.

[87] Pascal Mérindol, Benoit Donnet, Jean-Jacques Pansiot, Matthew J. Luckie, and Young Hyun. MERLIN : MEasure the Router Level of the INternet. In *7th Conference on Next Generation Internet, EURO-NGI, Kaiserslautern, Germany, June 27-29, 2011*, pages 1–8. IEEE, 2011.

[88] Jeffrey Mogul and Steve Deering. RFC 1191 : Path MTU Discovery. `http://datatracker.ietf.org/doc/rfc1191/`, November 1990.

[89] Gordon E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8) :114–117, April 1965.

[90] Elke De Mulder, Siddika Berna Örs, Bart Preneel, and Ingrid Verbauwhede. Differential power and electromagnetic attacks on a FPGA implementation of elliptic curve cryptosystems. *Computers & Electrical Engineering*, 33(5-6) :367–382, 2007.

[91] Fiona Fui-Hoon Nah. A study on tolerable waiting time : how long are Web users willing to wait ? *Behaviour & IT*, 23(3) :153–163, 2004.

[92] Jakob Nielsen. Nielsen's Law of Internet Bandwidth. `http://www.nngroup.com/articles/law-of-bandwidth/`, April 1998.

[93] Ventzislav Nikov. A DoS Attack Against the Integrity-Less ESP (IPSec). *IACR Cryptology ePrint Archive*, 2006 :370, 2006.

[94] Stephen Northcutt and Judy Novak. *Network Intrusion Detection, Third Edition*. New Riders Publishing, September 2002.

[95] Kenneth G. Paterson. A Cryptographic Tour of the IPsec Standards. *IACR Cryptology ePrint Archive*, 2006 :97, 2006.

[96] Kenneth G. Paterson and Arnold K. L. Yau. Cryptography in Theory and Practice : The Case of Encryption in IPsec. In *Advances in Cryptology - EUROCRYPT 2006*, Lecture Notes in Computer Science 4004, pages 12–29, St. Petersburg, Russia, May 2006.

[97] Andrea Pellegrini, Valeria Bertacco, and Todd M. Austin. Fault-based attack of RSA authentication. In *DATE*, pages 855–860, 2010.

[98] Jon Postel. RFC 0792 : Internet Control Message Protocol. `http://datatracker.ietf.org/doc/rfc792/`, September 1981.

[99] Jon Postel. RFC 0879 : The TCP Maximum Segment Size and Related Topics. `http://datatracker.ietf.org/doc/rfc879/`, November 1983.

[100] Rachid Saadi, Jean-Marc Pierson, and Lionel Brunie. T2D : a peer to peer trust management system based on disposition to trust. In *SAC*, pages 1472–1478. ACM, 2010.

[101] Jamal Hadi Salim, Robert Olsson, and Alexey Kuznetsov. Beyond softnet. In *Proceedings of the 5th annual Linux Showcase & Conference - Volume 5*, ALS '01, pages 18–18, Berkeley, CA, USA, 2001. USENIX Association.

[102] Jamal Hadi Salim, Robert Olsson, and Alexey Kuznetsov. Beyond softnet. In *Proceedings of the 5th annual Linux Showcase & Conference - Volume 5*, ALS '01, pages 165–172, Berkeley, CA, USA, 2001. USENIX Association.

[103] Claude E. Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 28 :657–715, 1949.

[104] Weidong Shi, Hsien-Hsin S. Lee, Mrinmoy Ghosh, Chenghuai Lu, and Alexandra Boldyreva. High Efficiency Counter Mode Security Architecture via Prediction and Precomputation. In *ISCA*, pages 14–24. IEEE Computer Society, 2005.

[105] François-Xavier Standaert, François Macé, Eric Peeters, and Jean-Jacques Quisquater. Updates on the Security of FPGAs Against Power Analysis Attacks. In *ARC*, pages 335–346, 2006.

[106] Pascale Vicat-Blanc/Primet, Vincent Roca, Johan Montagnat, Jean-Patrick Gelas, Olivier Mornard, Lionel Giraud, Guilherme Piegas Koslovski, and Tram Truong Huu. A Scalable Security Model for Enabling Dynamic Virtual Private Execution Infrastructures on the Internet. In *9th IEEE International Symposium on Cluster Computing and the Grid (CCGrid'09), Shanghai, China*, May 2009.

RÉSUMÉ

Dans cette thèse nous abordons le problème de la conception de passerelle IPsec très haut débit pour la sécurisation des communications entre réseaux locaux. Pour cela, nous proposons deux architectures : une passerelle purement logicielle sur un unique serveur, dite intégrée, et une passerelle utilisant plusieurs serveurs et un module matériel cryptographique, dite en rupture.

La première partie de nos travaux étudie l'aspect performance des deux architectures proposées. Nous commençons par montrer qu'un serveur générique est limité par sa puissance de calcul pour atteindre l'objectif de chiffrement et communication à 10 Gb/s. De plus, les nouvelles cartes graphiques, bien que prometteuses en terme de puissance, ne sont pas adaptées au problème du chiffrement de paquets réseau (à cause de leur faible taille). Nous mettons alors en place une pile réseau répartie sur plusieurs machines et procédons à sa parallélisation dans le cadre de l'architecture en rupture.

Dans un second temps, nous analysons l'intégration d'une passerelle dans un réseau, notamment l'interaction du protocole de contrôle ICMP avec IPsec. ICMP est particulièrement important pour atteindre le haut débit par son implication dans le mécanisme d'optimisation de la taille des paquets. Pour cela, nous avons développé IBTrack, un logiciel d'étude du comportement des routeurs, par rapport à ICMP, le long d'un chemin. Nous montrons ensuite qu'ICMP est un vecteur d'attaque contre IPsec en exploitant un défaut fondamental des normes IP et IPsec : le surcoût des paquets IP créé par le mode tunnel entre en conflit avec le minimum de la taille maximale prévue par IP.

TITRE

Compromis performance/sécurité des passerelles très haut débit pour Internet.

ABSTRACT

In this thesis, we explore the design of a high-bandwidth IPsec gateway to secure communications between local networks. We consider two gateway architectures: the first one, called "integrated gateway", is a purely software approach that uses a single server; the second one, called "split architecture", relies on a hardware security module and two standard servers.

The first contribution of this thesis consists in an evaluation of both architectures on the performance side. We show that a standard server lacks processing capacities to sustain 10 Gb/s networking and ciphering. Moreover, although new graphic card architectures seem promising, they are not appropriate to cipher network packets. Therefore we have designed and evaluated a prototype for the split architecture. Particularly, we show that the 10 Gb/s goal is hard to reach when using only the standards sizes and no software aggregation method, which creates jitter.

The second contribution of this thesis concerns the gateway integration inside a network, mainly at the ICMP/IPsec interaction level. Given the importance of ICMP in the Path Maximum Transmission Unit discovery (PMTUd), we developed IBTrack, a software which aims at characterizing router's behavior, with regards to their ICMP handling, along a path. Afterwards, we show that ICMP can be used as an attack vector against IPsec gateways by exploiting a fundamental flaw in the IP and IPsec standards: the IPsec tunnel mode overhead conflicts with the minimum maximal size of IP packets.