A Technical Comparison of IPSec and SSL

AbdelNasir Alshamsi * Takamichi Saito †
Tokyo University of Technology

Abstract

IPSec (IP Security) and SSL (Secure Socket Layer) have been the most robust and most potential tools available for securing communications over the Internet. Both IPSec and SSL have advantages and shortcomings. Yet no paper has been found comparing the two protocols in terms of characteristic and functionality. Our objective is to present an analysis of security and performance properties for IPSec and SSL.

1 Introduction

Securing data over the network is hard and complicated issue while the threat of data modification and data interruption is rising. The goal of network security is to provide *confidentiality*, *integrity* and *authenticity*.

Confidentiality is keeping the data secret from the unintended listeners on the network. Integrity is ensuring that the received data is the data was actually sent. Authenticity is proving the identity of the endpoint to ensure that the end point is the intended entity to communicate with.

The combination of these properties is the pillar of the security protocols. How to combine them is the question with many answers. Using a strong cryptographic key with a weak authentication algorithm may allow an attacker to disrupt the data. Using a strong authentication algorithm with a weak encryption algorithm may allow an attacker to decrypt the data. Using both strong authentication and encryption algorithm protects the data but it will decrease the transmission rate and could induce CPU consumption. Therefore, it is complicated to provide the best protection, the maximum throughput and the lowest overhead.

With the recent development of the security tools, so many protocols and powerful tools have been proposed, but the most famous, secure and widely deployed are IPSec (IP Security) [1] and SSL (Secure Socket Layer) [2].

In this paper we will provide a technical comparison of IPSec and SSL; the similarities and the differences of the cryptographic properties. The results of performance are based on comparing FreeS/WAN [3] as IPSec and Stunnel [4] as SSL.

2 IPSec

IPSec [1] is an IP layer protocol that enables the sending and receiving of cryptographically protected packets of any kind (TCP,UDP,ICMP,etc) without any modification. IPSec provides two kinds of cryptographic services. Based on necessity, IPSec can provide confidentiality and authenticity (1) or it can provide authenticity only (2):

1. ESP (Encapsulated Security Payload) [5]
2. AH (Authentication Header) [6]

ESP provides confidentiality, authenticity and integrity protection for the communication and it is distinguished by the ESP header attached to the packet. AH on the other hand ensures that authenticity and integrity of the data is protected and it is identified by the AH header attached to the packet. ESP header includes the necessary information for decrypting and authenticating the data where AH header includes the necessary information required for authenticating the protected data.

Establishing IPSec connection requires two phases: Phase 1 (ISAKMP SA) [7] and Phase 2 (IPSec SA) [8] (see table 1).

2.1 Phase 1

Phase 1 performs mutual authentication and produces the encryption key required to protect Phase 2. Phase 1 has two modes: Main Mode and Aggressive Mode. The differences between these two modes are

*Graduate School of System Electronics, 1404-1 Katakurcho, Hachiouji City, Japan. alshamsi@aqua.ts.it.teu.ac.jp

†Department of Computer Science, 1404-1 Katakurcho, Hachiouji City, Japan. saito@cc.teu.ac.jp

the number of messages exchanged and the ID protection (see table 2).

1. **Main Mode**

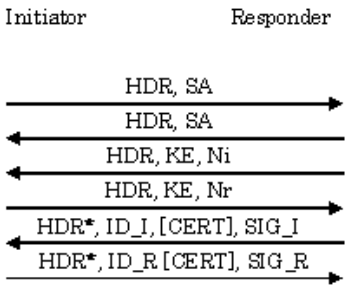

Figure 1: IPSec Main Mode

2. **Aggressive Mode**

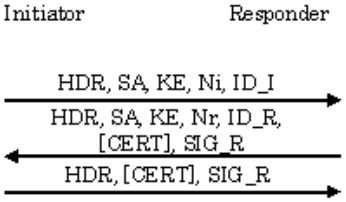

Figure 2: IPSec Aggressive Mode

2.2 Phase 2

Phase 2 negotiates the cipher and authentication algorithm required to protect further transactions. Phase 2 has one mode, Quick Mode.

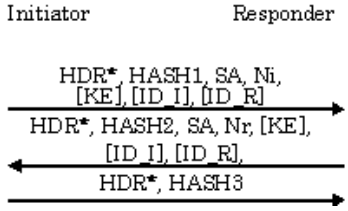

Figure 3: IPSec Quick Mode

The notations below were used to describe Figure 1,2 and 3.

IPSec Notation

HDR: ISAKMP header.

SA: Security Association.

KE: Diffie-Hellman exchanged public value.

Ni, Nr: the nonce.

ID_I, ID_R: the Initiator, Responder.

CERT: the certificate.

SIG_I, SIG_R: the signature for the Initiator, Responder respectively.

[x]: x is optional.

*** :** encryption must begin after the header.

2.3 Key Exchange and Authentication

Various methods of Key Exchange mechanism and Authentication methods are supported by IPSec.

Key Exchange Method:

1. DH
2. KINK[1] [9]

Authentication Method:

1. Pre-Shared Key (PSK)
2. Digital Signature [2]
3. Public Key
4. KINK (see 4.2)

The Hash Algorithms used by IPSec are MD5 and SHA-1 (see 4.3).

Table 1: IPSec Mode

Phase	Key Exchange Mode	Msg. Exchanged
Phase 1	Main Mode	6
	Aggressive Mode	3
Phase 2	Quick Mode	3

[1] KINK is a Kerberos based protocol that provides Key Exchange and authentication mechanism, Not standardized yet.

[2] such as RSA Digital Signature and DSA Digital Signature

Table 2: ID Protection

Mode	Authentication Method	ID Protection
Main	PSK	yes
	RSA/DSA Digital Sig.	yes
	Public Key	yes
Aggressive	PSK	no
	RSA/DSA Digital Sig.	no
	Public Key	yes

3 SSL

SSL (Secure Socket Layer) [2] is an Application layer protocol. SSL is mostly utilized to protect HTTP transactions, and has been used for other purposes like IMAP and POP3, etc. SSL is compatible with applications running only over TCP, but some modifications are required for the applications to run over SSL. Recent development of SSL software like Stunnel [4] has added an easiness of use to SSL. SSL is composed of the following protocols:

1. Handshake protocol
2. Change Cipher Spec protocol
3. Alert protocol
4. Application Data protocol

Handshake protocol is used to perform authentication and key exchanges. Change Cipher Spec protocol is used to indicate that the chosen keys will now be used. Alert protocol is used for signaling errors and session closure. Application Data protocol transmits and receives encrypted data.

3.1 Key Exchange and Authentication

Key Exchange Method:

1. RSA
   Client sends the *pre_master_secret* after encrypting it with Server's public key.
2. DH
   Client and Server exchange DH public values and produce the pre_master_secret independently.

Other Key Exchange methods are used with SSL[3] , but in this paper we will only refer to the above mentioned methods.

Authentication Method:

[3] Kerberos, FORTEZZA

1. Server Authentication
2. Client Authentication
3. Anonymous (see 4.2).

Client Authentication is in fact mutual authentication but the developers of SSL has referred to it as "Client Authentication". The Hash Algorithms used by SSL are MD5 and SHA-1 (see 4.3).

An example of a SSL Handshake is described in Figure 4:

Figure 4: SSL Handshake

Figure 4 describes SSL handshake in Client Authentication. The remove of the '*' attached exchanges represents Server Authentication.

SSL Notation

ClientHello: Client proposes supported cipher suite.

ServerHello: Server sends chosen cipher suite.

Certificate: Server sends certificate.

CertificateRequest: Server requests Client's Certificate.

ServerHelloDone: Server has sent all Handshake messages.

Certificate: Client sends Certificate.

ClientKeyExchange: Client sends pre_master_secret encrypted with Server's public key.

ChangeCipherSpec Client sends "New chosen cipher suite will be selected"

Finished: finished message

ChangeCipherSpec: Server sends "New chosen cipher suite will be selected"

Finished: finished message

4 Comparison of IPSec and SSL

4.1 Authentication Algorithm

IPSec supports the use of Digital Signature and the use of a Secret Key Algorithm, where SSL supports only the use of Digital Signature. The use of a random 2048 bit Secret Key is considered as strong as any other authentication methods. In the absence of Digital Signature algorithm, IPSec can still be implemented using the Secret Key but SSL can't be implemented.

4.2 Authentication Method

IPSec supports one type of authentication method while SSL supports a various types of authentication. They are described in table 3 and 4.

Table 3: IPSec Authentication Method

Authentication Method	Authentication Algorithm
Mutual Authentication	PSK
	RSA/DSA Digital Signature
	RSA Public Key
	KINK

Table 4: SSL Authentication Method

Authentication Method	Authentication Algorithm
Server Authentication	RSA (Challenge/Response)
	DSA Digital Signature
Client Authentication	RSA/DSA Digital Signature
Anonymous	none

4.3 MAC

MAC (Message Authentication Code) is used for authenticating the exchanged messages after the connection is established. Both IPSec and SSL require the implementation of HMAC-SHA-1 and HMAC-MD5. HMAC is a hash function that requires a secret key to produce message digest. The strength of the Hash Algorithm is based on the length of the output (see table 5).

Table 5: HMAC Algorithm Type

Protocol	MAC Algorithm	Hash Length
IPSec	HMAC-SHA-1-96 [10]	12 Byte
	HMAC-MD5-96 [11]	12 Byte
SSL	HMAC-SHA-1	20 Byte
	HMAC-MD5	16 Byte

4.4 Connection Mode

IPSec has two connection modes:

- Tunnel Mode
  This is established between Gateway-to-Gateway, Gateway-to-Host and Host-to-Host. It establishes a tunnel between the endpoint and it requires adding a new IP header to the original packet.

- Transport Mode
  Transport Mode is Host-to-Host connection. The data between the two entities are encrypted.

The advantage of Tunnel Mode is the elimination of the overhead caused by each channel. But the disadvantage is what could happen to the connection if the key was compromised.

SSL is a vice versa situation. SSL is one connection per one session type. Each session is independent but the throughput could fall down as the number of session's increases.

4.5 Remote Access

IPSec encounters some problems when it comes to remote access with PSK authentication in Main Mode. In the case of PSK, the ID is restricted to the IP address only. Since the IP address is not static, the proper shared key can't be located and as a result Remote Host can't be established. To avoid such incident the following methods are proposed:

1. Setting the remote host IP to 0.0.0.0 and using one Shared Key for remote host access.

2. Using Aggressive Mode, where the ID type is not restricted to the IP address and it is sent to the responder at the beginning of the negotiation.

3. Adapting User Authentication scheme like PIC [4] [12] or XAUTH [5] [13]

Using one key can compromise the security of the network if it is lost or stolen. Besides, changing the key from time to time must require all clients to adjust their setting. Aggressive Mode sends the Client ID in the clear and with that some risk exists. User Authentication can add overhead to the process, but the results are satisfying. XAUTH is proposed as a User Authentication protocol for IPSec where the user authentication is conducted after the completion of Phase 1 and before starting Phase 2. If the user is authenticated then the service is granted (see table 6). Remote access with Digital Signature doesn't require any modification.

Table 6: Remote Access Proposal for PSK

Solution	Risk	Overhead
One PSK key	high	low
Aggressive Mode	medium	low
User Authentication	low	high

SSL authentication over TCP is based on the exchange of RSA (Challenge/Response) or DSA Digital Signature during Server Authentication and RSA/DSA Digital Signature during Client Authentication and therefore remote access is performed without any needs for modification.

4.6 Around Transport Layer

IPSec Phase 1 negotiations are exchanged over the UDP (port 500 only). Thus, *Retransmit Timer* must be prepared to maintain. SSL Handshake is exchanged over TCP and unlike IPSec; the port can be changed according to the application.

As a Server, both IPSec and SSL are bound to specific ports where as a Client, IPSec is bound to specific ports but SSL is not (see table 7).

SSL works only over the TCP since UDP can cause data to be arbitrarily lost or re-ordered. IPSec avoids the UDP problem by adding a new TCP header to the original packet's field, which allow UDP or TCP based applications to work with IPSec. Supporting only TCP application is a shortcoming of SSL.

When IPSec is behind a firewall all ports of IPSec must be in a permanent listening status at the firewall. SSL situation is different, when it is behind a firewall,

[4] Pre-IKE Credential Provisioning Protocol
[5] Extended Authentication

Table 7: IPSec, SSL and Ports

Protocol	Mode	Ports
IPSec	Server	ESP 50/TCP
		AH 51/TCP
	Client	ESP 50/TCP
		AH 51/TCP
SSL	Server	HTTPS 443/TCP, etc.
	Client	any

the ports aren't open until SSL successfully negotiated at the firewall and then forwarded to the SSL requiring application.

4.7 Perfect Forward Secrecy

Both IPSec and SSL use PFS (Perfect Forward Secrecy) in their resumption session.

In the case of IPSec, the main goal for Phase 1 beside authentication is producing the encryption key required to safe guard Phase 2's exchange. Compromising Phase 1's key will result in decrypting Phase 2. This will allow an attacker to produce the key and decrypt the data exchanged between the two sides. PFS prevents such attack by exchanging new DH values each time a session is resumed.

In the case of SSL, PFS is implemented in the same manner as with IPSec when Ephemeral Diffie-Hellman is negotiated.

4.8 Order of Cryptographic Operations

IPSec encrypts the data first then creates MAC for the encrypted data. If a modified data were inserted in the middle of transaction, IPSec would verify the MAC before performing any decryption process [1].

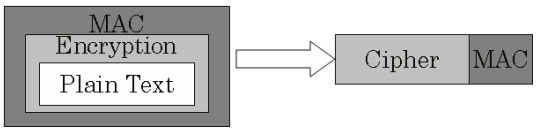

Figure 5: IPSec

SSL is the opposite; it creates the MAC for the plaintext first then encrypts the data. SSL on the other hand, is obligated to decrypt it first then verifies the MAC which could result in wasting CPU over decrypting modified packets.

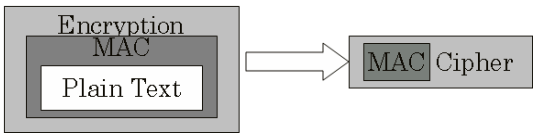

Figure 6: SSL

4.9 Cipher List Proposal

Because IPSec is a two phase protocol, it has a unique function called *bi-directional* [8]. That is when IPSec Phase 1 is established, both Initiator and Responder can initiate Phase 2 negotiation. SSL is a one direction protocol.

4.10 Interoperability

IPSec doesn't integrate well with other IPSec vendors [14]. Some cases require some modification. SSL is trouble free and well integrated.

4.11 Overhead Size

One disadvantage of IPSec is the extra size added to the original packet. SSL needs less overhead than IPSec. The extra required bytes for each protocol are described in table 8.

Table 8: Overhead Size

Protocol	Mode	Byte Size
IPSec Tunnel Mode	ESP	32
	ESP and AH	44
IPSec Transport Mode	ESP	36
	ESP and AH	48
SSL	HMAC-MD5	21
	HMAC-SHA-1	25

IPSec Tunnel Mode requires adding another 20 Byte IP header (see 4.4). All the data above don't include the padding bytes and the pad length.

4.12 Residing Layer

IPSec resides in the IP layer which allows it to work with the above layers smoothly, SSL resides in the Application layer and that is a problem for some application to work with SSL. As we mentioned, Stunnel is a solution for the TCP based applications to work with SSL. Since IPSec is an IP layer protocol. It could be placed inside the kernel or as a separate machine out side the PC. Placing IPSec outside the machine called bump-in-the-wire (BITW). Placing IPSec inside the machine between two layers is called bump-in-the-stack (BITS).

Because IPSec resides in the IP layer, it allows multi-users to use one tunnel between two endpoints while SSL allow multi-users to have individual connections and different encryption key for each connection. The merit of using one tunnel for multi-users, as with IPSec, is to lower the overhead caused by establishing individual connections. The merit of using independent connection, as with SSL, is that each user has individual session. Consequently, compromising one connection doesn't compromise the other connections.

4.13 Time of Handshake Process

The time to establish a session is another element. Table 9 shows how much time is needed to establish a session for IPSec . The results are based on the use of a 2048 bit RSA key and 1536 bit DH.

Table 9: IPSec Handshake Time

Mode	Establishing
Main Mode (PSK)	97 msec
Aggressive Mode (PSK)	56 msec
Main Mode (RSA)	170 msec

Table 10 shows the time required for establishing SSL session. The results are based on the use of a 2048 bit RSA key and 768 bit DH. Using 1536 bit in DH has consumed 1648 msec in the Client Authentication. That is considered extremely slow when it is compared with 768 bit.

Table 10: SSL Handshake Time

Mode	Establishing
Server Authentication	41.7 msec
Client Authentication	74.8 msec
Server Authentication (Diffie-Hellman)	66.1 msec
Client Authentication (Diffie-Hellman)	118.6 msec

4.14 Session Resumption and Rekeying

The concept behind resuming a session is reducing the handshake process while maintaining a full security

level. As mentioned in the Residing Layer Section, the fact that each protocol works in a different layer has influenced the concept of session resumption. With SSL, the secure channel is bound to the application that required the secure service. Once the application is finished the secure channel shuts down.

When SSL session is resumed within the expiration, the client and the server exchanges the session ID (32 Byte) from the previous session in the clear and with it both side can identify the pre-master-key and produce the new session key needed to secure the information. Table 11 shows the time needed to resume a session in SSL.

Table 11: SSL Resumption

Mode	Time
Server Authentication	1.3 msec
Client Authentication	
Server Authentication (Diffie-Hellman)	
Client Authentication (Diffie-Hellman)	

IPSec's session resumption concept is totally different from SSL and is called *Rekeying*. Since IPSec is not bound to any application and has two different Phases, Rekeying mechanism is a bit complicated.

SA life time is responsible for how and when rekeying should occur. ISAKMP SA and IPSec SA are not obligated to have the same life time. Therefore, IPSec SA can be shorter in time than ISAKMP SA. What happens when ISAKMP SA expires before IPSec SA? This issue has not been standardized yet but IKEv2 draft has proposed a standard for it [15]. At the moment two different concepts are implemented: *Continuous-channel* and *Dangling SA*.

- Continuous-channel
  If ISAKMP SA expires then IPSec SA must be deleted. The reason is that ISAKMP SA is responsible for exchanging informational messages such as delete notification and *Dead Peer Detection* [16].

- Dangling SA
  Even with the expiration of ISAKMP SA, IPSec SA is valid until it reaches the validity time since ISAKMP SA has completed it mission of authentication and governed the negotiated secure channel.

Then, when IPSec SA rekeying is required, the rekeying process will depend on the ISAKMP SA status; If ISKAMP SA expired before IPSec SA (Figure 7),

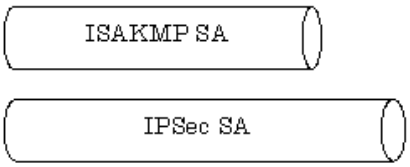

Figure 7: ISAKMP SA expires before IPSec SA

- In Continuous-channel, IPSec is deleted and a new Phase 1 and Phase 2 are negotiated.

- In Dangling SA, when IPSec SA asks for rekeying, then Phase 1 and Phase 2 are negotiated.

If IPSec SA expired before ISAKMP SA (Figure 8) and asked for rekeying in Continuous-channel or Dangling SA. Then, only Phase 2 is negotiated.

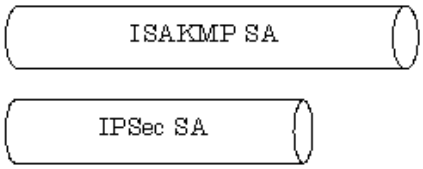

Figure 8: IPSec SA expires before ISAKMP SA

Table 12 shows the time needed to rekey an IPSec session within the expiration of ISAKMP SA.

Table 12: IPSec Session Rekeying for Phase 2

Mode	Time
Main Mode (PSK)	26 msec
Aggressive Mode (PSK)	
Main Mode (RSA)	

4.15 NAT Traversing

As mentioned in 4.6, SSL clients are not bound to a specific port, therefore the exisitence of NAT in the middle of the network doesn't affect the communications. IPSec clients on the other hand are bound to specific ports, thus having NAT or NAPT in the middle causes a problem for IPSec. Fortunately a solution has been proposed [17] and been used by many vendors.

4.16 Compression Algorithm

Compression is utilized by IPSec through a compression protocol called IPComp [18]. IPSec supports *DEFLATE*, *LZS* and *LZJH*. Unfortunately compression is used in a small range with SSL. Only *OpenSSL* [19] supports compression.

We have experimented the compression algorithm in two different environments and we had two opposite results. The first experiment was conducted on a 100 Mbps NIC and 100 Mbps Switching Hub. Compression Algorithm increased the throughput (see table 14 and 16).

The second experiment was conducted on a 1000 Mbps NIC and a 1000 Mbps Switching Hub. The Compression Algorithm decreased the throughput when used with most of the encryption algorithm except with 3DES where the throughput increased (see table 13 and 15).

The reason the first case has shown some improvement in throughput was the relevant speed between the compression, the encryption and the transfer. The network topology has caused a bottle neck. The throughput increased because the compression speed was faster than the transfer speed.

In the second case, the reason was the relation between the encryption speed and the compression. Most of the encryption algorithms are faster than the compression except for 3DES. Therefore, applying the compression to a higher speed encryption algorithm will cause the throughput to fall down.

4.17 Performance

The experiments were conducted on two machines with the following:

1. Red Hat 8 (kernel 2.4.20-20.8)
2. Pentium 4, 2.4 GHz, RAM 512 MB
3. NIC 100 Mbps and 1000 Mbps
4. 100 Mbps and 1000 Mbps Switching Hub
5. Super FreeS/WAN 1.99-8
6. Stunnel 3.26
7. Ethereal 0.9.16 [6]
8. Iperf 1.7.0 [7]

The results have shown a variation of throughput speed and CPU consumption.

[6] network protocol analyzer and time measuring tool
[7] throughput measuring tool

4.17.1 IPSec ESP-SHA-1

Table 13 shows the throughput on a 1000 Mbps network. CPU consumption varied between 94% and 97%. 3DES is the most consuming algorithm. When compression was applied the consumption was 98% regardless of algorithm or network status.

Table 13: 1000 Mbps Network (IPSec)

Algorithm	Throughput (Mbps)	
	No Compression	Compression
No Algorithm	427	N/A
DES	110	105
3DES	69.5	99.4
AES-128	156	104
BLOWFISH	123.5	105

Table 14 shows the throughput on a 100 Mbps network. The use of compression algorithm on a low bandwidth network has shown tremendous change in speed. CPU consumption was 76% for DES, 90% for 3DES and 57% for AES.

Table 14: 100 Mbps Network (IPSec)

Algorithm	Throughput (Mbps)	
	No Compression	Compression
No Algorithm	93.6	N/A
DES	89.3	104
3DES	70.7	101
AES-128	88.6	111

4.17.2 IPSec ESP-MD5

Table 15 shows the throughput on a 1000 Mbps network. CPU consumption varied between 87.7% and 93%.

Table 15: 1000 Mbps Network (IPSec)

Algorithm	Throughput (Mbps)	
	No Compression	Compression
No Algorithm	427	N/A
DES	137	113
3DES	75.7	107
AES-128	198	114
BLOWFISH	148	113

Table 16 shows the throughput on a 100 Mbps network. Compression algorithm has shown change in speed. CPU consumption was 57% for DES, 87% for 3DES and 44.3% for AES.

Table 16: 100 Mbps Network (IPSec)

Algorithm	Throughput (Mbps)	
	No Compression	Compression
No Algorithm	93.6	N/A
DES	89.8	122
3DES	78.8	115
AES-128	88.9	123

4.17.3 SSL

Our experiment includes the case of using exportable keys like EXP1024 and EXP512 RSA keys[8] , the throughput rate and CPU consumption don't show any change. For this reason, it will not be included in this paper.

Table 17 shows the result conducted on a 1000 Mbps network. CPU consumption varied between 86% and 93%.

Table 17: 1000 Mbps Network (SSL)

Algorithm	Throughput
No Algorithm	427 Mbps
3DES-EDE-CBC-SHA	86 Mbps
DES-CBC-SHA	152 Mbps
RC4-128-SHA	219 Mbps
RC4-128-MD5	246 Mbps
EXP-RC2-CBC-MD5	216 Mbps

Table 18 shows the result conducted on a 100 Mbps network. CPU consumption was 91.2% for 3DES, 58% for DES, 36.5% for RC4-SHA, 27.8% for RC4-MD5 and 87% for RC2

4.17.4 Transfer Speed of 100 MB of Data

Table 19 and 20 show the result of transferring a 100 MB of data over a 1000 Mbps network. Transferring a 100 MB over a 100 Mbps network has shown a slower transfer rate except with 3DES. 3DES didn't show a visible change in rate.

[8] US law requires using limited RSA encryption key length for exportable application. The export control was relaxed.

Table 18: 100 Mbps Network (SSL)

Algorithm	Throughput
No Algorithm	93.6 Mbps
3DES-EDE-CBC-SHA	75 Mbps
DES-CBC-SHA	90.3 Mbps
RC4-128-SHA	87.6 Mbps
RC4-128-MD5	90.2 Mbps
EXP-RC2-CBC-MD5	63.5 Mbps

1. IPSec
   As with IPSec we have tested the speed of various algorithm but we will only include the results of 3DES in various modes (see table 19).

Table 19: IPSec Transfer Speed

Algorithm	Time (Sec)
No Algorithm	2
3DES-SHA-1	12
3DES-MD5	10.5
3DES-SHA-1-DEFLATE	8.4
3DES-MD5-DEFLATE	7.8
AES-128-SHA-1	5.7
AES-128-SHA-1	4.5

2. SSL
   SSL has shown various types of speed. The best results could be achieved using RC4 with MD5. 3DES has shown a better performance than IPSec under the same circumstances (see table 20).

Table 20: SSL Transfer Speed

Algorithm	Time (Sec)
No Algorithm	2
3DES-EDE-CBC-SHA	9.8
DES-CBC-SHA	5.5
RC4-128-SHA	3.8
RC4-128-MD5	3.4
EXP-RC2-CBC-MD5	3.9

5 Conclusion

We presented the resemblance and the differences between IPSec and SSL. Each of the protocols has

unique properties. Choosing IPSec or SSL depends on the security needs. If a specific service is required and is supported by SSL, it is better to select SSL. If over all services or Gateway-to-Gateway communications are needed then IPSec is a good choice considering the following: IPSec uses a shorter form of HMAC than SSL, thus SSL data integrity is more secure. SSL is more compatible with firewall than IPSec, unless IPSec and Firewall are integrated in the same device. Unlike SSL, IPSec clients need a special IPSec software for remote access. In low bandwidth networks or dial-up networks using compression is beneficial, SSL doesn't support that. Pre-Shared scheme is easier to configure and doesn't require any PKI infrastructure,IPSec supports compression but unfortunately SSL doesn't support it. IPSec is capable of protecting wireless networks. In most cases IPSec doesn't interoperate well, so both sides of the connection are required to have the same vendor's devices (see table 21).

Table 21: IPSec vs. SSL

Function	IPSec	SSL
Configuration	hard	easy
Client Authentication	must	option
Pre-Shared Key	yes	no
Interoperability Problem	yes	no
TCP Application Support	all	some
UDP support	yes	no
Throughput Rate	high	high
Compression Support	yes	OpenSSL only
Handshake Time	slow	fast

References

[1] Sheila Frankel, *"Demystifying the IPSec Puzzle"*, Artec House Publisher, 2001

[2] Eric Rescorla, *"SSL and TLS, Designing and Building Secure Systems"*, Addison-Wesley, 3rd Printing, Aug. 2001.

[3] `www.freeswan.ca`

[4] `www.stunnel.org`

[5] S. Kent, R. Atkinson, *"IP Encapsulating Security Payload (ESP)"*, RFC 2406. Nov. 1998.

[6] S. Kent, R. Atkinson, *"IP Authentication Header"*, RFC 2402. Nov. 1998.

[7] D. Maughan M., Schertler M., Schneider J. Turner, *"Internet Security Association and Key Management Protocol (ISAKMP)"*, RFC 2408, Nov 1998.

[8] D. Harkins, D. Carrel, *"The Internet Key Exchange(IKE)"*, RFC 2409, Nov 1998.

[9] M. Thomas,J. Vilhuber, *"Kerberized Internet Negotiation of Keys (KINK)"*, Internet Draft. Jan 2003.

[10] C. Madson, R. Glenn, *"The Use of HMAC-SHA-96 within ESP and AH"*, RFC 2404, Nov. 1998.

[11] C. Madson, R. Glenn, *"The Use of HMAC-MD5-1-96 within ESP and AH"*, RFC 2403, Nov. 1998.

[12] Y. Sheffer, H. Krawczyk, Bernard Aboba, *"PIC, A Pre-IKE Credential Provisioning Protocol"*, Internet Draft, Oct. 2002

[13] S. Beaulieu, R. pereira, *"Extended Authentication within IKE (XAUTH)"*, Internet Draft, Oct. 2001.

[14] `www.ipa.go.jp/security/fy12/report/ipsec.html`

[15] Charlie Kaufman, *"Internet Key Exchange (IKEv2) Protocol"*, Internet Draft, Mar. 2004.

[16] G. Huang, S. Beaulieu, D. Rochefort, *"A Traffic-Based Method of Detecting Dead Internet Key Exchange (IKE) Peers"*, RFC 3407, Feb. 2004.

[17] T. Kivinen, A. Huttunen, B. Swander, V. Volpe, *"Negotiation of NAT-Traversal in the IKE"*, Internet Draft, Feb. 2004.

[18] A. Shacham, B. Monsour, R. Pereira. M. Thomas, *"IP Payload Compression Protocol (IPCOMP)"*, RFC 2393, Dec. 1998.

[19] `www.openssl.org`