

The Networking Perspective of Security Performance - a Measurement Study -

Heiko Niedermayer, Andreas Klenk, and Georg Carle
{Niedermayer, Klenk, Carle}@informatik.uni-tuebingen.de
<http://net.informatik.uni-tuebingen.de>

Computer Networks and Internet
University of Tuebingen
72076 Tuebingen, Germany

Abstract. The recent term Quality of Security Services leads directly to the question of the performance impact of security protocols like IPSec and SSL. The impact depends not only on the situation, but also on the configuration. We measured the processing delay and the throughput for implementations of IPSec under Linux with different kernel versions on current computers. Our focus is to cover the effect of the various parameters of IPSec. Most important for the IPSec performance is the choice of the cryptographic algorithms and hash functions. Our measurements indicate that the latter are becoming the bottleneck as fast encryption algorithms like the AES and Blowfish more and more replace the slow 3DES.

1 Introduction

The computational power of computers has been increasing ever since. Today, small handheld devices are more powerful than computers that are only some years old. Moreover, cryptographic functions have in many cases become more efficient, despite the need to persist computationally stronger adversaries. As a consequence, one might want to secure communication also in cases that are not highly critical. Another driving factor is the increase in wireless communication which is a general threat to confidentiality.

Considering the common Internet communication, it is usually insecure. Security in Internet communication is either managed on a per system basis by administrators or the applications themselves must deal with security. Especially applications frequently fall short of the users requirements. Examples are electronic ballot systems, online banking, confidential details during online transactions in the web, say payment details, etc.

More prevalent are security methods in case of wireless communication. However, WLAN security has been troublesome and WPA2 is not the default configuration, yet.

As a way of combining quality of service with quality of protection the new term Quality of Security Services (QoSS) has emerged. The standard assumption

is that communication may not only have different needs in quality of service, but also the need for protection differs. Varying the level of protection, however, makes only sense when there is a significant impact of the protection level on other parameters of the system. Such parameters can be CPU load, latency, or throughput.

2 Quality of Security Service

Many applications rely implicitly or explicitly on communication links which offer a certain quality of service (QoS). The latency of the link might determine, for instance, if real-time multimedia conferencing is feasible. The quality of service the end users experience is often limited by the weakest component of the processing chain of the communication. Either the network connection cannot hold the required QoS guarantees, middleboxes influence the communication parameters, or the end system itself cannot handle the load that the data flow imposes.

If only network characteristics are considered for QoS provisioning one might end up with a situation where the QoS of the network link might meet its constraints, but the applied security services could result in undesirable performance values for the application. Hence, the performance of the security services is an important factor for the QoS of the communication. A number of frameworks have emerged during the last years which can base their choice of the security services upon performance values and security ratings. The term *Quality of Security Service* (QoSS) was coined to emphasize the dependency between security algorithms and quality of service values.

2.1 Current research about Quality of Security Service

Levine was the first to discuss the effects of Quality of Security Service[6] in depth and to describe how to adapt the system to defined requirements. Her system offers security choices to reflect the different behavior of the mechanisms. The user, the application and the system determine the adequate algorithm jointly. The decision is derived from security ranges and performance ranges specified in form of policies. The core argument behind her reasoning is: If a user decides on a minimum security level for an application, would she ever agree to more security if that increases her costs?

The GSS-API[12] was developed prior to the QOSS approach to support applications with generalized security services. The per-message protection mechanism is enhanced by an option to take a so called Quality of Protection(QoP) parameter to select a particular security option based on performance values and the protection requirements of single messages. However, the use of the option is limited, because the QoP parameter depends on the implementation of the underlying algorithm which breaks the encapsulation of the mechanisms.

Ganz introduced the security broker[5] architecture for WLANs. This framework

chooses security services upon security requirements specified by the user, available network performance and the performance of security routines. There exist three security classes to choose from, each with a different level of protection and different performance.

Another contribution to adaptive security in WLANs was published by Saxena in [16]. The required security level depends mainly on the trust in the environment, say how hostile one expects the environment to behave. The paper argues that the spare processing and transmission resources are wasted in mobile environments if security is overprovisioned. Hence the tradeoff between security and performance is essential for the choice of security services.

ESAF[11] is a framework for adaptive security which supports end-to-end, layer independent and QoS aware configuration. Security mechanisms are virtualized and are thus exchangeable. The framework makes use of existing and generally trusted security protocols, for instance, IPSec and SSL. In the course of a negotiation between the partners mutually acceptable configuration options are chosen. The protection of the available algorithms and the estimated performance numbers determine the selection. Different stakeholders express their security and performance requirements by the means of scalar values. Policies describe the requirements in a high level description on the one hand and the available configuration options of the security services at the system level on the other hand. These policies contain scalar values to express a rating of the performance and security capabilities of the individual mechanisms. These scalars are required to make a meaningful selection of the mechanisms to reflect the users requirements. Hence the performance grading of a security mechanism is an important parameter.

All frameworks for the dynamic security adaptation have in common, that they require a thorough analysis about the performance of the available security protocols and demand for a quantification of security. The performance characteristics of existing security protocols have not yet been evaluated for the use by QoS aware systems. This submission strives to provide a performance overview of current security protocols and shows how scalar performance ratings can be derived from the results.

3 Related Work

The performance of individual cryptographic algorithms and their efficient implementation are topic of many special scientific conferences. However, the performance of cryptographic network protocols is not equally well documented. FreeS/WAN [4] is not only one of the first implementations of IPSec for Linux, its online documentation for version 1.99 is still a good source for IPSec performance implications. It primarily covers the CPU requirements to saturate a typical link using ESP with 3DES and AH with SHA-1. Modem, ISDN or DSL links can be saturated with even older computers (e.g. all i586 computers). For

Niedermayer, Klenk, Carle

standard 10 Mbps Ethernet (only 1 sender) an Intel Pentium with 300 Mhz is required.

Barbieri et al [2] studied in 2002 the use of IPSec for Voice over IP. Voice traffic consists of many packets with small payload, e.g. 40 bytes. The introduction of cryptography leads to an additional delay that can reduce the quality. Furthermore the bandwidth requirements increase due to the increased header size. They suggest a so-called compressed IPSec.

Alshamsi and Saito[1] just recently compared the performance of SSL and IPSec on current computers with Linux kernel 2.4.

4 IPSec

IPSec [10] is the *Security Architecture for IP* defined by the IETF which extends IP on the network layer. It strives to provide four security properties: message authentication, data integrity, replay protection and confidentiality.

4.1 Introduction

Upper layer protocols and IP communication profit from the security features of IPSec. It operates on top of IPv4 and is part of IPv6. Security can either be end-to-end or take some intermediate security gateways as end-points of the secured channel.

IPSec contains two security protocols. The Authentication Header (AH) [8] protocol protects the data integrity and provides for message authentication. The Encapsulated Security Payload (ESP)[9] protocol offers confidentiality as well as message authentication and data integrity. The ESP protocol can be combined with the AH.

The IPSec possesses two different protocol modes: *Tunnel Mode* for IP in IPSec encapsulation and *Transport Mode* for protection of the payload for host to host communication. These protocol modes are specified orthogonally to the security protocols and hence AH and ESP can each be used equally in Tunnel and Transport mode.

4.2 Authentication Header

This security protocols provides integrity, data origin authentication, and replay protection. Its scope is the whole static part of the IP packet including non-volatile IP header fields, the payload, but also the AH specific header information. The Authentication Header protocol can utilize different authentication algorithms. *Message Authentication Codes* (MAC) protect point-to-point connections. One way hash functions (e.g., MD5, SHA-1) or symmetric encryption algorithms (e.g., 3DES, Blowfish, AES) can be used.

The length of the Authentication header is 12 bytes plus the length of the authentication data. Usually, the output of the hash function is not included in full length, but truncated (in case of MD5 and SHA-1 to 12 bytes, in case of SHA2-256 to 16 bytes).

4.3 Encapsulated Security Payload

ESP offers similar protection for the payload like the AH and provides additionally for confidentiality. The ESP operates on the payload of the packet and provides additional protection for ESP header fields. It does not cover static fields of the IP header.

The ESP header is inserted after the IP header and before upper layer data (transport layer) or encapsulated IP packets in the Tunnel Mode. Confidentiality can be specified alone, whether in this circumstance other means for authentication like the AH protocol are required for secure operation. Traffic flow confidentiality can be achieved in conjunction with the Tunnel Mode.

The optional authentication uses the same functions for MAC generation as the AH. The already stated encryption algorithms serve for the confidentiality of the payload.

The length of the Encapsulated Security Payload header is 10 bytes plus the length of the padding, optionally the length of the authentication data, and the length of an Initialization Vector (usually 8 bytes) if required by the encryption algorithm.

4.4 Security Associations and Security Policies

IP itself is a datagram service. However, IPSec requires a common knowledge of keys and algorithms for all communication partners. Thus, the security context has connection semantics to keep state information for the applied security services.

The standard [10] introduces the notion of *Security Associations* (SA) as "a simplex 'connection' that affords security services to the traffic carried by it". One SA defines security services for a unidirectional connection either for AH or ESP but not for both. Multiple SAs can be effective for one connection, for instance, four SAs: AH, ESP, each inbound and outbound. The standard also defines how a SA can be identified: "A security association is uniquely identified by a triple consisting of a Security Parameter Index (SPI), an IP Destination Address, and a security protocol (AH or ESP) identifier." The SPI is used to identify the SA. The *Security Policy Database* (SPD) stores policies that define which inbound and outbound traffic must be protected by which security services. The SPD defines how SAs must be established and what parameters are necessary. The administration of the system local SPD is usually managed manually.

The *Internet Key Exchange* (IKE) is used to establish authenticated keying material and maintaining SAs. It runs on top of the ISAKMP [13] framework and utilizes the Diffie-Hellman algorithm to set up a shared session key. It can use pre-shared secrets or X.509 certificates to authenticate the entities. The IETF is currently standardizing the *Internet Key Exchange (IKEv2) Protocol* [7] which integrates previously independent standards (e.g., ISAKMP, Internet DOI) and introduces new functionalities (e.g., NAT traversal, Legacy Authentication, Remote Address Acquisition). However, IKEv2 is not interoperable with IKE.

5 Latency of IPSec processing

5.1 Measurement Methodology

In this section we present measurements of the processing delay of IPSec. To be more precise, the delay a packet receives from entering to leaving the IPSec layer. These measurement points were set using hooks in the Linux IP stack. Unless otherwise stated the results presented are from outgoing packets and were measured from IP_Local_Out to IP_Post_Routing hooks. The time measurements were done using the Pentium CPU time-stamp-counter (TSC) register. All measurements were performed using Intel Pentium 4 2,60 GHz computers with Linux 2.6.9 and its native IPSec and all unnecessary services terminated. For comparison we also used Linux 2.4 with StrongS/WAN[17].

The measurement software stored the results in an array during the evaluation and purged them to disk afterwards. For the measurement, single ICMP packets with different packet sizes were sent using the command ping. 10.000 samples were taken for each combination of the algorithms and outliers due to system specific distortions, say interrupts, were manually removed using the statistics software R [3]. The impact of the outlier removal on the mean is at most 2,7 %, but usually well below. Due to the low variation and the large number of samples the confidence intervals are narrow.

One limitation of this paper is that we solely look at the latency of sending a packet. However, there might be small differences for the results of packet reception caused by different performance of encrypting and decrypting a packet, impact of caching, etc.

5.2 Authentication Header

The Authentication Header Protocol is the IPSec protocol for message authentication and data integrity, usually achieved using cryptographic hash function in the HMAC construct. Table 1 presents our measurements with AH using the native IPSec of Linux 2.6.9. As expected SHA-1 is slower than MD5.

Algorithm	Mode	Cycles	Time
MD5	Transport	23665	9.10 μs
MD5	Tunnel	24591	9.46 μs
SHA-1	Transport	63628	24.5 μs
SHA-1	Tunnel	65916	25.3 μs

Table 1. AH - delay for packet with 1400 B payload between IP_Local_Out to IP_Post_Routing

The processing of AH with HMAC-MD5 needs approximately 23,500 cycles in Transport Mode. This corresponds to a data rate of 1.2 Gbps (ignoring other

overhead and internal limitations). The processing of SHA-1 takes approximately 64,000 cycles. The corresponding data rate would be 460 Mbps.

5.3 Encryption with Encapsulated Security Payload

The Encapsulated Security Payload is the IPSec protocol for confidentiality, usually provided by symmetric encryption, e.g. with block ciphers in CBC-mode as in all our measurements. Table 2 presents our measurements with ESP using the native IPSec of Linux 2.6.9. The AES is the fastest encryption algorithm in these measurements. Its delay for processing a segment with 1,400 bytes payload length is about 65,000 cycles, which corresponds to an assumed data rate of 440 Mbps (ignoring other overhead and internal limitations). The slowest algorithm is 3DES with a delay of 395,000 cycles and corresponding data rate of 74 Mbps.

Algorithm	Mode	Cycles	Time
AES-128	Transport	65629	25.2 μs
AES-128	Tunnel	66620	25.6 μs
AES-192	Transport	70976	27.3 μs
AES-192	Tunnel	72927	28.0 μs
Blowfish-128	Transport	112603	43.3 μs
Blowfish-128	Tunnel	116292	44.7 μs
3DES	Transport	394956	152 μs
3DES	Tunnel	398989	153 μs

Table 2. ESP - delay for packet with 1400 B payload between IP_Local_Out to IP_Post_Routing

5.4 Combining ESP and AH

A realistic scenario combines both ESP and AH to achieve a maximum of security. Table 3 lists some results. A typical combination is ESP with AES-128 and AH with SHA-1. The processing of IPSec in this case takes about 131,000 cycles, which corresponds to a data rate of 222 Mbps (ignoring other overhead). Using ESP with 3DES in this combination increases the processing overhead to 451,000 cycles (data rate 64.5 Mbps), which is more than factor 3! The Tunnel Mode that is used for scenarios like Virtual Private Networks. It is approximately 2,000 to 6,000 cycles slower than the Transport Mode.

5.5 Packet Size

All the results presented so far were for large packets that are close to the size limit imposed by the standard Ethernet. This is one typical packet size, but smaller packets are also common, e.g. for ACKs or real-time traffic. Table 4

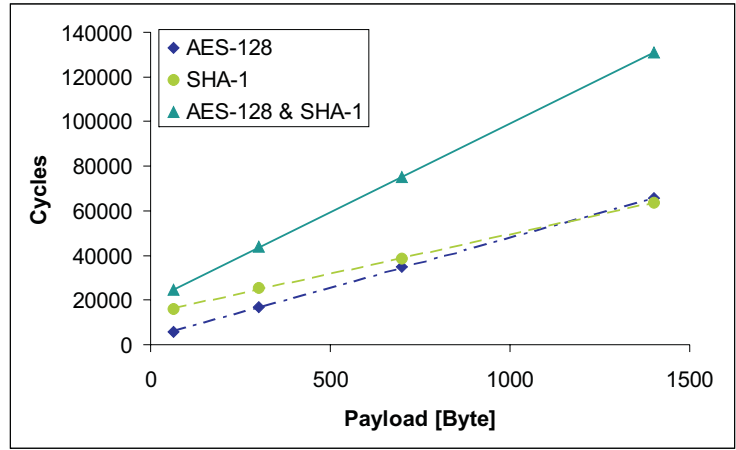


Fig. 1. IPsec with kernel 2.6.9 in Transport Mode Delay

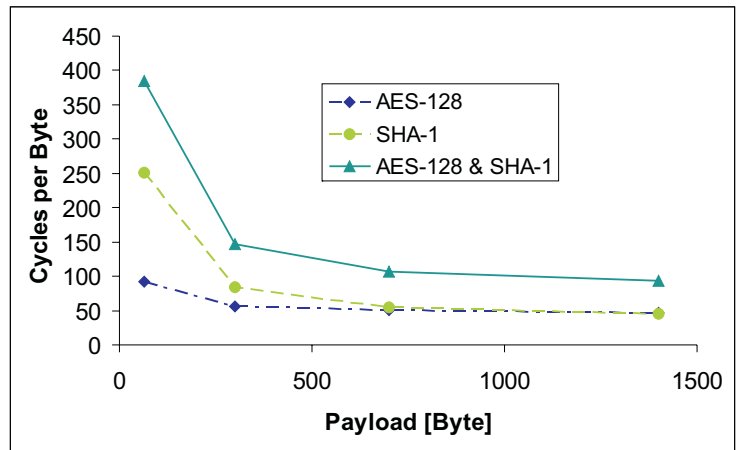


Fig. 2. IPsec with kernel 2.6.9 in Transport Mode Delay per Byte

ESP	AH	Tunnel Mode	Transport Mode
AES-128	SHA-1	133481	131012
3DES	SHA-1	456558	450979

Table 3. ESP+AH - delay in clock cycles for packet with 1400 B payload between IP_Local_Out to IP_Post_Routing

ESP	AH	Mode	64 B	300 B	700 B	1400 B
AES-128	-	Transport	5907	16890	35069	65628
AES-128	-	Tunnel	6607	17565	35233	66620
AES-128	SHA-1	Transport	24587	43944	74956	131012
AES-128	SHA-1	Tunnel	25084	44946	76666	133481
-	SHA-1	Transport	16037	25365	38773	63627
-	SHA-1	Tunnel	18433	25264	41031	65915

Table 4. Native IPSec/Linux 2.6.9, Delay in clock cycles for packets with different payload size between IP_Local_Out to IP_Post_Routing

presents the delay measurements for packets with the sizes 64 B, 300 B, 700 B, and 1,400 B.

Figure 1 visualizes the delays for different packet sizes for the measurements with IPSec in Transport Mode. The linear regression is also plotted and shows that a linear model

$$Latency_{avg}(Size) = a * Size + b$$

is already sufficient. a gives the number of cycles per byte and b the constant overhead in cycles. For the case with AES-128 and SHA-1 in Transport Mode linear regression gives $a = 79.5 \frac{cycles}{Byte}$ and $b = 19,700 cycles$.

Figure 2 gives another insight on the data. As expected, the overhead per byte is larger for small packets. For packets with a payload of 300 B and more the overhead per byte is already close to the overhead per byte for large packets. Thus, the constant term of the processing delay is mainly important for small packets, e.g. packets in audio streams with small payload sizes.

5.6 Comparison of Linux 2.6.9 and StrongS/WAN

When we started the measurements the use of Linux 2.4 was still common and IPSec was not yet part of the kernel. We used StrongS/WAN which is based on the FreeS/WAN project which stopped their development when a different IPSec implementation was to be integrated in the 2.6 kernel. Table 5 shows the results.

Even more interesting is the comparison of these results with the results from the previous sections where we used the native IPSec implementation of Linux

ESP	AH	Mode	64 B	300 B	700 B	1400 B
AES-128	-	Transport	20918	26294	37465	49585
AES-128	-	Tunnel	25964	31293	39816	54726
AES-128	SHA-1	Transport	44518	58224	75235	114530
AES-128	SHA-1	Tunnel	48847	62353	82495	115117
-	SHA-1	Transport	32186	40336	50160	69861
-	SHA-1	Tunnel	42665	44470	60486	80080

Table 5. StrongS/WAN/Linux 2.4 - IPsec-related Delay in clock cycles for packets with different payload size

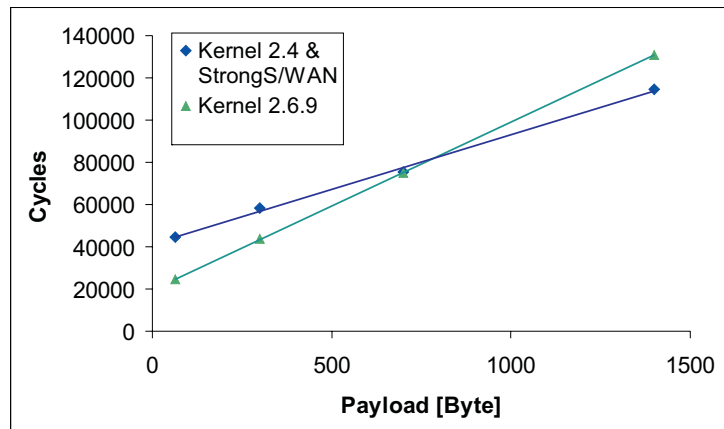


Fig. 3. Comparing the measurements for StrongS/WAN with the measurements with Linux 2.6.9

with kernel 2.6.9. Figure 3 shows the comparison of the results. StrongS/WAN performs better for large packets and the native IPsec of Linux 2.6.9 is better for small packets, e.g. 64 bytes packet size.

One might speculate that while the algorithms are better optimized in StrongS/WAN, native IPsec is naturally better integrated into the kernel and has less overhead itself.

6 Throughput

6.1 Measurement Methodology

In this section we present throughput measurements mainly for IPsec with various configurations. All measurements were performed using Intel Pentium 4 2,60 GHz computers with Fedora Linux and its native IPsec implementation with all unnecessary services terminated. The computers were directly connected via Gigabit Ethernet. Iperf was used as a tool to perform throughput measurements.

The values are mean values taken from at least 3 Iperf runs (9 runs for encryption only and 18 runs for hash function only) with each run having 30 seconds duration.

At first, we performed the measurements with the same kernel (Fedora Linux 2.6.9) where we performed the latency measurements within the IP stack. Just recently we repeated the measurements with the new 2.6.13 kernel. The new version has an improved DES implementation among other improvements in the kernel. However, the general performance increase was surprising to us, but our tests with another 2.6.9 kernel variant showed results similar results to our 2.6.9 reference distribution. However, there is a variation between installations with the different distributions.

Unless otherwise stated we given values are from measurements with the kernel 2.6.9 in a standard Fedora installation.

6.2 Authentication Header

Table 6 and table 7 provide an overview of the throughput measurements with AH. AH with MD5 hardly reduced the throughput of IP datagrams. It is interesting that the newer 2.6.9 kernel has a lower IP and MD5 throughput, but all other IPsec combination profit from its use. The use of SHA-1, the most commonly used cryptographic hash function, leads to a throughput of roughly 300 Mbps (kernel 2.6.9) and 350 Mbps (kernel 2.6.13). The use of the most secure hash function in our measurements, SHA-2-256, reduced the performance of AH to 186 Mbps.

Algorithm	kernel 2.6.9	kernel 2.6.13
IP	825 Mbps	817 Mbps
AH with MD5	610 Mbps	599 Mbps
AH with SHA-1	298 Mbps	349 Mbps
AH with SHA-2-256	186 Mbps	189 Mbps

Table 6. AH performance in Transport Mode

Algorithm	kernel 2.6.9	kernel 2.6.13
IP	825 Mbps	817 Mbps
AH with MD5	595 Mbps	580 Mbps
AH with SHA-1	294 Mbps	343 Mbps
AH with SHA-2-256	183 Mbps	186 Mbps

Table 7. AH performance in Tunnel Mode

6.3 Encryption with Encapsulated Security Payload

The Encapsulated Security Payload is the IPSec protocol for confidentiality, usually achieved using symmetric encryption, e.g. with block ciphers in CBC-mode. Tables 8 and 9 present the results. The performance of all the algorithms dramatically increased when switching from kernel 2.6.9 to 2.6.13. The throughput of ESP with AES-128 increased from 314 Mbps (kernel 2.6.9) to 456 Mbps (kernel 2.6.13).

Blowfish is designed to have a performance independent of the key size. This can also be seen in the results.

With the 2.6.13 kernel, Fast Ethernet can almost be saturated using ESP with 3DES.

Algorithm	kernel 2.6.9	kernel 2.6.13
IP	825 Mbps	817 Mbps
ESP with AES-128	314 Mbps	456 Mbps
ESP with AES-192	295 Mbps	419 Mbps
ESP with Blowfish-128	192 Mbps	336 Mbps
ESP with Blowfish-192	192 Mbps	337 Mbps
ESP with DES	132 Mbps	212 Mbps
ESP with 3DES	66 Mbps	97 Mbps

Table 8. ESP performance in Transport Mode

Algorithm	kernel 2.6.9	kernel 2.6.13
IP	825 Mbps	817 Mbps
ESP with AES-128	306 Mbps	441 Mbps
ESP with AES-192	287 Mbps	404 Mbps
ESP with Blowfish-128	191 Mbps	325 Mbps
ESP with Blowfish-192	189 Mbps	324 Mbps
ESP with DES	127 Mbps	206 Mbps
ESP with 3DES	64 Mbps	95 Mbps

Table 9. ESP performance in Tunnel Mode

6.4 Combining ESP and AH

True security requires both encryption and message authentication. Thus, it is necessary to combine both. We skip the values of combinations with DES as is insecure due to its small key size (56 bits relevant, key itself is 64 bit).

	MD5	SHA-1	SHA-2-256
AES-128	231 Mbps	166 Mbps	123 Mbps
AES-192	225 Mbps	160 Mbps	120 Mbps
Blowfish-128	164 Mbps	127 Mbps	99 Mbps
Blowfish-192	163 Mbps	127 Mbps	99 Mbps
3DES	60 Mbps	56 Mbps	50 Mbps

Table 10. ESP+AH performance in Transport Mode, kernel 2.6.9

	MD5	SHA-1	SHA-2-256
AES-128	309 Mbps	223 Mbps	148 Mbps
AES-192	289 Mbps	213 Mbps	142 Mbps
Blowfish-128	244 Mbps	190 Mbps	129 Mbps
Blowfish-192	244 Mbps	190 Mbps	130 Mbps
3DES	88 Mbps	81 Mbps	69 Mbps

Table 11. ESP+AH performance in Transport Mode, kernel 2.6.13

Tables 10 and 11 present the results. The fastest reasonable IPSec combination is ESP with AES-128 and AH with SHA-1. Its throughput is 166 Mbps (kernel 2.6.9) or 223 Mbps (kernel 2.6.13) in Transport Mode. Comparing this section with the results of the ESP and AH sections we additionally note that message authentication with the hash functions is increasingly dominating the performance, especially considering that SHA-2-256 is the only hash function in this study that is not yet heavily under attack.

6.5 AH vs ESP

With IPSec it is not necessary to use ESP and AH for achieving confidentiality and message authentication as ESP supports both. Many experts recommend the use of ESP and AH, because AH also authenticates non-volatile fields of its IP header.

AH adds 12 B extra overhead and additionally authenticates the outer IP header. So, the combination of ESP and AH should be slower. However, the extra 12 bytes AH header are small compared to payloads of more than 1400 B.

Comparing the results presented in Table 12 there is no significant impact on the performance whether ESP and AH or ESP with authentication is used. Thus, the recommendation is to use ESP and AH as it provides better authentication with no or negligible extra-cost. An exception to this recommendation are small packets and the additional overhead of 12 B for AH might be unacceptable.

6.6 Comparison with SSL-based Tunnels

Finally, we compare IPSec tunnels with tunnels based on SSL. Besides the native IPSec of Linux 2.6 we use the SSL tools stunnel [14] and OpenVPN [15].

Algorithms	ESP	ESP & AH
AES-128/SHA-1	167 Mbps	166 Mbps
AES-128/SHA-2-256	125 Mbps	123 Mbps
Blowfish-128/SHA-1	126 Mbps	127 Mbps
Blowfish-128/SHA-2-256	99 Mbps	99 Mbps
3DES/SHA-1	56 Mbps	56 Mbps
3DES/SHA-2-256	51 Mbps	50 Mbps

Table 12. ESP vs ESP+AH, kernel 2.6.9

Algorithms	stunnel	OpenVPN
null	-	519 Mbps
AES-128 & SHA-1	180 Mbps	117 Mbps
Blowfish-128 & SHA-1	-	149 Mbps

Table 13. Performance of SSL tunnels

The performance values for the SSL-based tunnels are given in Table 13. IPSec with MD5 achieves a better performance than OpenVPN without encryption and authentication.

Using OpenVPN with AES is slow compared to Stunnel. However, the Blowfish implementation for SSL is rather efficient. Thus, if Blowfish is preferred as encryption algorithm OpenVPN is an interesting solution.

In general, the SSL tunnel tools partially outperform IPSec of kernel 2.6.9, but are usually slower than IPSec of kernel 2.6.13.

7 Discussion

In this section, we finally discuss our results and try to put them in a general context.

7.1 Observations from the measurements

Our results exposed some interesting characteristics of the behavior of IPSec. The average processing latency increases linearly for the evaluated algorithms which is not surprising. However, there is a significant overhead for small packets below 300 bytes. We experienced that the behavior of the security services depends a lot on the IPSec implementation and also the linux kernel. The tested encryption algorithms of Linux kernel 2.6.13 showed remarkably better performance values than the ones of kernel 2.6.9. There are also differences between the StrongS/WAN implementation and the IPSec of the kernel 2.6.9. StrongS/WAN under kernel 2.4 performs better for small packets but cannot keep up with the current kernel for packets bigger than 700 bytes. These observations indicate that a careful implementation of the security services and the network stack have a large impact on the performance.

7.2 Hash functions

Recently, there has been a discussion about the necessity to start a contest for the standardization of a new and strong hash function. This was mainly initiated by the fact that MD5 is broken and SHA-1 has been heavily under attack. However, these attacks are mainly important for the security of digital signatures. Authentication in Internet communication is still considered secure.

Our results from the IPsec measurements add one more argument to the discussion about hash functions. ESP-AES performs better than AH-HMAC-SHA1 and ESP-Blowfish better than AH-HMAC-SHA-2-256. Thus, message authentication has turned into the bigger performance bottleneck than encryption. This has been different for the last decades and was unanticipated when we started the evaluation. Thus, the development and deployment a new secure and fast cryptographic hash function seems to be desirable.

7.3 Performance impact of security services

Security services operate on a rate out of reach for standard computers in the recent years. The performance of cryptographic algorithms and hash functions is no obstacle anymore for their wide scale deployment. Even the slowest combination (50 Mbps, 3DES/SHA-2-256) exceeds the connection speed of typical home users by far (e.g. current DSL in Germany at 6 Mbps).

However, performance remains still an issue. Particularly servers with high bandwidth connectivity, say VPN gateways demand for highly efficient algorithms. Moreover, servers which do not exclusively serve as security gateway but provide other functionalities must limit the impact of security services. Resource constrained devices, on the other hand, do not possess nearly as much resources as our test systems. Security is only one issue that may influence the performance. Even when its load is not the dominating bottleneck it is not necessary to do it inefficiently and waste energy.

The bandwidth of Gigabit Ethernet networks still exceeds the capacity of our test system to secure traffic. The expected evolution of network technology in fixed and wireless networks toward higher bandwidths may increase the gap.

7.4 Quality of Security Service

Most QoS aware systems require a rating of the available security services. We will show exemplarily for ESAF [11] how the discussed algorithms can be classified. It is important to keep in mind that such ratings are subjective in nature and can only represent some usage scenarios. Hence the ESAF considers only locally available ratings to determine which protocols might be acceptable. There is no exchange of the ratings during the negotiation of the communication context for a connection.

Our results from the presented measurement study are useful to derive the performance levels of encryption and authentication services as shown in Table 14. The other scalar value expresses a rating about the assumed security for each

Algorithm	Confidentiality	Performance
AES-128	8	8
AES-192	9	7
DES	2	4
3DES	8	2
Blowfish-128	9	6
Blowfish-192	10	6
Algorithm	Authentication	Performance
HMAC-MD5	2	9
HMAC-SHA-1	5	6
HMAC-SHA-2-256	8	3

Table 14. Possible values for ESAF policies

algorithm. This rating is subjective in nature as well and can have a large impact on the choices the framework makes.

Let us consider an example. An application wants to establish a communication link with some security requirements. One requirement is that the minimal authentication security level is at least 4. The HMAC-SHA-1 would be chosen in our example, because its security rating is sufficient and it is faster than HMAC-SHA-2-256. If the HMAC-MD5 would possess a better security rating, say 4, it would be selected due to the better performance value.

8 Conclusion

We presented a performance study of security protocols and security services. Our focus was on the measurement of the different configuration options of IPSec. Reasonable IPSec configurations (e.g. ESP with AES-128 and AH with SHA-1) on our test systems with 2.6 GHz processors achieved up to 220 Mbps over a Gigabit Ethernet link. These numbers are promising and demonstrate that the IPSec security services do not limit the communication bandwidth for typical usage scenarios, say in home networks, for Internet connectivity or between workstations in business environments.

The emerging Quality of Security Service research that deals with performance vs security issues can profit from our results. We gave an example how to quantify the performance of the configuration options. Adaptive algorithms can use such ratings as an input to their utility function.

9 Acknowledgements

We would like to thank our students Andreas Rabi and Andreas Korsten for their programming efforts. Parts of this research were gratefully supported

by the Landesstiftung Baden Wuerttemberg within the SEMOBIS project (Semantically Oriented Software Engineering for Mobile Information Systems in an Entrepreneurial/Business Context).

References

1. Abdel Nasir Alshamsi and Takamichi Saito. A technical comparison of IPsec and SSL. In *19th International Conference on Advanced Information Networking and Applications (AINA 2005)*, 2005.
2. Roberto Barbieri, Danilo Bruschi, and Emilia Rosti. Voice over IPsec: Analysis and solutions. In *18th Annual Computer Security Applications Conference (ACSAC 2002)*, 2002.
3. The R Project for Statistical Computing. <http://www.r-project.org>.
4. FreeS/WAN. <http://www.freeswan.org>.
5. Aura Ganz, Se Hyun Park, and Zvi Ganz. Security broker for multimedia wireless lans: Design, implementation and testbed. 1998.
6. Cynthia Irvine and Timothy Levin. Quality of security service. 2000.
7. Charlie Kaufman. Internet key exchange (ikev2) protocol. Internet Draft (draft-ietf-ipsec-ikev2-17.txt), 2004.
8. Stephen Kent and Randall Atkinson. Ip authentication header. Internet Draft (RFC2402), 1998.
9. Stephen Kent and Randall Atkinson. Ip encapsulating security payload (esp). Internet Draft (RFC2406), 1998.
10. Stephen Kent and Randall Atkinson. Security architecture for the internet protocol. IETF, 1998.
11. Andreas Klenk, Marcus Masekwoy, Heiko Niedermayer, and Georg Carle. ESAF - an extensible security adaptation framework. In *NordSec 2005 - The 10th Nordic Workshop on Secure IT-systems*, October 2005.
12. J. Linn. Generic security service application program interface, version 2. IETF, 1997.
13. Douglas Maughan, Mark Schertler, Mark Schneider, and Jeff Turner. Internet security association and key management protocol (ISAKMP). Internet Draft (draft-ietf-ipsec-isakmp-08), 1997.
14. Stunnel multiplatform SSL tunneling proxy. <http://stunnel.mirt.net>.
15. OpenVPN. <http://www.openvpn.org>.
16. Anshuman B. Saxena. An adaptive security framework for wireless adhoc networks. *Wireless World Research Forum (WWRF)*, 2004. Euro-Labs.
17. StrongS/WAN. <http://www.strongswan.org>.