An Overview of the SSL Protocol and Application to Virtual Private Networks

Rich Larsen

SANS GIAC/GSEC Practical Version 1.4b

September 29, 2003

An Overview of the SSL Protocol and Application to Virtual Private Networks

Rich Larsen

Abstract

Increasingly, businesses are deploying remote access and extranet solutions using the public Internet in lieu of centralized modem pools or leased WAN's. In order to ensure the security of these connections, organizations are relying on virtual private networks (VPN's) which allow secure access to a company's internal networks and systems from users connected to the Internet. Traditionally, these VPN solutions have employed the IPSEC suite of protocols for security. One of the major downsides of IPSEC-based VPN's is that client software needs to be loaded and administered on each device for it to access the VPN. This can be problematic for large-scale VPN deployments and/or extranet solutions where users may be employees of a different organization and only need temporary access. Recently, VPN solutions based on Secure Sockets Layer (SSL) protocol have emerged as a viable alternative to IPSEC. SSL is supported natively in all the commercial browser products and has proven to be a secure Internet protocol. In this paper, I plan to provide an overview of the history and protocol implementation details of SSL and also provide a comparison of the benefits and drawbacks of both IPSEC and SSL-based VPN's.

An Overview of the SSL Protocol and Application to Virtual Private Networks

Rich Larsen

Background

Increasingly businesses and individuals are utilizing the World-Wide Web to conduct their commerce. Recent legislation such as the Health Insurance Portability and Accountability Act (HIPPA) and the Gramm-Leach-Bliley Act (GLBA) has mandated that organizations ensure the security of their data wherever the transactions are conducted. Organizations must implement information security protocols which address issues such as:

- Protecting confidentiality of sensitive data
- Ensuring the integrity of data
- Authenticating participants in the transaction

Traditionally, businesses have deployed Virtual Private Networks (VPN's) to permit them to exchange data securely over the public Internet. Up until recently, nearly all VPN solutions have utilized the IPSEC (Secure IP) protocol which is detailed in RFC 2401 [1]. However, VPN's based on IPSEC have drawbacks which make them difficult to deploy as an enterprise-wide remote access or extranet solutions. VPN's based on Secure Sockets Layer (SSL) and its successor, Transport Layer Security (TLS) are rapidly emerging as a viable alternative for remote access VPN's. Infonetics Research, a leading IT research firm, estimates that the market for SSL-based VPN equipment will reach nearly $871 million by 2005 which would represent a cumulative annual growth rate of 143% [2]. In this paper, I plan to provide a brief history of the SSL protocol, describe the details of the SSL protocol (TLS is very similar) and examine the use of SSL/TLS as a feasible alternative for remote access VPN's.

History of SSL

A security protocol for HTTP protocol traffic was first proposed by the CommerceNet consortium in the early 1990's with a specification for secure HTTP (S-HTTP) [3] protocol which was implemented into a "secure" version of the Mosaic web browser. The basic idea was to encrypt and decrypt data at the endpoints of a connection. CommerceNet planned to charge customers to utilize its security protocol. The SSL protocol was developed by Netscape Communications in the early 1990's to support the deployment of its browser technology. Netscape realized that in order to expedite the adoption of electronic commerce, it needed to ensure the security of connections between parties over the Internet. Netscape was more successful than CommerceNet because it freely distributed its technology with the browser and made the protocol details available for public download and scrutiny.

Netscape could have also chosen to incorporate this technology as an add-in for its browser which it would make available to customers for a fee. However, this approach would result in a proprietary solution and would not provide a unified solution from which non-HTTP-based applications could benefit. Therefore, Netscape decided to develop an application-independent security protocol. The new protocol would sit on the top of the existing transport layer protocol (i.e., TCP or layer 4 in the OSI network stack) and provide an open interface to any application. The specific implementation details of the protocol would be irrelevant to the application developers; they would simply replace their TCP socket calls with SSL calls. Netscape also decided to make the details of the protocol public so it could be subjected to the scrutiny of the security community. The first public release of SSL protocol specification (version 2.0) was issued in November 1994 [4]. Due to several documented shortcomings with version 2.0, Netscape subsequently released version 3.0 of SSL in March, 1996 [5]. SSL version 3.0 is the most widely used version of SSL and is the basis for TLS version 1.0 which is detailed in RFC 2226 [6]. SSL and TLS are fairly similar; however, TLS requires support for additional encryption algorithms. Thus, TLS 1.0 and SSL 3.0 are not compatible. However, TLS does provide a means for the server to "back down" and support SSL 3.0 clients.

SSL has become the protocol of choice for securing Internet commerce primarily because of its support in all of the mainstream browser products and the fact that it is has held up to the scrutiny of the Internet security community. Schneier and Wagner performed a in-depth analysis of SSL 3.0 and conclude that the protocol " provides excellent security against eavesdropping and passive attacks and can be made fairly secure against active attacks with proper implementation" [7].

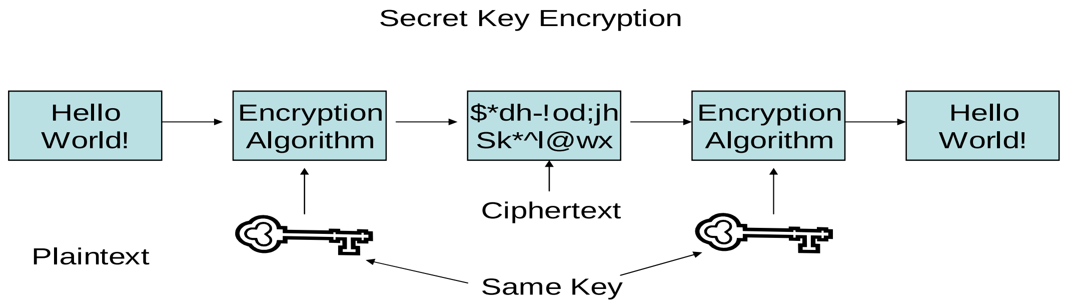
The SSL Protocol Details

According to the protocol specification [5] , the primary goal of SSL is to ensure data privacy and reliability between parties while communicating over unsecured networks. SSL accomplishes this by providing for:

- Privacy of the connection.
- Authenticity of the peers to a connection.
- Integrity of the connection.

SSL uses public key encryption to secure the exchange of shared secret keys which are used to encrypt and decrypt the actual data transmission. Secret key cryptography (also called symmetric key cryptography) utilizes the same key for the sender and receiver to encrypt and decrypt the data. This is the most intuitive and oldest form of cryptography. The key is a string of binary digits (0's and 1's) which are combined with the plaintext message in a predetermined manner to produce encrypted ciphertext. The ciphertext can then be transmitted securely over the Internet (Figure 1). As the length of the key increases, the security of the algorithm increases. Typical symmetric key sizes today are 56, 128 or 168 bits long. The mathematical operations used for combining the key and data are known as the "encryption algorithm".

Figure 1



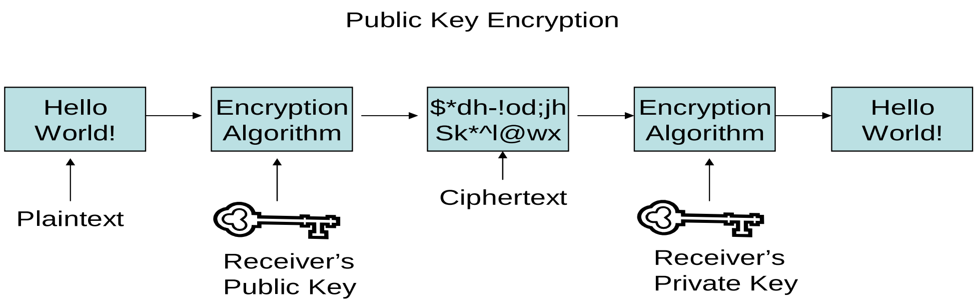
Examples of secret key algorithms include Data Encryption Standard (DES), RC4, Triple DES (3-DES), and Advanced Encryption Standard (AES). The basic premise behind symmetric key algorithms is the alternating use of “diffusion and confusion”. Confusion ensures that any relationship between the plaintext and ciphertext is minimized. There should be no feasible means to determine the plaintext by examining copies of ciphertext. Diffusion further helps to dissipate the statistical structure of the plaintext over the bits in the ciphertext.

A major roadblock to deploying symmetric key algorithms is that both parties must posses the common secret key before they can exchange encrypted messages. Distribution of that key to all parties in a transaction is a classical “catch-22” scenario. Since the encryption algorithms are widely published, knowledge of the secret key allows any attacker to decrypt successive messages exchanged between the parties. In addition, the attacker could also encrypt bogus messages and send them by masquerading as one of the parties. Thus, provisions must be made for a secure means of key distribution to all parties. In organizations such as the military where there is a clearly defined list of participants, the keys could be distributed by some out-of-band method such as the trusted human courier with a briefcase chained to his wrist. However, in the realm of e-commerce, all the parties to a transaction may not be known at the time that the system is deployed so a priori key distribution is not always feasible.

Public key encryption is often referred to as asymmetric encryption because it utilizes different but related keys for encryption and decryption.
Each party must possess two distinct keys referred to as their public and private keys. These keys are related since they are usually generated as a pair and once generated it is mathematically infeasible to determine the private key from the

public key. In order to encrypt a message, the sender would obtain the receiver's public key and apply some public encryption algorithm to the plaintext and the key to produce the ciphertext (Figure 2).

Figure 2



Since the public key is designed to be made publicly available, it could be posted to a website or emailed to the parties engaged in the transaction without concern about security during distribution. However, only the holder of the private key is able to decrypt messages encrypted with his public key. It is assumed that the private key is never divulged by the owner. An example of a public key algorithm is RSA which is named for its creators: Rivest, Shamir and Adelman [8].

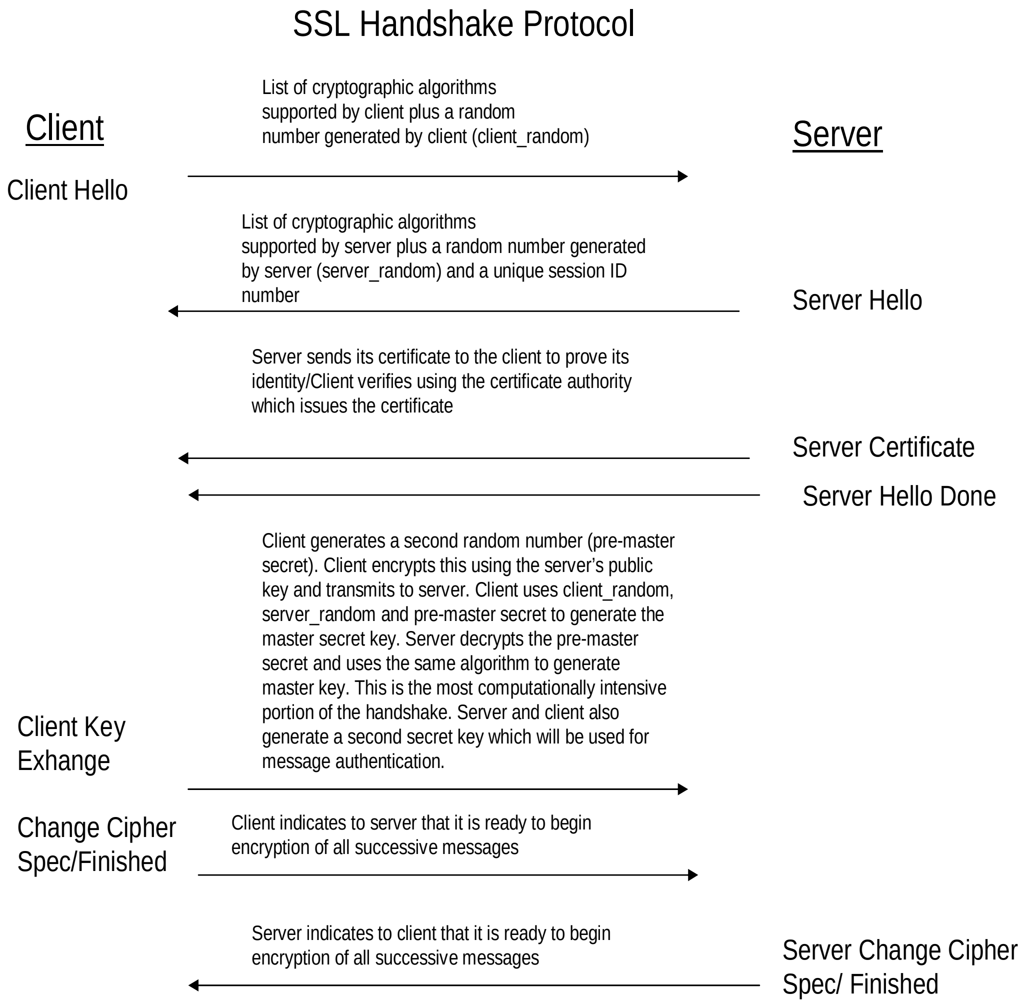
Another feature of public key cryptography is the use of digital signature. A sender could digitally "sign" a message using his private key which could be verified by the receiver using the sender's public key. This approach is used for message authentication since the sender is the only person who should have possession of the private key. For example, a person could digitally sign a contract and transmit it electronically to the bank which could verify the signature. Digital signature is not possible with symmetric key cryptography since all the parties possess the same key there is no way to authenticate an individual user.

Since public key cryptography appears to solve the issue of secure key distribution, one may ask why we continue to use secret key encryption. Decrypting a message with public key algorithms is on the order of 1000 times slower than using secret key. Secret key technology utilizes bit manipulations and/or combinations which are easily performed using basic computer hardware such as shift registers. However, public key utilizes algorithms which require complex exponentiation and thus more sophisticated hardware and considerable CPU cycles. In addition, as the key size of a public key algorithm is increased the number of CPU cycles required for decryption increases exponentially whereas with secret key algorithms, the key size has relatively little impact on the performance of the algorithm. In addition, typical key sizes for public key algorithms are on the order of 512, 1024 or 2048 bits which is much greater than for secret key algorithms.

One way to capitalize on the strength of both algorithms is to utilize public key encryption to distribute the secret keys. In other words, the secret key is the “message” which is in encrypted using the public key algorithm. Once the receiver has decrypted the message, subsequent data transfers take place using secret key algorithms which have higher performance. A variation on this theme is when the parties exchange some information over a secure public key channel which can be used by each party to independently generate the identical secret key which is subsequently used to encrypt all data transfers. The key that is generated is only used for that particular session. If these two parties subsequently communicate with each other, a new key is generated. This is the technique that is used by SSL. A good overview of SSL and TLS is found in [9].

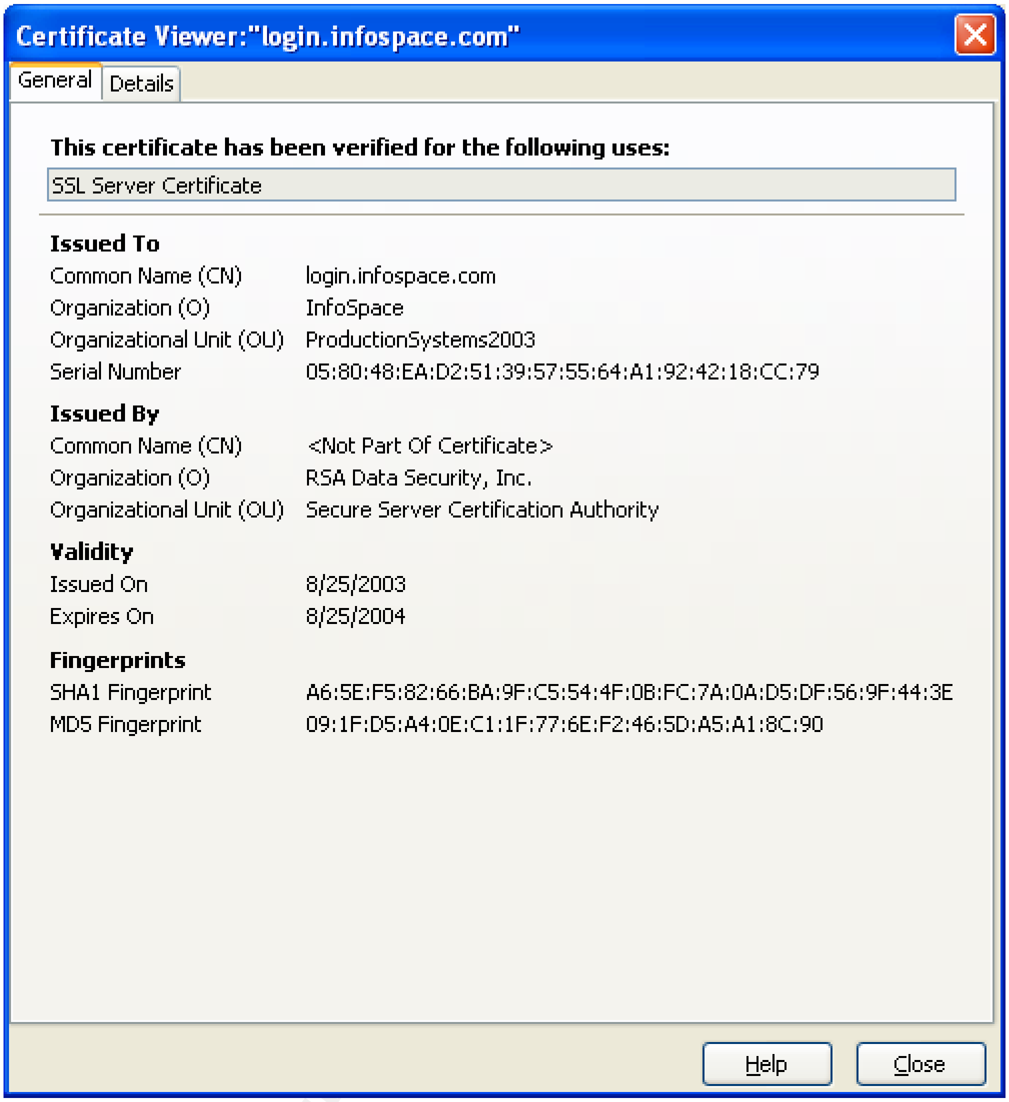
The SSL protocol is comprised of several sub-protocols with the two major ones being the handshake and the record protocols. The handshake protocol is used to allow the client and server “to authenticate each other and negotiate an encryption algorithm and cryptographic keys before the application protocol transmits its first byte of data” [5]. During the data transfer phase, the record protocol utilizes the keys generated during the handshake protocol to encrypt and decrypt subsequent data exchanges. Figure 3 shows the exchanges between the client and server which take place during the handshake phase. The client initiates a new handshake phase by sending a “HELLO” message to the server on a particular TCP port (usually port 443 for secure HTTP). The message contains the list of cipher specs (encryption protocols and key sizes) and compression protocols that the client’s browser understands as well as a random number generated by the client (client_random). The client_random is composed of a 32-bit UNIX timestamp value plus a 28 byte pseudo-random number generated by the client’s browser. The server responds to the client by sending a HELLO message with the cipher specification it has selected along with a random value (server_random) with the same format as client_random and the server’s certificate. The certificate is nothing more than the server’s identity and public key which has been confirmed by a “certificate authority” (CA). The CA has personally verified the identity of this server and has attached its own digital “fingerprint” to the certificate. Figure 4 provides an example of some of the information that is contained in a certificate. Notice the fingerprints are provided using two different algorithms since not all browsers support all algorithms. Most browsers already contain a list of “trusted” CA’s and corresponding public keys for large certification authorities such as Verisign or Entrust.

Figure 3


SSL Handshake Protocol
Client
Server
List of cryptographic algorithms supported by client plus a random number generated by client (client_random)
Client Hello
List of cryptographic algorithms supported by server plus a random number generated by server (server_random) and a unique session ID number
Server Hello
Server sends its certificate to the client to prove its identity/Client verifies using the certificate authority which issues the certificate
Server Certificate
Server Hello Done
Client generates a second random number (pre-master secret). Client encrypts this using the server's public key and transmits to server. Client uses client_random, server_random and pre-master secret to generate the master secret key. Server decrypts the pre-master secret and uses the same algorithm to generate master key. This is the most computationally intensive portion of the handshake. Server and client also generate a second secret key which will be used for message authentication.
Client Key Exhange
Change Cipher Spec/Finished
Client indicates to server that it is ready to begin encryption of all successive messages
Server indicates to client that it is ready to begin encryption of all successive messages
Server Change Cipher Spec/ Finished

If the CA is unknown the client may also need to obtain the certificate for the CA and install it into the browser. CA Certificates can be downloaded from the web, however, this is risky since another party could masquerade their certificate as belonging to the CA. The client uses the CA's public key to independently calculate the fingerprint on the certificate and compares it to what was received to ensure that the certificate has not been modified. The server also provides the client with a session_id number. Since the handshake phase of SSL is computationally intensive, the client and server also exchange a session_id number so they can rapidly restore the session in the event that the TCP connection is dropped. In fact SSL provides for an abbreviated handshake resumption protocol which is not as computationally intensive as the initial handshake. Obviously, there is always risk involved with reusing the same session_id and key so most browsers will renegotiate a new session_id and key periodically. However, the client may connect to the server several times during a session since generally each request for a new page requires at least one TCP connection by the client.

Figure 4 – A typical digital certificate



Certificate Viewer:"login.infospace.com"

This certificate has been verified for the following uses:

SSL Server Certificate

Issued To	
Common Name (CN)	login.infospace.com
Organization (O)	InfoSpace
Organizational Unit (OU)	ProductionSystems2003
Serial Number	05:80:48:EA:D2:51:39:57:55:64:A1:92:42:18:CC:79
Issued By	
Common Name (CN)	<Not Part Of Certificate>
Organization (O)	RSA Data Security, Inc.
Organizational Unit (OU)	Secure Server Certification Authority
Validity	
Issued On	8/25/2003
Expires On	8/25/2004
Fingerprints	
SHA1 Fingerprint	A6:5E:F5:82:66:BA:9F:C5:54:4F:0B:FC:7A:0A:D5:DF:56:9F:44:3E
MD5 Fingerprint	09:1F:D5:A4:0E:C1:1F:77:6E:F2:46:5D:A5:A1:8C:90

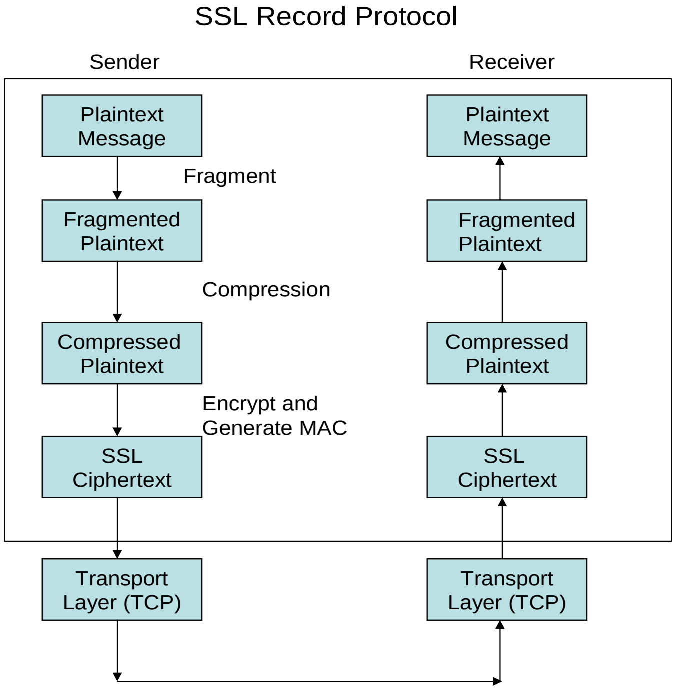
Once the client has verified the identity of the server, it has the option of transmitting its own certificate to the server and performing mutual authentication. This step is not required by SSL because many clients do not possess individual digital certificates. However, organizations who wish to ensure the authentication of clients can generate and distribute certificates to each client and require the server to authenticate the client. Of course the certificate would also need to be loaded in the browser before the client can connect so the user would not be able to gain access from a random computer that does not possess the certificate. After the authentication has been performed the client generates a second random number (pre-master secret) and encrypts this information with the server's public key and transmits it back to the server. The client also uses an algorithm which takes the client_random, server_random and pre-master secret as input and generates the master (secret) key. This is the key that will be used for encrypting data transmissions. The client also generates a second secret key which is used for authenticating the subsequent transmissions using a message authentication code (MAC). Once the client has completed its tasks successfully it transmits a "Change Cipher Spec" message (another sub-protocol of SSL) to the server which indicates that it intends to encrypt future data transmissions with the calculated secret key. The server performs the exact same calculations as the client and transmits a Server Change_Ciper Spec message to the client. At this point the handshake protocol is complete and server and client begin to exchange data using the SSL record protocol.

The record protocol is the actual workhorse of SSL. It is responsible for delivering and receiving the encrypted SSL frames to and from the network transport layer (i.e., TCP). A diagram of the SSL record protocol is shown in Figure 5. The record protocol takes a stream of data from the application layer and fragments it into chunks of no more than about 16K bytes each. These chunks are then compressed (optionally) and encrypted using the secret session key. Finally the record layer creates a packet containing the following:

- Content type- defines the application that must be used to process the data (e.g., IMAP, Telnet, FTP)
- Protocol Version Number- usually “3” for SSL version 3.0
- Length of the packet in bytes
- Payload (encrypted and optionally compressed)
- Message Authentication Code (MAC)

This frame is then passed down to the transport layer in the network stack for delivery to the receiver. At the receiver’s end, the frame is passed from the transport layer to the SSL record layer which then performs the decryption and expansion and passes the data back up to the correct application based on the value of the content type field. The receiver also verifies the MAC to ensure that the data has not been modified in transit.

Figure 5



In summary, SSL is the most widely used security protocol on the Internet. Originally developed by Netscape Communications, SSL provides for:

- Data Confidentiality- using symmetric key encryption protocols based on a secret “session” key which is renegotiated periodically.
- End point authentication- SSL requires the client to verify the identity of the server and also provides for optional mutual authentication
- Message integrity- SSL checks to ensure that data transferred between the client and the server has not been modified.

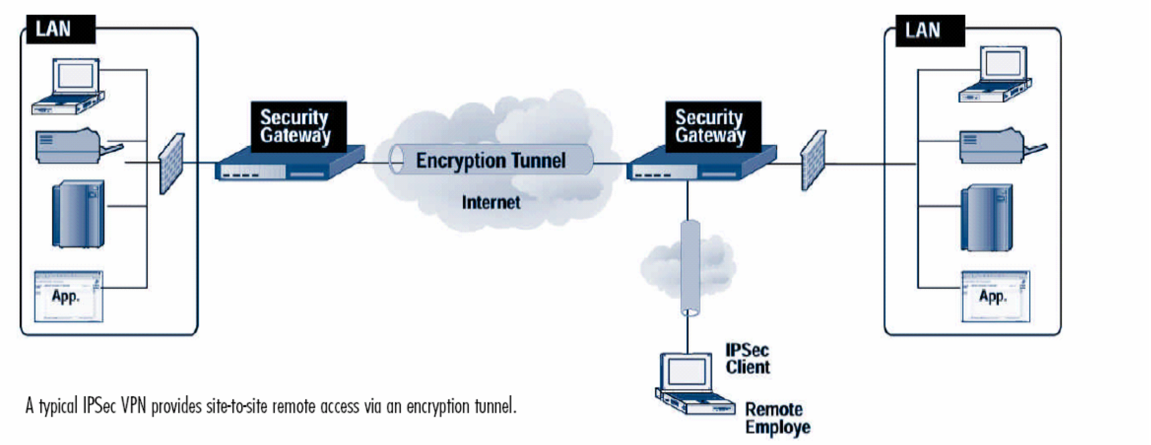
SSL is application-independent since any application can utilize its security without understanding the implementation details. SSL sits between the transport and application layers in the OSI network stack and is often referred to as “transport layer security” (TLS) which also happens to be the name of the Internet standard successor to SSL. In addition, SSL is inherently supported by all of the major web browser products. This implies that generally no additional software needs to be loaded onto the client in order to support SSL connections. These two features are key points for the following discussion on application of SSL to virtual private networking.

Application of SSL to Remote Access Virtual Private Networking

Until recently, SSL was primarily used to secure transactions taking place within an Internet browser application using HTTP protocol. Most of us have probably used SSL, signified by the clasped padlock symbol in the lower corner of the browser window. Since SSL is positioned between the application and transport layer, it is ideal for securing application-layer protocols such as HTTP, FTP, email, etc. However, it is also quickly becoming a very viable option for companies wishing to deploy Virtual Private Networking (VPN) solutions.

It is not uncommon these days to see employees connecting remotely to their corporate Intranet from airports or coffee shops. According to Siemens AG, nearly 33 million workers are now considered “mobile” (meaning they work remotely at least half time) and this number is expected to grow to nearly 50 million by 2005. Organizations have responded to this desire for mobility by deploying VPN’s to extend the reach of their corporate Intranet to mobile workers. Typically these VPN’s have utilized IPSEC to provide security for the connection. IPSEC is actually a suite of security protocols which provide data confidentiality, authentication of endpoints and anti-replay protection. SSL also provides many of these same benefits. However, unlike SSL, IPSEC works by establishing a secure network-layer “tunnel” between the endpoints through which all traffic is funneled. Figure 6, which is taken from [10], shows a typical deployment scenario for an IPSEC-based VPN. In this case an organization has remote sites or offices (designated by the LAN on the left-hand side). Instead of purchasing a private WAN link between the sites, the organization has chosen to route its inter-site traffic over the public Internet. The security “gateways” are hardware devices which sits at the border between the organizations’s Intranet and the public Internet (usually outside of the corporate firewall). In this case the company has LANs at both the main site (right side of the diagram) and the remote site. All the users at the remote site require access to applications which are housed on servers connected to the LAN at the main site. The company also has mobile workers who also require access to the Intranet.

Figure 6



A typical IPSec VPN provides site-to-site remote access via an encryption tunnel.

In order to create the VPN, the security gateways must establish an IPSEC tunnel. The details of how this is done are provided in [1] but essentially a security "association" is established between the sites using a handshake protocol. The security association defines the parameters (symmetric keys, etc.) that will be used for this connection. This information is stored in a security association "database" which is maintained by each gateway. Since there could be multiple sites connecting to the main site, each site also establishes a security association with the main site. Once the associations are established, IPSEC (like SSL) utilizes secret key algorithms such as DES, 3DES or AES to perform the encryption/decryption of the data. From this point on the two networks appear as one network. There should be no indication to a worker in the branch office that his traffic is being routed across the Internet other than some additional delay due to Internet congestion. She should be able to access all of the data and applications that she is permitted to access. This makes IPSEC an attractive solution for companies with multiple sites who wish to consolidate their enterprise applications at one location.

The scenario for the remote employee is similar to what was described previously for the branch office. However, instead of a hardware solution, the user typically has had some client software loaded on his computer by the company's IT department. This software provides the same functionality as the hardware solution with regards to performing the handshake, establishing the security association and initiating the IPSEC tunnel. However, in this case the tunnel is connecting a single computer to the corporate intranet. The client software must be configured specifically to negotiate with the security gateway at the home office. In addition, the client software is typically configured to provide some proof of authentication of the remote user. This can be accomplished using a username/password combination, a client digital certificate or Secure-ID that that has been issued to the client. Of course, this scenario assumes that the IT department has been able to gain access to the mobile user's computer to install and configure the client software or that the worker was able to do this himself using instructions provided by the IT group. This is not always a valid assumption especially if the person is 100% remote or occasionally works from home and utilizes a personally owned computer. In addition, there may be employees from partner organizations who require temporary access to the corporate intranet for a particular project. The company may not wish to provide these people with full access to their internal network. Unfortunately, many of the commercial IPSEC solutions do not provide highly granular access management capabilities.

As mentioned previously, IPSEC VPN's operate by providing a virtual extension of the corporate intranet to remote sites or workers. Once connected, the remote node becomes just like any other node on the corporate intranet and is usually given the same level of access to the network as employees who are physically located at the home office. Users who authenticate through the VPN are often given access to the corporate intranet which tends to be less well-protected against attacks if you assume the "candy" model of security ("crunchy on the outside and chewy on the inside"). Unless the company provides each remote user with a managed laptop, it is difficult to ensure that their computer has been patched or is running updated virus detection software. In addition, the user may share the computer with other members of his family, roommates, etc. who use the computer for general Internet access. This means that the computer is potentially compromised. Once the compromised computer is connected to the corporate Intranet via an IPSEC tunnel, it could potentially infect other computers on the company's Intranet.

In response to the drawbacks of IPSEC-based VPN's discussed above, vendors began to evaluate alternative solutions. SSL began to emerge as an alternative to IPSEC for remote access VPN's primarily for two reasons: its proven security and the fact that it is already incorporated into every major commercial browser product (no need for additional client software). These benefits have propelled SSL-based solutions into the fastest growing segment of the VPN market. John Girard, VP and Director of Research for Gartner Group predicts that "*by YE04, 60 percent of corporate users will regularly use thin-client VPNs rather than full, fat-client VPNs for access to business data*" [11]. Note that this is only referring to the remote access type VPN which supports individual mobile users not the intersite VPN scenario. IPSEC is still clearly superior for this purpose and will continue to dominate this segment of the market. SSL is not a full replacement for IPSEC VPN's but more of a complementary solution in the area of remote access VPN's. This is evident in that some of the vendors who produce SSL VPN solutions also continue to offer IPSEC-based VPN products. In the next section of this report, I will attempt to compare and contrast the IPSEC and SSL VPN solutions on a number of criteria including:

- Client software requirements
- Security (i.e., encryption and authentication)
- Application Support
- Performance/Scalability

Some of this material is obtained from a comparison of IPSEC and SSL provided in [12]. At the end of this discussion, I provide a tabular summary of results of this comparison (Table 1).

Client Software Requirements

SSL-based VPN solutions clearly dominate IPSEC with regards to alleviating the need for installation, management and support of client software on individual computers. Typically, enterprises consist of a heterogeneous mix of hardware and software. Unless the central IT organization controls all the hardware it may be difficult to enforce specific architectural standards across the organization. IPSEC solutions require client software be loaded and configured on each remote computer prior to access. In addition, IPSEC client software from a particular vendor may not interoperate with gateways from a different vendor. This is problematic for providing access to external partners whose company may use a different vendor's solution. In addition, vendors may not provide a VPN client which supports all operating systems. Furthermore, IPSEC has documented issues with Network Address Translation (NAT). Users who are connecting from behind NAT-based firewall products may have difficulty since not all NAT routers support IPSEC or may require configuration changes to work. This imposes a heavy burden on IT support personnel who will need to troubleshoot specific problems for each user. SSL, on the other hand, is already incorporated into all mainstream commercial web browsers alleviating the need to install additional software. In addition, most vendors of SSL-based VPN's provide additional functionality through signed plugins which can be deployed to the clients automatically through the browser.

Security

Both SSL and IPSEC support essentially the same suite of encryption standards: 40/128-bit RC4, 56-bit DES, 168-bit Triple-DES. In addition, IPSEC also supports AES which is not supported by SSL but is supported by TLS [13]. One benefit of IPSEC is that each connection is governed by a security association. With IPSEC it is easier to mandate use of a particular encryption specification by configuring the gateway and client software accordingly. With SSL this configuration occurs within the browser and not all browsers support all encryption standards. In addition, browsers may negotiate the encryption strength downward to support the least common denominator between the client and server. For example, some browsers may back down to 40-bit RC4 which is not terribly secure. Of course, the administrator can specify a particular minimum level of security on the gateway but this increases the risk that some clients may be unable to connect due to their particular browser settings or version.

Regarding authentication, both IPSEC and SSL provide for mutual authentication as part of the specification. By virtue of having the IPSEC client software installed on the computer, there is some inherent level of authentication of the device. In addition, the IPSEC client can be preconfigured to accept only a particular server's certificate or with a username/ password that is specific to that organization. By default, SSL prompts the user when a new certificate is being sent from the server and the user has the option to accept or reject the certificate. Most untrained users will probably accept the certificate if they think that will allow them to get to their information more quickly. However, this may result in a "man-in-the-middle" attack. Obviously, the server certificate can be preinstalled into the browser but this defeats some of the benefit of SSL in that it does not require additional configuration of the client.

However, one potential security risk with SSL results from untrained users who do not properly close their browser application when they have completed their session. This could be a significant risk in the case of access from a shared public kiosk. If the browser is not closed it will still retain the credentials from the previous user and another user of the same computer may be able to gain access to the company's network. Several of the vendors provide solutions for this involving inactivity timeouts on the browser however, this is only a partial solution since there is still always a period of time when the vulnerability exists. In addition, connecting from uncontrolled public access computers is risky since the computer could have a keystroke recorder installed which could be used to replicate a user's session.

Application Support

Since IPSEC operates at the network layer it can generally support any application that runs on an IP-based network. As discussed previously, IPSEC works like an extension of the corporate intranet so that any application which is available on the corporate intranet should theoretically be able to operate across an IPSEC tunnel without modification. This makes IPSEC a more seamless solution for users who need to access multiple types of client-server applications.

SSL was originally developed to support browser-based applications. However, it also supports several of the typical Internet-based applications such as email, file transfer and chat. However, many client-server applications would need to be rewritten ("Webify'ed") in order to work over SSL. Several of the vendors provide application-specific proxies for some common applications. The web browser on the client communicates with the proxy gateway which performs the translation to the backend server. However, this can have an impact on performance and not all applications are supported. If the company uses a wide suite of custom developed client-server type applications, then the SSL solution may not be practicable.

Scalability/Performance

IPSEC solutions tend to be more scalable as the number of remote sessions is increased. Many IPSEC solutions are capable of supporting up to 5,000 concurrent tunnels with a singular hardware gateway. In addition, most of the vendors provide for modular support to allow companies to scale up their hardware as the number of remote users is increased. The downside of IPSEC scalability is that an increase in number of users requires the IT department to spend more time configuring and supporting the IPSEC client software. Thus, the IPSEC solution may not be practically scalable. The SSL protocol is more processing-intensive than IPSEC (due to the use of public key algorithms) and there is typically a performance penalty associated with increasing the number of sessions beyond a preset limit. However, several vendors offer proprietary SSL acceleration solutions which offload much of the processing to other hardware devices which have been optimized for SSL processing. Of course, this tends to increase the cost of the solution and there is a tradeoff that needs to be made as the number of remote users is increased.

Since both IPSEC and SSL require encryption and decryption of data, performance is typically slower than it would be running the application with no security. Applications which are highly sensitive to delay and jitter (e.g., voice or video streaming applications) may not be well-suited for these environments. In addition, SSL is a session-oriented protocol meaning that the security parameters are negotiated at the beginning of each session regardless of the number of individual connections which occur during a session. Applications which are not session-aware (such as LAN-based client/server applications) may cause performance problems with SSL because they will need to reestablish the security parameters more frequently and this is the most processing-intensive phase of the protocol.

Table 1- Comparison of IPSEC and SSL-based VPN solutions

	IPSEC	SSL
Client Software Requirements	**Very high**- Requires pre-configured vendor-specific client application to be installed on each client. Can potentially be distributed to clients from central server. Users cannot connect from computers which have not been configured (lower mobility). High overhead for IT support team.	**Low**- Support provided with most browsers. However, older browsers may not support more secure encryption protocols and certificates may need to be loaded into browser. In general, greater support for mobility of users.
Security	**High**- security is based on preconfigured client application which specifies security agreement between the client and server. Supports the following encryption specs: • 40/128-bit RC4 • 56-bit DES • 112/168-bit Triple DES • 128/192/256-bit AES	**Moderate**- protocol itself is secure but may allow for access from uncontrolled computers which may be compromised; users do not close browser sessions. Supports same encryption protocols as IPSEC (except for AES) but not all browsers support all specs. may drop back to lowest common denominator.
Application Support	**Very high** for applications- nearly any app that works over IP will work over IPSEC. This includes most custom-developed applications.	**Moderate**- supports nearly all standard Internet applications inherently (email, FTP, HTTP). Access to legacy (non-web) applications may be problematic although vendors provide proxy solutions for some apps.
Performance	**Good-** slower than unsecured connection due to encryption, key management, etc.	**Moderate-** significantly slower than unsecured web connection; can use hardware accelerators to improve performance
Scalability	**Moderate**- adding users more complex than SSL; each user needs to have client software configured. However, hardware can generally support up to 5-10K simultaneous connections	**Very high-** adding users require little or no administrative overhead; no configuration of client device. Hardware can support > 10K simultaneous sessions.

Summary and Conclusions

Users are demanding ubiquitous access to their data and applications. In order to reduce costs, organizations are migrating away from centralized dial-in modem banks and private WAN links and are utilizing the public Internet more frequently for remote connectivity. In addition, organizations are deploying extranets to provide remote access to their partners. In order to ensure security of these connections, organizations are deploying virtual private network (VPN) technology. Traditionally, VPN's have utilized IPSEC which is a suite of security protocols used to establish secure "tunnels" between sites or between a remote user and the company network. IPSEC has proven to be secure and reliable but it requires client software to be loaded and configured on each individual device making it more difficult to manage for large-scale remote access deployment. In addition, mobile users must carry a configured laptop in order to access the network. This can impose a high administrative and cost burden on the organization's IT staff.

SSL is also a proven Internet security protocol. Developed by Netscape Corp., SSL has been used to secure web browser sessions using the HTTPS protocol. Like IPSEC, SSL provides for privacy, authentication and data integrity between the client and server. Unlike IPSEC, SSL is incorporated into all of the commercial browser products so it does not generally require additional client software. In addition, there are specific "secure" versions of web, email and file transfer protocols that utilize SSL natively. Recently, vendors have begun to release VPN solutions that are based on SSL instead of IPSEC since it alleviates many of the issues with client software management. Many applications are being developed or rewritten to be "web-enabled" which makes SSL an ideal choice in that environment. As a result, Gartner Group is predicting that by the end of 2004, a majority of remote users will utilize VPN's based on SSL. SSL-based VPN's provide several advantages over IPSEC specifically in the area of reduced management overhead and scalability. However, IPSEC is still dominant as an intersite VPN solution since SSL is not designed for this scenario. SSL also falls short in the area of application support since some legacy client-server applications may not operate over an SSL connection.

IPSEC and SSL should be considered complementary solutions to the problem of providing secure, standards-based and ubiquitous remote access to the corporate intranet for mobile users and partners. Both solutions possess benefits and limitations depending on the particular circumstances. Each solution needs to be evaluated independently based on requirements for a particular implementation. In this paper, I have provided a number of criteria by which a solution can be evaluated. However, this is not meant to be an all-inclusive list and many vendors of VPN solutions provide additional criteria. IPSEC and SSL will continue to exist in harmony and bring value to the organization's specific remote access requirements.

References

[1] Kent, S. and Atkinson, R., "Security Architecture for the Internet Protocol," IETF November 1998,http://www.ietf.org/rfc/rfc2401.txt

[2] Wilson, J., "Application-Layer (SSL) VPN Gateways Begin Shipping; Market to Surge to $871 Mill by 2005," Infonetics Research,May 292003, http://www.infonetics.com/resources/press_release_vpn_ssl.htm

[3] Shostack, A., "An Overview of SHTTP," May 1995,http://www.homeport.org/~adam/shttp.html

[4] Hickman, K., "SSL Protocol Specification Version 2.0," Netscape Communications,November1994, http://wp.netscape.com/eng/security/SSL_2.html

[5] Freier, A. O., Karlton, P., and Kocher, P. C., "The SSL Protocol Specification Version 3.0," Netscape Communications,November1996, http://wp.netscape.com/eng/ssl3/ssl-toc.html

[6] Dierks, T. and Allen, C., "TLS Protocol Specification Version 1.0," IETF,January1999, http://www.ietf.org/rfc/rfc2246.txt?number=2246

[7] Wagner, D. and Shneier, B., "Analysis of the SSL 3.0 Protocol," Counterpane Systems April 15 1997,http://www.counterpane.com/ssl-revised.pdf

[8] Rivest, R., Shamir, A., and Adelman, L., "On Digital Signatures and Public Key Cryptography," MIT, Cambridge, MA MIT Computer Science Labs Technical Memorandum 82, April 1977,

[9] Opplinger, R., *Internet and Intranet Security*. Norwood, MA: Artech House, 2002.

[10] Aventail-Corp., "Comparing Remote Access Options: IPSEC VPNs vs. SSL VPNs," Aventail Corp., Seattle, WA 2002,http://www.aventail.com/downloads/pdfs/IPSECvsSSL_WP.pdf

[11] Girard, J., "Mobile Security Exposures, Trends and Remedies," Gartner Group SPA-18-6438, 2002 (Article only available to Gartner subscribers)

[12] Nortel_Networks, "IPSEC and SSL: Complimentary Solutions," Nortel Networks, Research Triangle Park, NC NN102260-110802, 2002,http://www.nortelnetworks.com/solutions/ip_vpn/collateral/nn102260-110802.pdf

[13] Chown, P., "AES CipherSuites for Transport Layer Security (RFC 3268)," IETF,June2002, http://www.faqs.org/rfcs/rfc3268.html