Università degli Studi di Milano

Facoltà di Scienze Matematiche, Fisiche e Naturali

Dipartimento di Tecnologie dell'Informazione

Corso di Dottorato di Ricerca in Scienze Informatiche
XIII ciclo

Tesi di Dottorato di Ricerca

A Design Methodology for HW/SW Security Protocols

Alberto Ferrante
Matr: R05112

Relatore: **Prof. Vincenzo Piuri**
Correlatori: **Prof. Luigi Dadda**, **Dr. Jeff Owen**

Anno accademico 2004-2005

Università degli Studi di Milano

Facoltà di Scienze Matematiche, Fisiche e Naturali

Dipartimento di Tecnologie dell'Informazione

Corso di Dottorato di Ricerca in Scienze Informatiche
XIII ciclo

Tesi di Dottorato di Ricerca

A Design Methodology for HW/SW Security Protocols

INF/01

Alberto Ferrante
Matr: R05112

Tutor:	**Prof. Vincenzo Piuri**
	....
Coordinatore dottorato:	**Prof. Gianni Degli Antoni**
	....

Correlatori: **Prof. Luigi Dadda**
ALaRI - Università della Svizzera italiana, Lugano, Svizzera
Dr. Jeff Owen
ST Microelectronics Inc., San Jose, California, USA

Anno accademico 2004-2005

. . . To my parents. . .

Contents

List of Figures

List of Tables

Acronyms

AES Advanced Encryption Standard.

AH Authentication header.

API Application Protocol Interface.

ASIC Application-Specific Integrated Circuit.

CBQ Class-based Weighted Fair Queuing.

DES Data Encryption Standard.

DH Diffie-Hellman.

Diffserv Differentiated Services.

DMA Direct Memory Access.

DoI Domain of Interpretation.

DoS Denial of Service.

ECC Elliptic Curve Cryptography.

ESP Encapsulating Security Payload.

HMAC Hash Message Authentication Code.

IETF Internet Engineering Task Force.

IKE Internet Key Exchange.

ISAKMP Internet Security Association and Key Management Protocol.

IP Internet Protocol; if not differently specified, the version 4 of this protocol.

IPv6 Internet Protocol version 6.

IPSec IP Secure.

IPComp IP Compression.

L2F Layer 2 Forwarding

L2TP Layer 2 Tunneling Protocol.

MD5 Message Digest algorithm 5.

MTU Maximum Transfer Unit.

NoC Network on Chip.

NP Network Processor.

PAD Peer Authentication Database.

PPP Point to Point Protocol.

PPTP Point to Point Tunnelling Protocol.

PQ Priority Queuing.

WFQ Flow-based Weighted Fair Queuing.

QoS Quality of Service.

RFC Request For Comments.

RTL Register Transfer Level.

SA Security Association.

SAD Security Association Database.

SHA Secure Hash Standard.

SoC System on Chip.

SPD Security Policy Database.

TCP	Transmission Control Protocol.
TLS	Transport Level Security.
UDP	User Datagram Protocol.
UML	Unified Modelling Language.
VPN	Virtual Private Network.

Acknowledgments

The first people I would like to acknowledge are my parents and all my family who have supported me during these last twentynine years.

This thesis would not even exist without the help of Vincenzo, who is my advisor: his precious suggestions helped me a lot during these three years of PhD... Thank you Vincenzo!

A special acknowledgment goes to the referees of this thesis, Prof. Miroslaw Malek, Prof. Eduardo Sanchez, and Prof. Renato Stefanelli.

I would also like to acknowledge Prof. Dadda, Jeff, Fabien, and Marco for their invaluable contribution to this thesis. Furthermore, Jeff is the one who came up with the idea of working on these topics, therefore he deserves a special thank.

All the people at the ALaRI institute, such as Prof. Sami, Umberto, and all the staff deserve to be mentioned for their support and understanding during the last years. I would also like to take the opportunity to acknowledge all the students I met at ALaRI: all of them taught me a lot; some of them also helped me during this research... Antonietta, Antonio, Rodrigo, Sathish, and Uljana, thank you!

I would also like to mention my friend Sara: her help with the English in some parts of this thesis has been greatly appreciated!

Last but not least, I would like to thank all my friends, who continuously supported me during my life.

Introduction

The ability to communicate has become of fundamental importance for every activity of the human life: companies need to connect different seats and to communicate with their customers and partners; human beings need to communicate with others, with companies, and institutions.

Security and privacy is also an important need of the modern world: the rising competitiveness among industries imposes an increasing level of protection for each Company's confidential information; private information of the human beings also need to be protected or, anyway, only pieces of information should be able to be revealed by each person to certain recipients.

This need for security is obviously in contrast with the need of communication. As a matter of fact, sending information over any communication mean could expose them to possible evesdroppers. The only way to solve this contrast is to introduce some security mechanisms in communications. Communication security is crucial for economic and social development. Many security mechanisms have been studied and deployed over the years. Presently used mechanisms are based over cryptography or, for few very advanced applications, quantum cryptography. The latter technology is the future for a limited number of applications as it requires peculiar technological conditions. Traditional cryptographic techniques will probably continue to be used for common applications for many years. Wether the possible advent of quantum computers will determine the end of traditional cryptographic techniques or not, need to be clarified. As a matter of fact, quantum computers will be able to factorize large prime numbers in very small times. This is potentially dangerous for some public-key cryptographic algorithms (e.g., RSA) but not for others. These algorithms can be substituted with others that are not based on prime number factorization (e.g., ECC) and the whole cryptosystem will be able to work without any problem. In any case experts believe that actual secured data communications – if properly configured – are usually the most secure part of the whole IT system. The less secure parts of each IT system are the ones involving humans. As a matter of fact one may have the most advanced and well configured security mechanisms, but human factor can play a fundamental role in revealing classified information. Companies' employees can reveal (by intention or not) important information without the real need to involve the IT infrastructure. Solving this problem is far more difficult than any other: technical problems can be solved by adopting new technologies, but the human problem can only be solved by educating people to security. This education process may take many years to give proper effects.

An area of communication security is the one of security protocols. These protocols offer a way to use cryptographic algorithms for providing communication security. As a matter of fact, cryptographic algorithms are not usable by themselves: they need mechanisms for exchanging keys between the parties that are involved and for managing the secure connections. This is what secure protocols exactly do.

Secure protocols are based on cryptographic algorithms and these algorithms are very resource consuming. Specialized hardware is therefore used to support high network performances as general purpose CPUs cannot often provide the necessary computational capacity. Gilder's [52] and at the Moore's [53] laws say that this situation is not going to improve with time: while Moore says that computational capacity is doubling every 18 months, Gilder says that available network bandwidth doubles every 12 months.

Goal of the work presented in this dissertation is to study a comprehensive design methodology for mixed hardware/software architectures dedicated to security protocols. The IPSec (IP Secure) protocol suite is taken as a reference for this work, as it has assumed great importance, being also included as mandatory security mechanism in the new version of the IP protocol, IPv6.

The work here reported can be subdivided into different phases:

1. Study of the application requirements (IPSec suite protocols and a study about virtual private networks); this is necessary to understand the problems that may arise by using these protocols.

2. Profiling of one of the current IPSec software implementation: this allows to understand in which cases hardware acceleration is really necessary and the performance requirements.

3. Study of the actual hardware/software architectures for secure protocols.

4. Study of advantages disadvantages of the actual architectures.

5. Development of an abstract model of IPSec; this model will be used as a reference during the design phase. Starting from the model a testing methodology for IPSec-based systems will be developed.

6. Optimization of relevant aspects of presently used IPSec implementations.

7. High-level design of an innovative System on Chip for efficiently processing IPSec traffic.

In the following chapters the previously discussed topics are presented.

In Chapter 1 an explanation of the main technologies involved in virtual private networking is given. Focus is mainly concentrated on the security protocols which are involved. The IPSec suite of protocols is mainly presented, as well as an introduction on cryptography.

Chapter 2 provides a performance analysis of the IPSec suite of protocols. Experimental results there reported show that IPSec is very resource consuming and hardware accelerators are crucial in reaching high performances. This is also confirmed by other works found in the literature. Performance considerations also help in configuring IPSec-based networks.

Chapter 3 provides an overview of the existing hardware accelerators on the market. A classification of security protocol dedicated accelerators is firstly given. A list of currently available accelerators and network processors is then provided. As most of them are commercial products only marketing information are often available.

Chapter 4 gives an overview of the current scenario of the security protocol implementations along with an evaluation of the future trends in the field.

Chapter 5 provides an innovative model of the IPSec suite of protocols by using UML. As IPSec is very complex, these specification can help understanding it and the relations between its different parts. The use of UML allows for abstracting the model from the implementation, thus providing a suitable base both for design, including HW/SW partitioning, and for testing. A testing methodology based on the UML model is presented at the end of this chapter.

Chapter 6 proposes some optimizations to the presently used HW/SW interfaces for IPSec accelerators. These optimizations are both useful for presently used systems and for developing new architectures for IPSec. As a matter of fact, optimizing the present systems also allows to understand their problems. The optimizations we propose in this chapter consist of some packet scheduling algorithms which allow using together multiple cryptographic accelerators and software implementations of the cryptographic algorithms. Reference system is composed by a normal PC architecture hosting a certain number of accelerators. Only cryptography-related operations are offloaded to them. This is a scheme used on present low-end servers. This scheme can anyway be replicated, by considering different processing and communication speeds, on different, more performant, architectures. An algorithm for extracting better performances when small packets are processed in such an architecture is also presented. All the algorithms presented in this chapter allow for optimizing very important parts of IPSec and for designing an enhanced, more flexible, architecture.

Chapter 7 presents a study over a high-performance comprehensive solution for IPSec. Different high level architectures are presented along with the main requirements. Main parts of these architectures are developed in detail.

Chapter 8 presents some proposals for designing two functional blocks of the IPSec SoC: the core part of the blocks implementing the IKE protocol and the database query functionality. These two functional blocks were chosen because there exist no implementation of them in the literature.

Chapters 5, 6, 7, and 8, along with Section 2.2, contain the original contribution of this work. Parts of these sections were published in [20, 18, 41, 19, 23].

To summarize, the original results shown in this dissertation are:

- a performance measurement of IPSec;
- an abstract model of IPSec and a testing methodology for IPSec-based systems;
- three packet scheduling algorithms for multi-accelerator based systems;
- the high-level architecture of an innovative SoC for IPSec;
- a SystemC model of IKE, allowing to perform optimal HW/SW partitioning of it;
- the internal high-level design of two functional blocks of the SoC.

1

Virtual Private Networks and Network Security

Since many years the need of internal Companies communication has increased. The ability to support mobile users and to connect seats located in different places have become of fundamental importance in everyday business. In this chapter we present some techniques presently used to support these functionalities and the main technologies behind them.

1.1 Virtual Private Networks

To support communications out of local networks, private separate networks were firstly used. This solution gives good performances and security, but it is very costly as it requires to rent private communication lines. As the Internet has become pervasive, the idea of Virtual Private Networks (VPNs) has become popular. The main idea of VPNs is to use secure protocols to build secure communication channels and to allow the machines connected to these channels to act as if they were connected to the same private networks. The main idea is therefore to virtually build a private network over a public one: VPNs use obfuscation through secure tunnels, rather than physical separation, to keep communications private [127, 42]. VPNs have become popular for many reasons:

- ubiquitous coverage: The Internet offers wider coverage compared with the private data network infrastructures. Adding new destinations to VPNs usually consists of modifying some configuration files; adding new destination to private networks usually consists of adding new circuits and possibly sign interconnection agreements between different providers,

- cost reduction: for VPNs there is no need to purchase and maintain special purpose infrastructures. General purpose internet connections are sufficient to allow VPN access,
- security: VPN use cryptography to provide data confidentiality and integrity. In private networks security usually relies only on the telecommunication service provider's physical security practices.

Main VPN scenarios are shown in Figure 1.1 and in Figure 1.2. The first figure shows a mobile user connected to a company's network through a VPN. This configuration, that is usually called *road warrior*, allows the mobile user to access the company's internal network theoretically as if he was connected to it from inside. Whether the access to company's network resources is limited or not, depends on the security policy that have been deployed in the specific network. Usually the mobile user's machine gets a virtual address belonging to the private company's network. The second figure shows two private networks connected together by means of a VPN. This is typical when two or more seats of the same company need to communicate and to share information. A mix of the previously two described scenarios can be deployed for providing different seats interconnection and access to mobile users. In each one of these schemes there is one fundamental network component that is the *secure gateway*. This machine manages the secure communications and it usually runs a firewall, a gateway, and a VPN server. The Firewall is for filtering the connections to the internal network. Depending on the policy that has been selected, non-secured connections can be refused or not. In any secure network non-secured connections should anyway have limited access to internal resources. The gateway is for routing the traffic, while the VPN server is responsible of managing the VPN connections.

VPNs are usually created by associating two different protocols, one for data security and one for emulating a point to point connection. Layer 2 tunnelling protocols are specifically designed to tunnel Point-to-Point Protocol (PPP) [122] frames through an IP network. PPP protocols are used to route privately addressed packets through a publicy addressed infrastructure. For the road warrior configuration, the remote uses sets up a PPP connection, tunnelled on IP, to the secure gateway. Once a PPP connection

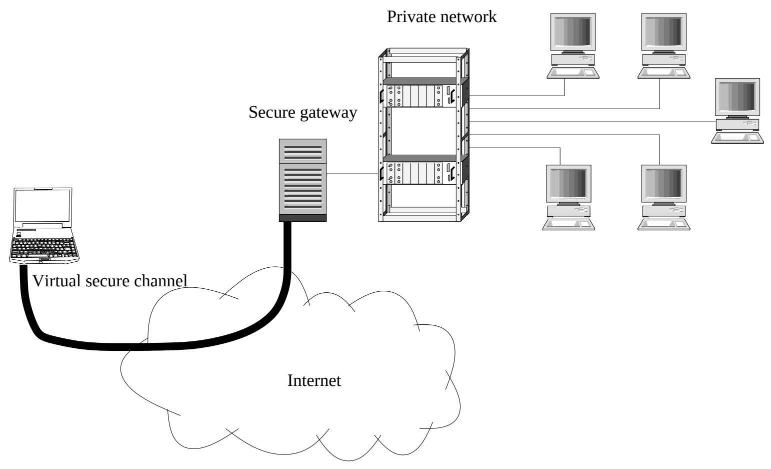

Figure 1.1: A mobile user connected to a private network through a VPN.

has been created, all the traffic that need to be routed to the internal network, is tunnelled over the PPP connection. For emulating the point to point connection, different protocols are available and are presently used. Some of them are L2TP [36, 123], PPTP [68], and L2F [15]. The last ones are proprietary protocols of Microsoft and Cisco, respectively. L2TP, that stands for Layer 2 Tunnel Protocol merges the characteristics of the other two.

The main secure protocol actually used is IPSec. As a matter of fact this protocol supports the creation of secure tunnels in a native way and was thought for being used on secure gateways. IPSec is the protocol on which this work will mostly concentrate, therefore a detailed description of this protocol is given in section 1.3.

1.2 Introduction to cryptography

There are two kinds of cryptographic algorithms, the symmetric and the public key ones. The former ones are faster and very secure, but need to have a pre-shared secret key. The latter are slower but not less secure (if the right key dimension is chosen) and do not need to have a pre-shared secret key. A brief description of the two algorithm classes and a presentation of the Diffie-Hellman protocol are discussed below.

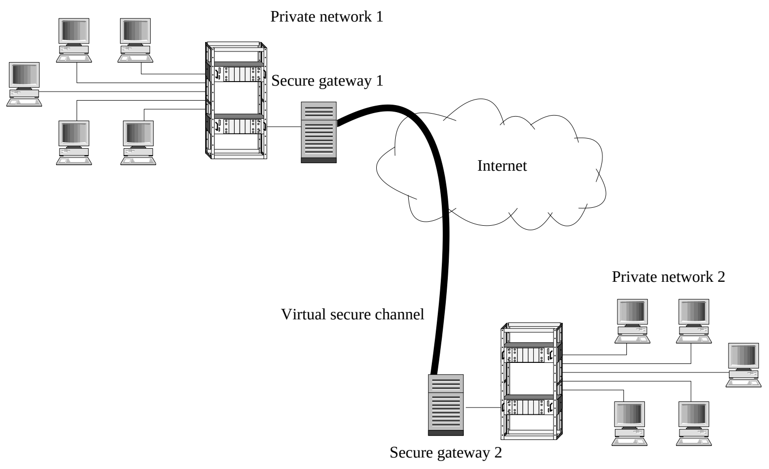

Figure 1.2: Two private networks connected together through a VPN.

1.2.1 The symmetric key algorithms

The algorithms in the "symmetric key" class work by using a pre-shared secret key; essentially, some transformations involving that key are applied to the data to be codified. Some different working patterns exist for these algorithms, based on the application for which they have been designed. For example there is the CBC mode that is suitable for applications in which big blocks of data need to be encrypted [87, Chapter 1].

In the last years the most widely used of this class of algorithms has been triple-DES (Digital Encryption Standard), a variation of the old (1977) DES. A new cryptographic algorithm called AES (Advanced Encryption Standard) has been selected from competing candidates by NSA, outdating the triple-DES. AES has now become the standard algorithm adopted by the NSA [37] and has been already deployed on many systems.

1.2.2 The public key algorithms

This kind of algorithms solve the problem of having a pre-shared key by using asymmetric cryptography techniques. Two keys for every peer are needed, one that is called "private" and that is known only by the owner, and another called "public" and known to everyone that wants to communicate with that recipient. Some transformations, based on the public key, are applied to every communication directed to that recipient. This makes the data inaccessible to everyone that does not have the private key. As a matter of fact the inverse transformations cannot be applied by only knowing the public key [87, Chapter 12].

Nowadays the most widely used algorithm of this class is RSA, but new algorithms, such as ECC (Elliptic Curve Cryptography), have been developed and probably will become the dominating ones very soon. As a matter of fact ECC has the advantage of requiring smaller keys for obtaining the same security level as RSA. As a matter of fact, secure keys for RSA are thousands of bits long, while secured keys for ECC are just hundreds of bits long.

1.2.3 The Diffie-Hellman protocol

Diffie-Hellman provides a solution to the key exchange problem by allowing two parties, never having met in advance or having shared keying material, to establish a shared key secret by exchanging messages over an open channel. The key is exchanged in the following way:

- the first peer (A) chooses a *random secret*, which is a sequence of random bits, called x; it does some operations on it and sends the result (h) to B;
- the second peer (B) chooses a random secret called y, does some operations on it and sends the result (k) to A;
- B receives h from A and computes the key using h and y;
- B receives k from B and computes the key using k and x.

The key protection is given by the fact that the operations performed on x and y to obtain h and k, need a long computational time to be inverted. Therefore possible third parties discovering the values of h and k should not be able to compute the key in a reasonable amount of time.

The operations that can be applied to x and y to obtain h and k can be based either on elliptic curves or on exponentials in the discrete fields. The former case is based on the same principles of ECC, while the other is based on the RSA approach.

See [87, Chapter 1] for more information about Diffie-Hellman and the key exchange procedures.

1.2.4 Authentication algorithms

These types of algorithms are used to certify that a specific piece of information comes from a certain peer and that it has not been modified by any third party. This can be done by computing a hash function of the data and to apply an encryption algorithm (that can be a public or a symmetric key one) [87, Chapter 1] to the result of this operation. A hash function is one that, when applied to some data, provides a different short code for every different packet of data, or, at least provides equal codes for different data with a really low probability.

1.3 Security Protocols

Many different protocols adding security to communication have been proposed so far. The main goal of these protocols is to use the previously explained cryptographic algorithms to provide some security in untrusted environments (e.g. on the Internet).

Most of these protocols are dedicated to specific applications, like, as in the case of SSL, to secure web transactions. There are also some general purpose protocols such as TLS and IPSec. These two protocols were designed to operate at different levels of the ISO/OSI stack, being this their main macroscopic difference. While TLS [118] operates at application level (it acts on TCP packets), IPSec operates at network level (it acts on IP datagrams). This fundamental difference makes IPSec to be more flexible since it can be used with all the applications that use the IP protocol (i.e., almost all the existing network-enabled applications) in a transparent way. TLS can only be used over the TCP level and therefore only by applications using this protocol. Another advantage of IPSec is that, acting at a lower level in the ISO/OSI stack, it allows to be used also on intermediate machines between the first sender and the ultimate receiver of the datagrams.

IPSec has become more and more used, both for its flexibility and because it is included in IPv6 as a mandatory-to-implement part [108].

In the following subsection a brief description of IPSec is given.

1.3.1 The IPSec protocol suite

IPSec has been defined by the Internet Engineering Task Force (IETF) through a set of Requests For Comments (RFCs). While these documents do not define real standards, they are widely adopted for Internet protocols. As a matter of fact they provide a way for collective development (RFCs are developed by suitable working groups) of proposals. IPSec has become a *de-facto* standard over the years and it is going through the ISO standardization process.

IPSec is a suite of protocols developed to provide secure communications on untrusted networks adding some security services to the Internet Protocol (IP) level (i.e., the ISO-OSI network layer) [72]. As a matter of fact this suite of protocols operates on top of layer 3 but below layer 4 (TCP) of the ISO/OSI model. This infers that it encrypts data independently among the different packets. If packets happen to be lost, the layer 4 sees only validated information [99]. IPSec is the IETF proposed standard for "layer 3 real-time communication security". The IPSec suite of protocols has been gaining importance since its inclusion as mandatory security mechanism for IPv6 [109]. Some reports by international security organisms like CERT/CC (one of the most important computer security incident response team) [1] also refers to IPSec as a proposed solutions to security problems of the actual communication protocols. An example is reported in [31]; this document describes a vulnerability of the TCP protocol, and proposes IPSec adoption as a solution.

IPSec is composed of two different security protocols:

- Authentication Header (AH);
- Encapsulating Security Payload (ESP);

The former is used to protect the IP headers, while the latter is for protecting the content of the IP datagrams. A third protocol, the Internet Key Exchange (IKE), is proposed for key exchange and algorithm negotiation operations. The first two protocols can be combined in different ways to offer different levels of security services according to the established system security policy.

The main method used for AH is to apply an authentication algorithm on the header fields that are not going to be changed during the packet transmission. HMAC-MD5 is the algorithm proposed [70, 71] as a minimum requirement for IPSec conformance. Other algorithms such as the HMAC-SHA-1 have been always used and other new ones, such as HMAC-SHA-256, HMAC-SHA-348, and HMAC-SHA-512 [93], have been developed and are being used within IPSec. In addition, there is another required-to-implement authentication algorithm that is the NULL algorithm: an algorithm that does nothing.

The ESP protocol is implemented by applying an encryption algorithm on the data to be transmitted, but not on the IP header (except from the case of IPSec tunneling, as explained later in this section). The required-to-implement algorithms for IPSec compliant implementations are DES and the NULL algorithm, but, since IPSec was designed with flexibility and extensibility in mind, other encryption methods, such as AES, can be easily added.

Both in AH and ESP a simple and efficient anti-reply mechanism is provided: a monotonically increasing 32-bit counter is used to implement this feature [71]. Anti-reply is a process in which if someone were to intercept one of the packets exchanged by the two peers that are communicating, he could not use that packet to reply to one of the two peers to obtain reserved information (such as the symmetric key) – he would need to know the value of a field that is cryptographically encoded. Anti-reply is also called "partial sequence integrity".

To summarize:

- AH provides connectionless integrity, data origin authentication, and optional anti-replying service;
- ESP may provide confidentiality (using encryption) and may also provide connectionless integrity, data origin authentication, and anti-reply service if used in tunnel mode.

The Security Policy

System security settings are defined into the system security policy. Settings related to IPSec are stored into a database called Security Policy Database (SPD). The records of this database store the information about the security settings for all possible connections. A default policy must be defined for connections that do not match any other record into the database. The SPD needs to be queried for each outgoing packet and, theoretically, for each incoming packet.

Security Associations

The concept of *Security Association* (SA) is fundamental to IPSec. A SA is formally defined as a record in the Security Association Database (SAD). This database is used to store all the information that are necessary to configure the IPSec-based connections, including the symmetric keys that are used by them. A SA can be considered as a secure connection between two peers. An association in which either the AH or the ESP protocol (but not both) is used to communicate, is called IPSec Security Association (IPSec SA). The suffix "IPSec" is used to distinguish that kind of SAs from the Internet Security Association and Key Management Protocol (ISAKMP) SAs that can be used only for key exchanging and algorithm negotiation. In this document, unless specified, all the SA-word occurrences will be related to IPSec SAs. SAD need to be queried for each incoming or outgoing packet that involves IPSec processing.

An IPSec Security Association is a one-way association between two peers, so, in order to have a bi-directional communication channel, the creation of two SAs is needed.

For providing particular protection services, multiple SAs can be employed; this is called a "SA bundle". The order of the SA sequence is defined by the security policy, therefore if both AH and ESP are needed it will be necessary to create two SAs, one for AH and the other for ESP, so that these two SAs will be "nested" as required by the security policy.

We can note that, since a strong enough encryption algorithm is used, using ESP can offer the maximum level of protection available (see the ESP tunnel mode in the next section), so that nesting AH and ESP SAs seems unnecessary. As a matter of fact, AH is often seen as an additional, but not useful complication added to IPSec [99, 77]. From what it is stated in the IPSec RFCs, the AH protocol seems to have been kept mostly for backward compatibility purposes.

The transport and the tunnel mode

Two different modes are available both within the AH and the ESP protocols. These two modes provide support for different services; thus, datagrams are processed in different ways, depending on the mode.

The use of tunnel mode allows the inner IP header to be protected, concealing the identities of the (ultimate) traffic source and destination.

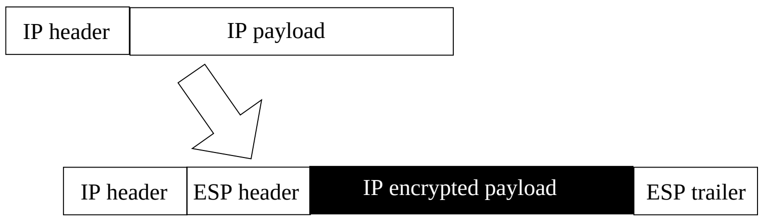

Figure 1.3: Datagram transformation for ESP transport mode

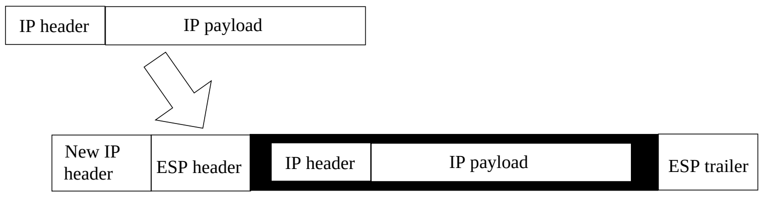

Figure 1.4: Datagram transformation for ESP tunnel mode

ESP padding can also be invoked to hide the real packet's size. The tunnel mode was primarily thought for being used in gateways or routers.

The Transport mode provides protection only for the upper layer protocols. Hence, by using ESP, the IP header is not protected, while using AH only some selected IP header fields are protected. As shown in Figure 1.3 and 1.4, the ESP transport mode protects the datagram's data payload, while ESP tunnel mode protects both the IP headers and the data payload. In the same manner, Figure 1.5 shows the behavior of the AH protocol in transport mode: the IP header is the only protected (hashed) part there. In Figure 1.6 the behavior of the AH protocol in tunnel mode is displayed: there, both the IP header and the payload are protected (hashed). In both the figures the AH field represents the part added by the AH protocol (hash, SPI, . . .). In the four figures shown here, the parts colored in black are the ones protected by cryptography or hashing; in AH the IP headers are never completely protected (i.e. some fields are not hashed, as explained before). All four figures are referred to IPSec used in combination with IP v.4. Slightly different figures can be drawn for IP v.6, since its structure allows a better IPSec integration. The effects obtained by using tunnel and transport modes are exactly the same for both IP v.4 and IP v.6. Please note that the four figures shown here are only a simplified view of the IP datagrams used for IPSec. More detailed information about the IPSec modes and the exact composition of the datagrams can be found in [70, 103].

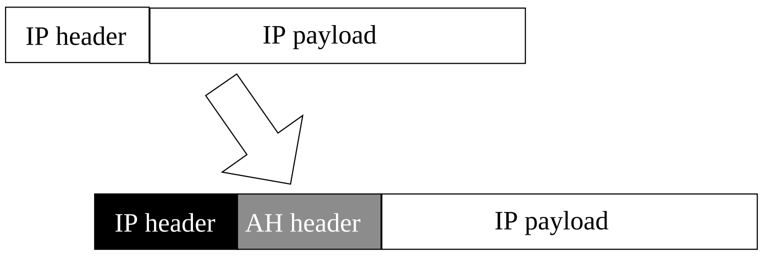

Figure 1.5: Datagram transformation for AH transport mode

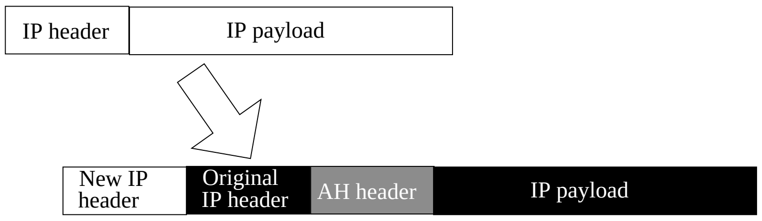

Figure 1.6: Datagram transformation for AH tunnel mode

The Internet Key Exchange Protocol

As stated before, a mechanism for symmetric key exchange and for algorithm negotiation is needed. It is important to note that the keys have to be exchanged in a secure way while the algorithm negotiation can be done in a non-protected way; this comes from one of the fundamental principles of cryptography, that says that the strength of a cryptographic algorithm is not given by hiding the algorithm itself, but by hiding the key. There is also a provision for protected negotiation in order to hide the identity of the peers or some other private information.

All mechanisms related to the creation of an IPSec SA must be done at the application layer and are described by the Internet Key Exchange (IKE) protocol. IKE is the interpretation of the Internet Security Association and Key Management Protocol (ISAKMP) in the IPSec domain. Therefore IKE is said to be the Domain of Interpretation (DoI) of ISAKMP. ISAKMP is a protocol describing how key exchange and algorithm negotiation should be done over the Internet network. The creation of an IPSec SA is completed in two phases [54]:

1. *Phase 1*: an ISAKMP SA between the two peers is created;
2. *Phase 2*: the ISAKMP SA created in Phase 1 is used to negotiate the information about the IPSec SAs that have to be created.

ISAKMP SAs are nothing more than a kind of secure tunnel for the creation of IPSec SAs.

IKE provides several methods for the Phase 1 negotiation with different levels of protection. The key exchange mechanism is based on the Diffie-Hellman algorithm. In Figure 1.7, one of the Phase 1 negotiation method is shown: here a public key encryption algorithm is used for authentication, while the secure channel for the Phase 2 is created during the second message exchange between the two peers, by using a symmetric encryption algorithm. The algorithms to be used are negotiated during the first message exchange between the initiator – which is the peer that propose to start an ISAKMP SA negotiation – and the responder, which is the other peer "responder": first the initiator sends, using the SA negotiation payload field, several complete algorithm proposals as defined by the system security policy, and then the responder puts the only proposal that it could accept in conformance with its security policy database (see section The Security Policy Database (SPD)) into the same field of its reply message.

A Phase 2 "quick mode" exchange is shown in Figure 1.6. The Phase 2 accomplishes the creation of a pair of independent SAs, one for each communication direction. A new pair of IPSec SAs is created by exchanging only three messages:

1. the Initiator requests a new pair of IPSec SAs proposing the encryption and authentication algorithms for use in each of those SAs payload;

2. the responder may accept one of the initiator's proposals by always using the SA payload field; the information for symmetric key creation are also exchanged within this message;

3. the Initiator confirms the creation of the two IPSec SAs by means of a message encrypted using the symmetric key exchanged within message 2.

After these steps, the two peers are able to communicate through the two secure (unidirectional) channels created during that negotiation. The Initiator of Phase 2 can be any of the two peers irrespective of which of the two was the Initiator during the Phase 1.

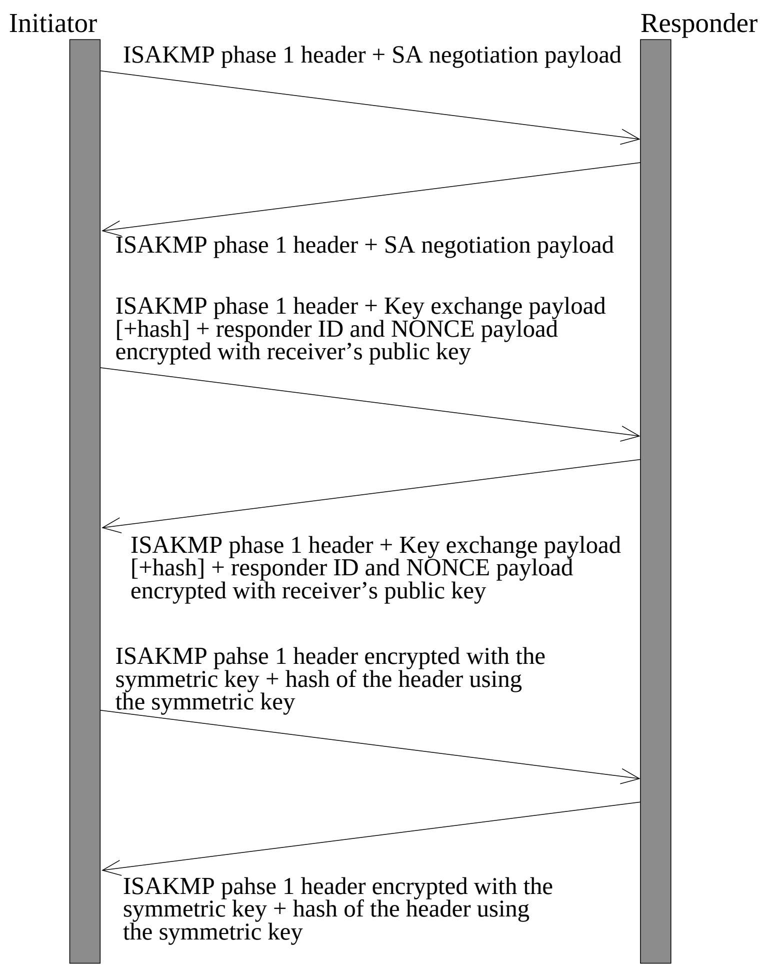
Initiator
Responder
ISAKMP phase 1 header + SA negotiation payload
ISAKMP phase 1 header + SA negotiation payload
ISAKMP phase 1 header + Key exchange payload [+hash] + responder ID and NONCE payload encrypted with receiver's public key
ISAKMP phase 1 header + Key exchange payload [+hash] + responder ID and NONCE payload encrypted with receiver's public key
ISAKMP pahse 1 header encrypted with the symmetric key + hash of the header using the symmetric key
ISAKMP pahse 1 header encrypted with the symmetric key + hash of the header using the symmetric key

Figure 1.7: IKE Phase 1 exchange

IPSec Packet Processing

Figure 1.9 and 1.10 show the processing steps which need to be applied to outbound and inbound IPSec packets, respectively. The main ones can be summarized as: SPD and SAD lookup, cryptographic processing, and header processing.

1.3.2 New Version of the IPSec Suite

The original IPSec RFCs date back in 1998; in the last years they are being updated to fix some problems of the first version of the suite. The first important improvement in the available draft is in readability of the

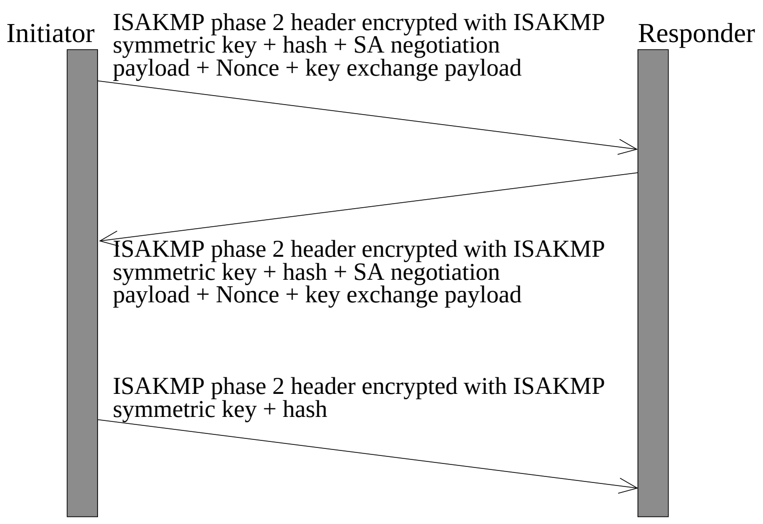
Initiator
Responder
ISAKMP phase 2 header encrypted with ISAKMP symmetric key + hash + SA negotiation payload + Nonce + key exchange payload
ISAKMP phase 2 header encrypted with ISAKMP symmetric key + hash + SA negotiation payload + Nonce + key exchange payload
ISAKMP phase 2 header encrypted with ISAKMP symmetric key + hash

Figure 1.8: IKE Phase 2 *quick mode* exchange

documentation. As a matter of fact, one of the main problem of the old RFCs was on that. Having a too complex definition of the protocols, have lead to have no implementation that are completely compliant with the RFCs. This have caused some compatibility issues between the different implementation.

AH and ESP are now defined in [73] and [74], respectively. There are no major differences among the old and the new versions of these protocols. The requirements for the cryptographic algorithms are now explained in a separate document [40]. The new version of the RFC2401 [75] introduces some suggestions for IPSec implementations and a better description of this suite. Some improvements over the processing model are introduced there. In this document a new database (along with the SAD and the SPD) is being introduced. This database is called Peer Authorization Database (PAD) and it provides a link between a SA management protocol (e.g., IKE) and the SPD. Along with other information, the PAD contains all the data that are necessary to authenticate peers (how to do it and possible shared secrets or certificates).

The IKE protocol has also been reviewed and a a version 2 is being written [69]. The main goal of this re-design is to obtain a simplest – but equally flexible and efficient – protocol. The main idea is that the old version of IKE is too complex and it provides too many usage modes. Most of them are insecure or can be substituted by only one. For this reason only the *Main Mode* has been kept for *Phase 2*. In IKEv2 a new database,

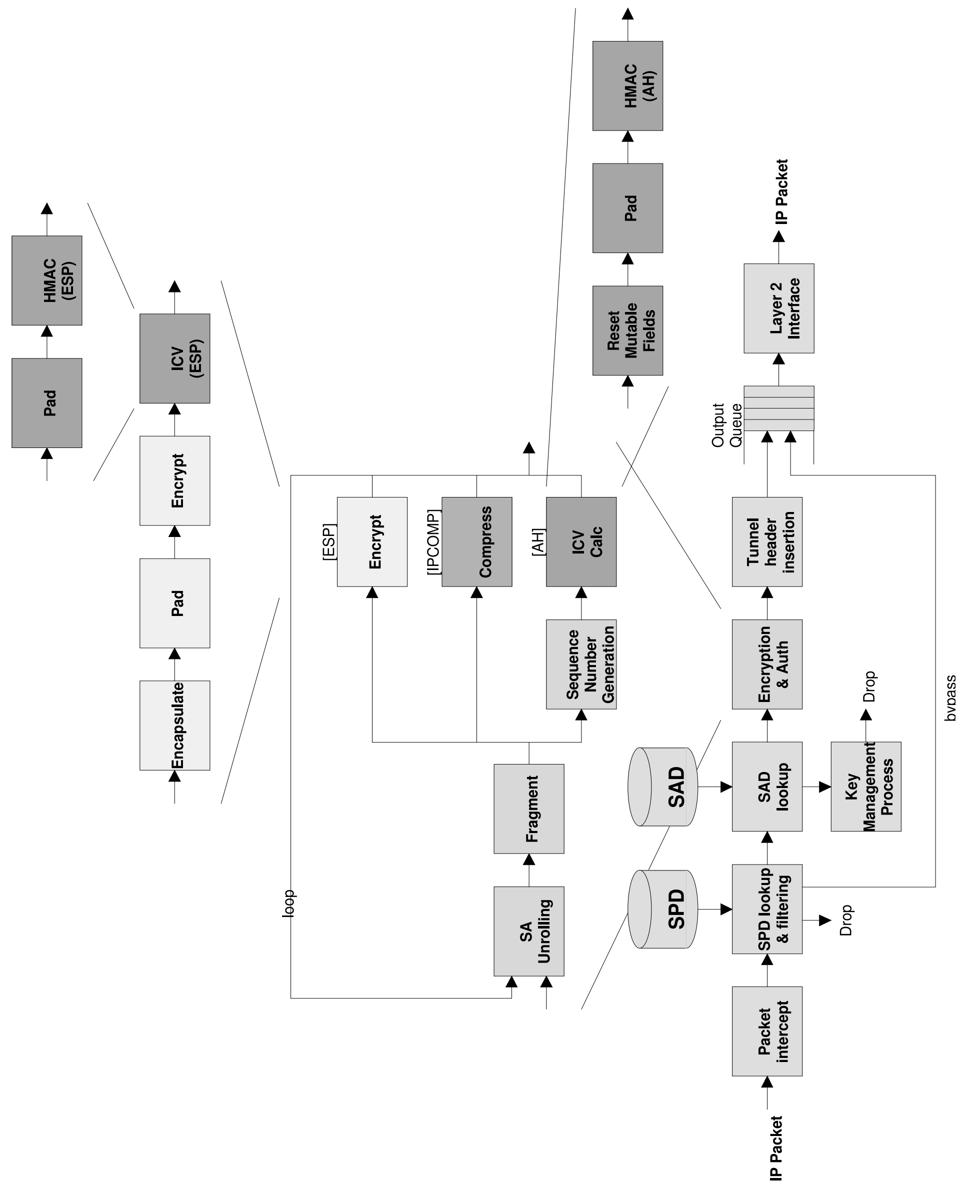

Figure 1.9: IPSec outbound packet processing.

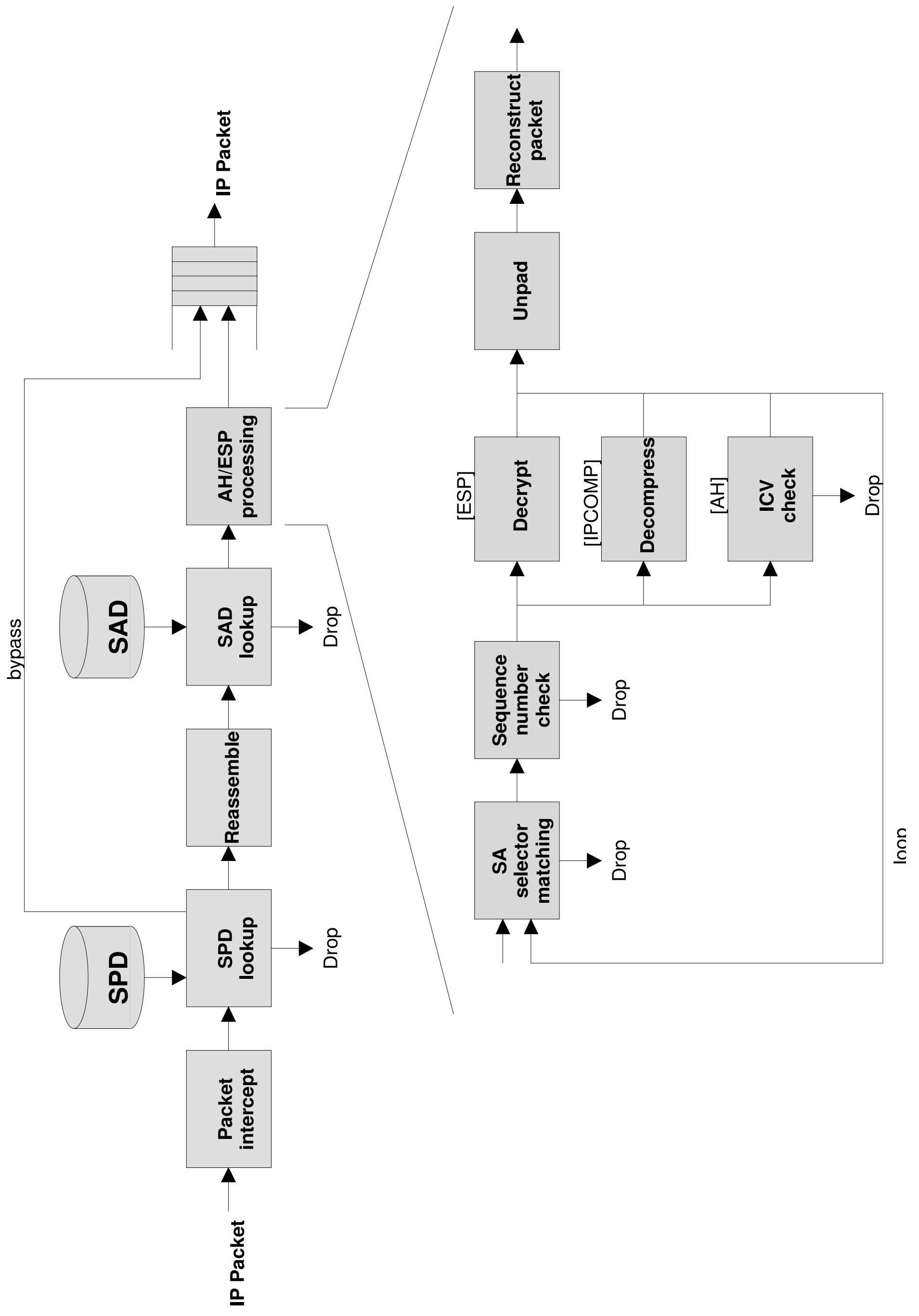

Figure 1.10: IPSec inbound packet processing.

the Peer Authentication Database (PAD), has been also introduced. The resources that are necessary for peers authentication (i.e., the public keys, or the certificates) are stored there. This database is just used by IKE and not directly managed by it.

1.3.3 Preliminary Evaluation of Hardware Requirements

As explained before, the security protocols rely heavily on cryptographic algorithms. It is pretty well known that these algorithms need many computational resources. When many connections need to be processed at the same time (in a concurrent way) also the security policy and security database processing has to be taken into account. As a matter of fact, the SPD needs to be queried for each packet traversing the system.

Symmetric Cryptographic Algorithms

With specific reference to IPSec, most of the processing power will be required - considering only one connection at a time - for symmetric encryption/decryption and for authentication of the data/headers.

The CPU load for symmetric cryptographic algorithms is maximum for the Triple-DES algorithm. AES and DES uses around one third of the resources used by Triple-DES.

For authentication HMAC-SHA-1 and HMAC-MD5 are usually used. The latter is considered to be more secure, but slower than the former. A new class of algorithms called HMAC-SHA-2 has been recently introduced. SHA-2 is an evolution/extention of SHA-1 that produces longer (256, 384, 512 bit) hashes. Therefore HMAC-SHA-2, that consists of HMAC applied on SHA-2, is slower (but safer) than HMAC-SHA1.

Public Key Cryptography Algorithms

As explained before for IPSec a security association negotiation phase has to take place for each secure connection to be established. In this phase public key cryptography algorithms may be used. Public key cryptography algorithms are the ones requiring most resources, but the data to be processed in this phase are very few. Considering many connections to be managed at the same time (this is what really happens in a secure gateway), the public key cryptography algorithms may play an important role in the throughput that a machine running IPSec can provide. While from the single connection stand point a long public algorithm processing introduces only a delay in the negotiation of a new security association, in a concurrent environment this may slower the whole system. Let us think about a system where a few milliseconds long public key cryptography computation can slower all the other processes, including the symmetric key cryptography ones. If many processes based on public key cryptography are present at the same time, this can result in a slow system.

1.4 Network Quality of Service

Network Quality of Service (QoS) is the ability to provide different levels of service to different fluxes of data. There exist two kinds of QoS: the hard and the soft ones [113, 107]. The former allows to negotiate some network parameters such as throughput and maximum latency and guarantees that these constraints are respected. The latter tries to provide different quality of treatment to different data fluxes. This is usually obtained by assigning different priority levels to these fluxes. The priority levels are decided at network level and in IPv4 they can be associated to each IP datagram by using a 3-bit header field. In IPv6 the size of this field has been increased to 12 bits, thus allowing to support more priority levels [109]. Priorities can be managed in different ways. The simplest one is to use a FIFO policy on the incoming packets, but other more effective ways exist: Priority Queuing, Custom Queuing, Flow-based Weighted Fair Queuing, and Class-based Weighted Fair Queuing are the most used ones. It is important to note that QoS is only useful in congestion management. When no congestion is experienced on the system, QoS does not introduce any benefice over the flux management and over the performances. Hard QoS is usually implemented through the *diffserv* infrastructure, which is based on priority management.

QoS has been assuming an increasing importance in the VPN environment, as services requiring real time support such as, for example, voice over IP are emerging.

2

IPSec Hardware Requirements Study

As already explained in the previous chapter, most of the algorithms used within the IPSec context (mainly the cryptographic and the compression ones) need a lot of computational resources. This can make processing data at line speed not feasible in certain cases. In this chapter a more detailed evaluation of the resources needed by IPSec is provided.

2.1 IPSec Performance Analysis

[24] provides a performance evaluation of a software implementation of IPSec in an IPv6 environment. The same evaluation is also provided for other protocols such as SSL. The IPSec implementation considered in that paper is the one included in the OpenBSD operating system. The results shown were obtained by considering 1GHz Intel Pentium III machines connected together through a 1Gbit/s network. The paper shows results of different tests, the first one is a data transfer between two hosts, while the second one is a data transfer passing by IPSec-based secure gateways. The authors of the paper concluded that enabling IPSec kills network performance in all cases. The authors also tested a cryptographic accelerator: this allowed to improve network performance by up to 100%. Even by using the accelerator, network performance are judged to be not very good. This makes the authors conclude that current generation of hardware cryptographic accelerators is not sufficient to support obiquitus use

of encryption. Furthermore, performance of actual crytpographic algorithms is good only for not too small packets. Considering that a large percentage of TCPI/IP packets (by far the most used protocol of the networks) are 40-byte length, this weakness is to be considered very important.

Some other performance studies are also available both as academic papers and as websites. [46] reports some performance considerations on the FreeS/WAN [4] IPSec implementation. A methodology for estimating the CPU overhead obtained with different IPSec configurations is also reported on this website. [121] reports some results obtained on a 100Mbit/s network by considering the IPSec Linux implementation included in 2.6 kernel series [76, 27]. This website only provides results for ESP in transport mode; performance results were obtained by considering the outdated triple DES with a 192-bit key as encryption algorithm and HMAC-SHA-1 as authentication algorithm. In the following section we provide an evaluation of network performance. In opposition to the previously presented documents, we also consider the CPU usage and effort spent as main parameters. A particular focus is put on the new cryptographic algorithms, AES and SHA-2. Different IPSec configurations are compared and their needs evaluated. The experiments not only take into account the security part of IPSec, but also the IPComp protocol. This protocol is included in IPSec and allows compressing the IP payloads by means of a compression algorithm [114]. This gives in many cases the possibility (depending on the type of data) to reduce the number of bytes to be sent on the network virtually widening its banwidth.

The target of this study is a secure gateway for low-end market. These machines are gaining importance as the number of network-enabled home devices increases.

2.2 IPSec Performance Measurement

In this section we describe the tests we conducted on an IPSec-based network and we discuss the related results. The aim of these tests was to provide a base from which to evaluate the requirement of IPSec for supporting 100Mbit/s and 10Mbit/s traffic. The contents of this section was published in a conference paper [20].

Our goal was to understand the CPU requirements of different configurations of the IPSec suite of protocols. Comparing different IPSec configu-

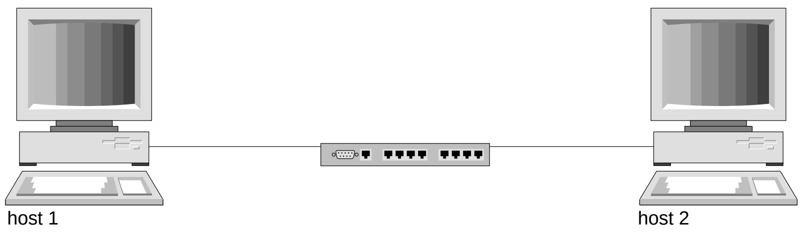

Figure 2.1: Test network structure.

rations was not our main goal, but it was done to understand the influence of the different parts (encryption, authentication, compression, . . .) of the protocols. The test we designed is based on sending a long piece of data (1Gbyte) between two PCs.

Here follow a description of the hardware, the software, and the IPSec configurations we used for the tests. The results we obtained by considering a 100Mbit/s and a 10Mbit/s network are then presented.

2.2.1 Hardware and Software Configuration of the Test Network

The test were done at the ALaRI institute of the University of Lugano by using two PCs running Linux RedHat 7.3 (kernel 2.4.18) patched with the FreeS/WAN 1.99 [4] implementation of IPSec. FreeS/WAN was also patched with the J. Ciarlante's modular algorithm patches (adding support for AES, NULL, and SHA-2 algorithms) [35]. While new Linux kernel releases (2.6) provide a new IPSec implementation, FreeS/WAN was chosen for these tests as it is well known and quite highly optimized for performance. Tests with the new implementation should anyway give results that are not to different from the ones here provided. Linux was chosen both because it is easy to modify (this will be very useful for our future works) and because it provides easy ways of applying measures. The network environment was based on the IPv4 protocol, but further tests will be conducted with IPv6 in the future.

The two PCs we used are 500MHz Intel Pentium III based and were connected to a 100Mbit/s network, as shown in Figure 2.1. While these PCs are to be considered obsolete machines, the results we show here are useful as we are considering small home-gateways as well as embedded systems. The results we obtained are also to be considered partially scalable to larger and more powerful systems.

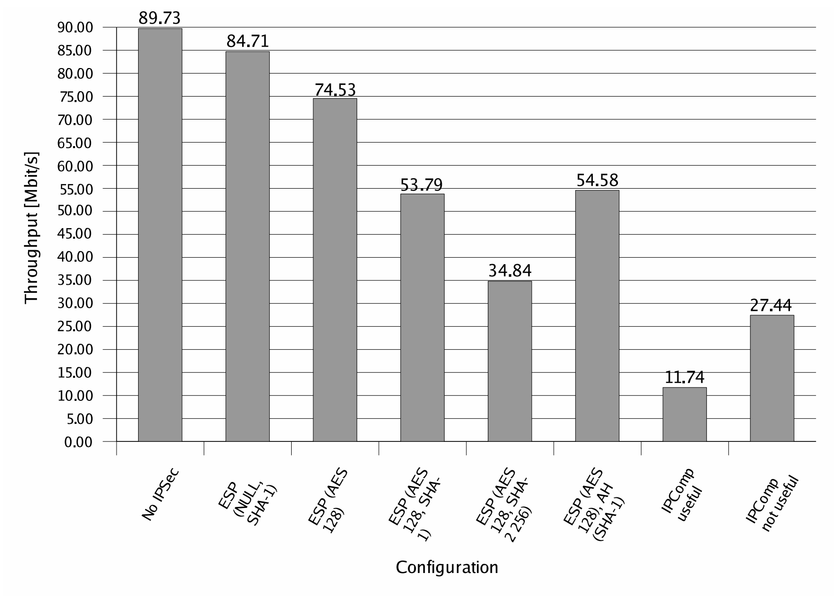

Figure 2.2: Network throughput for a 100Mbit/s network.

The tests were partly conducted by means of the *Netperf* tool [66], a tool for network performance evaluation. A set of Bash [45] scripts [86] were used to take track of instantaneous processor usage and network traffic. The scripts use the information available in the Linux *proc* interface [43]. This interface provides direct access to kernel settings and information.

2.2.2 Description of the Tests

We chose to use different IPSec configurations in order to be able to represent different usage scenarios and to understand which is the influence of each main part of IPSec on performance. Some of the configurations we chose can be used in real systems, while others are for test only.

For ESP encryption the AES symmetric cryptographic algorithm has been selected. So far the Triple-DES algorithm has been the default algorithm used in FreeS/WAN (single DES is the algorithm required for IPSec RFCs conformance). This algorithm is much slower than AES in software (up to 3 times, depending on the implementations) and is an outdated NIST standard for symmetric key cryptography. For ESP authentication the HMAC-SHA-1 algorithm was mainly used even if some tests were conducted by using HMAC-SHA-2 with a 256-bit signature.

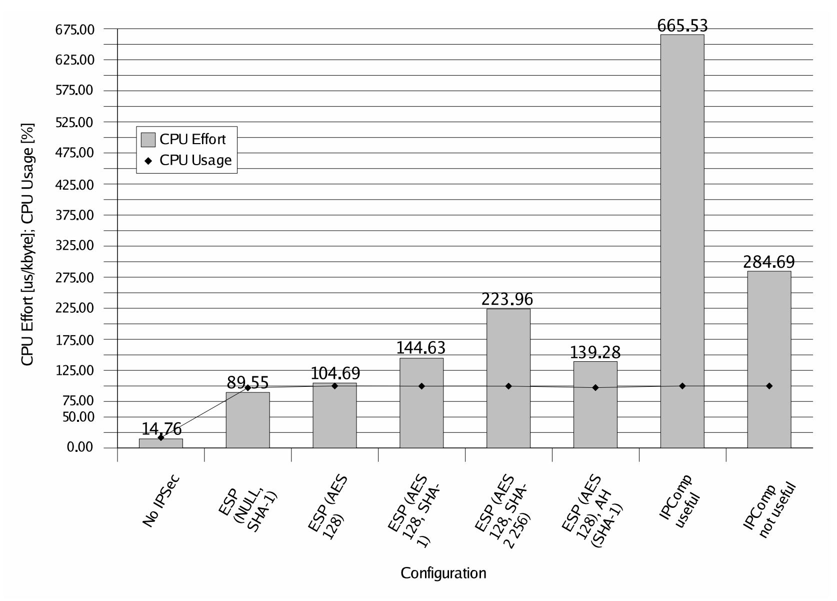

Figure 2.3: CPU effort comparison for a 100Mbit/s network. Numbers on the top of the bars represent the CPU effort.

The mode always selected for the tests is the tunnel mode, this is because the tests are dedicated to secure gateway machines. In FreeS/WAN it is possible to use the tunnel mode even on host-to-host connections by configuring the ends of the tunnel to be the hosts themselves.

Other tests were conducted by using the IPComp protocol (deflate [96] algorithm). The usage of this protocol was associated to ESP with AES 128-bit encryption and HMAC-SHA-1 authentication. We evaluated the effects of IPComp in two different cases. The first one corresponds to sending data on which compression has a good effect (i.e. the result of the compression operation is shorter than the original data[1]); in the second case a piece of data that cannot be further compressed (a bzip2 [67] file in our case) is sent. In the latter case sent datagrams are not compresses (as they cannot be), but a compressibility test needs to be run. The compressibility test consists of running compression on the payloads and comparing their dimensions with the original ones. Compressed payloads are used when they are smaller then the original ones; they are discarded otherwise.

[1]In some cases compression does not produce useful results; this happens either because the piece of data to be compressed is too small or because it cannot be further compressed. In such cases the result of the compression is larger – or, in any case, not smaller – than the original piece of data.

While it is widely known that the dimension of datagrams has a large influence on the performance of the protocols, we decided not to change this parameter during our tests. As explained before, our focus was not to globally evaluate IPSec performances, but to understand its requirements. Therefore we decided to use the datagram size of 1500byte. This is because we decided not to introduce a further parameter in our analysis which does not directly influence relative results. Furthermore, while the 1500byte packet size represents only the 10% of the packets sizes used during internet communications (the most part of the datagrams is 40byte length only), datagrams of this size carry around the 50% of the whole traffic [12, 84]. Some considerations about packet length and how this parameter influence the usage of cryptographic accelerators can be found in [90].

Since a long data set was chosen (1Gbyte) the tests were run only once each. Using a long data set instead of a short one, does not influence performance. In fact Netstat sends an user-defined amount of data by resending many times the same packet which is small enough to be held into the main memory. This is done in order not to have the measured performance influenced by the ones of the mass storage devices.

2.2.3 Results

100Mbit/s network

The throughput obtained on a 100Mbit/s network using different IPSec configurations is reported in Figure 2.2. While in Figure 2.3 the CPU effort expressed as CPU load in percentage and in time needed to process each sent kilobyte is shown. In all the figures *HMAC-SHA-1* and *HMAC-SHA-2* are respectively shortened to *SHA-1* and *SHA-2*.

Analyzing the reported results, some considerations can be pointed out. While the network capacity is the limiting factor for the network throughput when IPSec is not used, the CPU becomes the limiting factor as soon as IPSec with some form of encryption/authentication is used. Enabling the IPSec ESP protocol in tunnel mode and using authentication-only allows to sustain a data throughput close to the one obtained without enabling IPSec. Unfortunately in this case the CPU usage rises to 97%, more than 5 times the one obtained for the no-IPSec configuration. The authentication-only configuration is rarely used in practice, since it provides no data confidentiality. A configuration that may be usable in

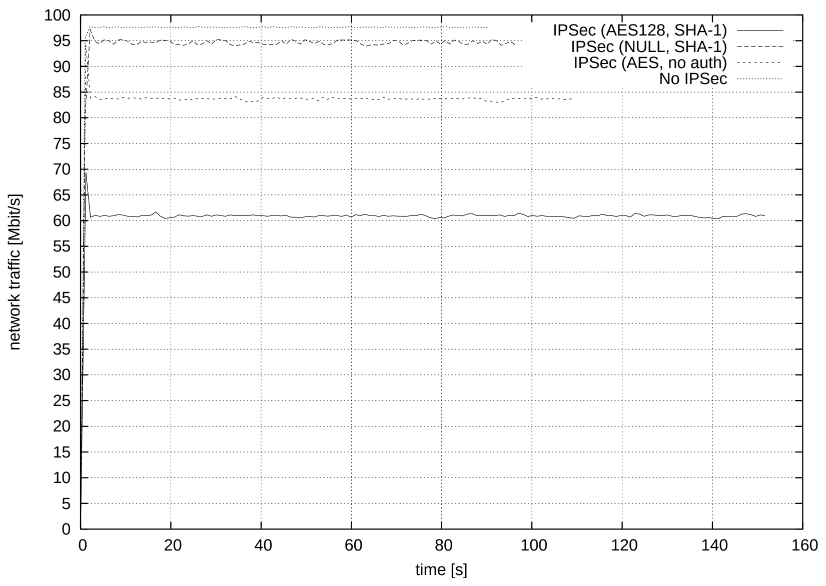

Figure 2.4: Sender PC Network output traffic for a 100Mbit/s network.

real systems – even if is usually believed to be not very safe – is that in which IPSec is used within the ESP protocol in tunnel mode with (AES) encryption enabled and without authentication. In this case the network throughput is decreased by the 17% with respect to the no-IPSec configuration; the CPU usage with this configuration is about 100%. This performance decrease is unacceptable in many cases. Enabling the HMAC-SHA-1 authentication, network throughput dramatically decreases from around 74Mbit/s to 53.8Mbit/s. Changing the authentication algorithm with the new HMAC-SHA-2, further decreases the network throughput to 34.8Mbit/s, less than half of the available bandwidth. Using the AH authentication associated with the ESP encryption, produces similar results to the ESP encryption plus authentication case. A representation of the output traffic measured on the sender PC for some of the main adopted configurations is shown in Figure 2.4. The average network traffic is slightly different from the network throughput since it also includes the protocols' headers. It is interesting to analyze the data relative to the CPU time used for processing every sent kilobyte. This gives an idea on the effort needed to process data for each configuration. Encryption only requires an effort that is 17% higher than in the authentication-only case. Encryption and authentication requires an effort that is 66% higher than authentication-only and 38% higher than encryption only.

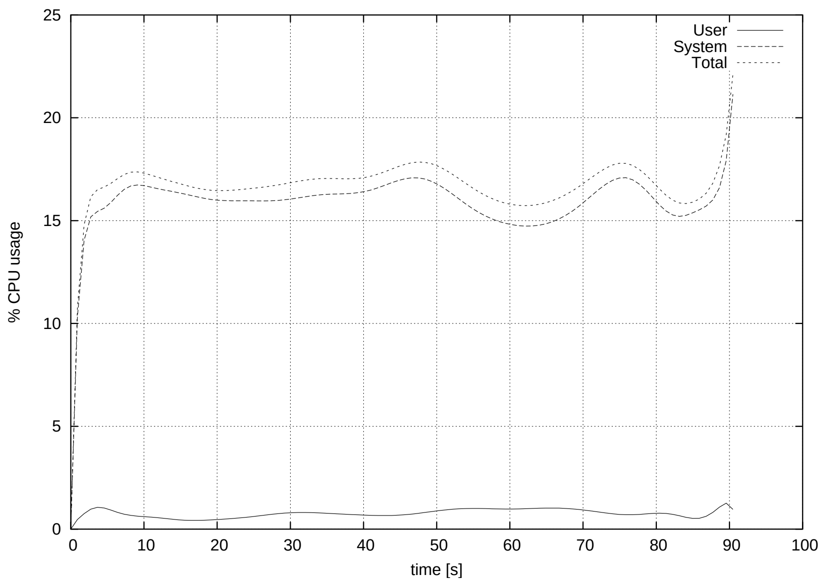

Figure 2.5: Instantaneous CPU load for a 100Mbit/s network when IPSec is not enabled.

Introducing IPComp further lowers the network throughput in every case, since, as explained before, at least the compressibility test needs to be executed. When IPComp is useful the throughput lowers to 11.74Mbit/s, 7.6 times slower than in the no-IPSec case and 4.5 times slower than in the IPSec "encryption+authentication(HMAC-SHA-1)" case. This happens even if the total network traffic (composed of data plus protocols' headers) is reduced by 46%. Even in the "not useful compression" case, the network throughput lowers considerably (27.44Mbit/s). This is due to the computational load introduced by the compressability test. The CPU effort is in both cases 665.53μs/kbyte and 284.68μs/kbyte respectively, much higher than all the other previously considered cases.

It is also possible to examine the CPU user and system load distribution for the cases presented above. The CPU load obtained for the "no IPSec" and for the "IPSec - AES - HMAC SHA-1" case are shown in Figure 2.5 and in Figure 2.6. From these figures it is evident that only the system CPU load is increasing when IPSec is enabled. This is easy explainable since all the IPSec-related processing is performed in kernel mode.

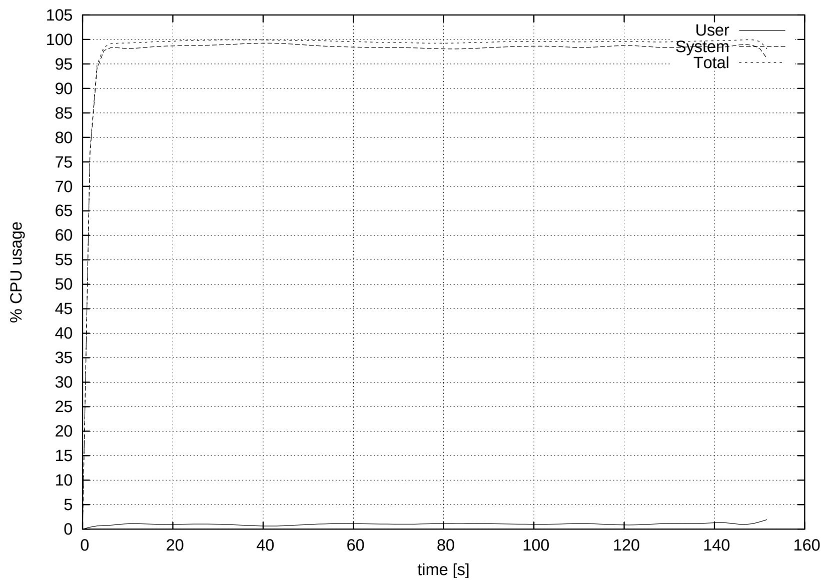

Figure 2.6: Instantaneous CPU load for a 100Mbit/s network when IPSec (ESP - AES 128bit - HMAC SHA-1) is enabled.

10Mbit/s network

Some tests were also conducted on a 10Mbit/s network. The network configuration utilized is the same as before, but the sender's bandwidth was limited through the kernel's device queue manager [30]. This is not a very precise method (in fact some little bursts at higher bandwidth are allowed), but we consider it to be precise enough for our evaluations.

In this case the CPU is no longer the limiting factor for the network throughput except in the case when IPComp is enabled and the data to be sent can be compressed. The CPU usage doubles when ESP encryption and authentication is enabled (the CPU utilization is around 2% when IPSec is not used and around 4% when it is used). By enabling IPComp and performing the tests with a file that can be compressed, an interesting result is obtained: the CPU usage goes to 100% so that the CPU become the limiting factor for the network throughput, but the throughput itself rises to 11.67Mbit/s (for a normal 10Mbit/s network without using IPSec we can obtain a maximum throughput of 9.58Mbit/s). This is due to compression that allows for the reduction of the total network traffic by 43% (from 1,070Mbyte to 603Mbyte). If more CPU power were available, the data transfer would be even faster. As a matter of fact 603Mbyte can be transferred in around 481s giving a network throughput of 17.44Mbit/s,

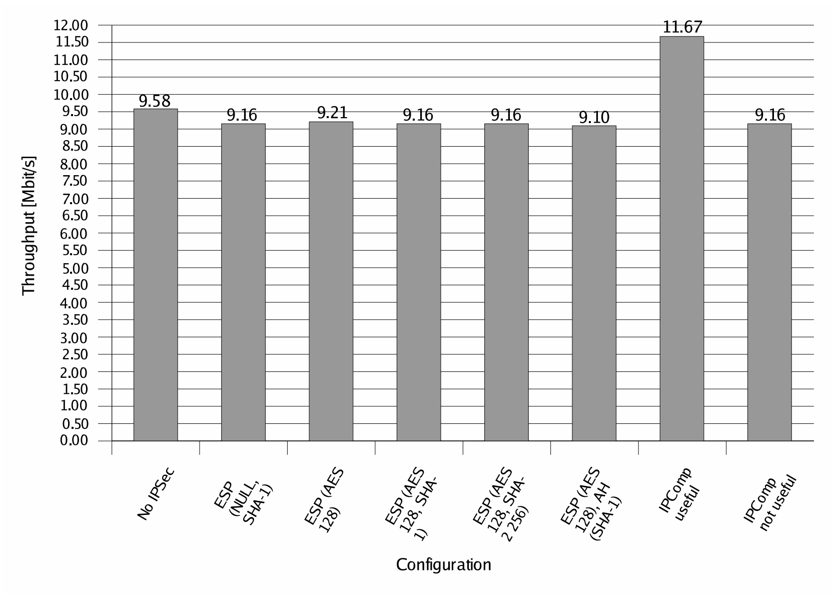

Figure 2.7: Network throughput for a 10Mbit/s network.

around 1.8 times faster than the "no IPSec" case. The network throughput results are shown in Figure 2.7.

The CPU effort obtained in this case is shown in Figure 2.8; the effort sustained by the CPU for running IPSec on a slow network is quite low, excluding the case of using IPComp on compressible data. In that case the CPU effort is 42 times the no-IPSec case. While in the 100Mbit/s network case the compressibility test introduced a further slowdown on the network throughput and the CPU effort to rise, in the 10Mbit/s network, both parameters do not change in a noticeable way using this configuration.

2.3 Remarks on Performance

Some evaluations can be done on the results shown above. We need to take into account that, considering the case of an IPSec-based secure gateway, we would also have the computational load due to managing large security association and security policy databases, in addiction to managing the connections and running the VPN server. The same machine will then possibly need to manage firewall rules (if also used as firewall) and routing tables (if also used as a router).

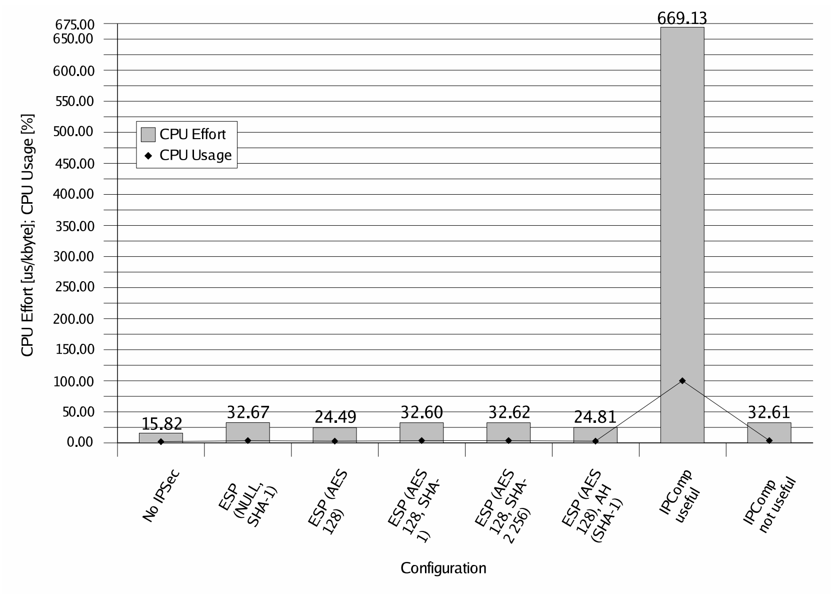

Figure 2.8: CPU effort comparison for a 10Mbit/s network. Numbers on the top of the bars represent the CPU effort.

Supposing, as normally done, that Gilder's Law and Moore's Law forecasts are right, the available network bandwidth is growing faster than CPU's computational capacity. Therefore, new strategies for supporting IPSec - and, more in general, secure protocols - need to be studied. When very high network bandwidths are considered, many effects have to be taken into account even if hardware accelerators are used. Often just adding a hardware accelerator is not enough and some hardware/software optimizations have to be put in place to obtain reasonable performances [83].

Even for slower networks some form of hardware acceleration can be a desirable option. As we have seen, using IPComp could allow us to reach considerably higher network throughput (and possibly lower power consumption due to network interface) and this could be really important in limited bandwidth conditions (for example DSL). At the same time, using IPComp can be resource consuming for devices on slow networks (e.g. small embedded systems). If a small IPSec co-processor (including IPComp acceleration) could be added to these devices, their network performance, efficiency, and security could be considerably improved.

While a hardware cryptographic accelerator is really key in reaching high performances on big systems, it may also be useful in lowering the CPU usage on small, slow systems.

2.4 Guidelines for IPSec Configuration

Some guidelines can be also derived from the results we have obtained. A first consideration should be done on the often used "encrypt everything" policy. As a matter of fact, using encryption when it is not really necessary it is just a resource wasting. In many cases people sending information over the Internet are not really concerned of their privacy, they are just concerned of their authenticity (i.e., being able to verify that data have not being changed during their transmission). Let us think, for example, about a network for collecting air pollution information. This network can be formed by many local measurement equipments sending data to a central database server though the Internet. In this case there is usually no interest in hiding information even though it is important to be be able to verify that the data have not being modified during their transmission. In this case, as in many others, a pure authentication-only policy (implemented, for example, through the ESP protocol with the HMAC-SHA2 algorithm) is enough to provide the necessary protection to data. MD5 and SHA1 have been recently broken [124, 125]. HMAC-SHA2 is therefore suggested to be used instead of HMAC-MD5 and HMAC-SHA1.

When also encryption is required, the correct algorithm need to be chosen to obtain a good level of performance and security. As a matter of fact triple-DES should not be used anymore, both because it has been declared obsolete by NIST and because it is slower than AES. Also the symmetric algorithm key-length selection plays a fundamental role in determining the performance that can be obtained. Using longer key sizes guarantees an higher level of security, but it requires more computational resources. As a matter of fact, 128-bit keys for AES are more than enough to protect most of the present normal communications. For example, the National Security Agency (NSA) has adopted AES for its classified documents: 196 and 256-bit keys are required by them for top-secret level information only [37].

Summarizing, the security-performance trade-off should be carefully evaluated before deploying an IPSec-based system not to waste too much resources without obtaining real benefits from the security stand point.

The IPComp protocol deserves some additional considerations: as shown before, it is very useful in some cases, but also very performance killing in some others. Evaluations on the usefulness of IPComp can be done a priori, by studying the traffic that will be sent over the considered channel. If average packet size is, for example, very small, IPComp should not be used. As a matter of fact, in this case most of the packets will not be compressed and a lot resources will be spent in non-useful compressibility tests. IPComp should also not be used in all the cases in which it is known that higher-level protocols already apply compression on data. As a matter of fact, also in this case a lot of resources can be wasted in non-useful compressibility tests.