Overview of CSS SSL

Secure Sockets Layer (SSL) is an application-level protocol that provides encryption technology for the Internet, ensuring secure transactions such as the transmission of credit card numbers for e-commerce Web sites. SSL provides the secure transaction of data between a client and a server through a combination of privacy, authentication, and data integrity. SSL relies upon certificates, private-public key exchange pairs, and Diffie-Hellman key agreement parameters for this level of security.

This chapter contains the following major sections:

- SSL Cryptography Overview
- Overview of the SSL Module Functions in the CSS

SSL Cryptography Overview

The CSS uses the SSL Acceleration Module and a special set of SSL commands to perform the SSL cryptographic functions between a client and a server. The SSL functions include client and server authentication, private-key and public-key generation, certificate management, and data packet encryption and decryption.

The SSL module supports SSL version 3.0 and Transport Layer Security (TLS) version 1.0. The module understands and accepts an SSL version 2.0 ClientHello message to allow dual version clients to communicate with the CSS through the SSL module. In this case, the client indicates an SSL version of 3.0 in the version 2.0 ClientHello, which informs the SSL module that the client can support SSL version 3.0. The SSL module returns a version 3.0 ServerHello message.

Note Although there are very few clients on the market today that support only SSL version 2.0, the SSL module will be unable to pass network traffic if the client supports only version 2.0.

A typical SSL session with the SSL module requires encryption ciphers to establish and maintain the secure connection. Cipher suites provide the cryptographic algorithms required by the SSL module to perform key exchange, authentication, and Message Authentication Code (MAC). See the "Specifying Cipher Suites" section in Chapter 3, Configuring SSL Certificates and Keys for details about the supported cipher suites.

This section provides an overview on SSL cryptography as implemented through the SSL module in the CSS. It covers:

- SSL Public Key Infrastructure Overview
- SSL Module Cryptography Capabilities

SSL Public Key Infrastructure Overview

SSL provides authentication, encryption, and data integrity in a Public Key Infrastructure (PKI). PKI is a set of policies and procedures to establish a secure information exchange between devices. Three fundamental elements characterize the PKIs used in asymmetric cryptography. These three elements provide a secure system for deploying e-commerce and a reliable environment for building virtually any type of electronic transactions, from corporate intranets to Internet-based e-business applications.

These elements include:

- Confidentiality
- Authentication
- Message integrity

Confidentiality

Confidentiality means that unintended users cannot view the data. In PKIs, confidentiality is achieved by encrypting the data through a variety of methods. In SSL, specifically, large amounts of data are encrypted using one or more symmetric keys that are known only by the two endpoints. Because the symmetric key is usually generated by one of the endpoints, it must be transmitted securely to the other endpoint. Secure transmittal of a symmetric key is generally achieved by two mechanisms, *key exchange* or *key agreement*.

Key exchange is the most common of these two secure transmittal mechanisms. In key exchange, one device generates the symmetric key and then encrypts it using an asymmetric encryption scheme before transmitting it to the other side. Asymmetric encryption requires that both devices have a public key and a private key. The two keys are mathematically related; data that can be encrypted by the public key can be decrypted by the private key, and vice versa. The most commonly used key exchange algorithm is the Rivest Shamir Adelman (RSA) algorithm.

For SSL, the sender encrypts the symmetric keys with the public key of the receiver. This ensures that the private key of the receiver is the only key that can decrypt the transmission. The security of asymmetric encryption depends entirely on the fact that the private key is known only by the owner and not by any other party. If this key were compromised for any reason, a fraudulent Web user (or Web site) could decrypt the stream containing the symmetric key and the entire data transfer.

In *key agreement*, the two sides involved in a data exchange cooperate to generate a symmetric (shared) key. The most common key agreement algorithm is the Diffie-Hellman algorithm. Diffie-Hellman depends on certain parameters to generate the shared key that is calculated and exchanged between the client and the server.

Authentication

Authentication is necessary for one or more devices in the exchange to verify that the party to whom they are talking is really who they claim to be. For example, assume a client is connecting to an e-commerce website. Before sending sensitive information such as a credit card number, the client verifies that the server is an e-commerce website. In certain instances, it may be necessary for both the client and the server to authenticate themselves to each other before beginning the transaction. In a financial transaction between two banks, both the client and the server need to be confident that the other is who they say they are. SSL facilitates this authentication through the use of digital certificates.

Digital certificates are a form of digital identification to prove the identity of the client to the server. A Certificate Authority (CA) issues digital certificates in the context of a PKI, which uses public-key and private-key encryption to ensure security. CAs are trusted authorities who “sign” certificates to verify their authenticity. Clients or servers connected to the CSS must have trusted certificates from the same CA, or from different CAs in a hierarchy of trusted relationships (for example, “A” trusts “B,” and “B” trusts “C,” therefore “A” trusts “C”).

A certificate ensures that the identification information is correct, and that the public key actually belongs to that client or server. Digital certificates contain information such as details about the owner, details about the certificate issuer, the owner’s public key, validity and expiration dates, and associated privileges.

Upon receiving a certificate, a client can connect to the certificate issuer and verify the validity of the certificate using the issuer’s public key. This ensures that the certificate is actually issued and signed by an authorized entity. A certificate remains valid until it expires or is terminated.

Message Integrity

Message integrity is a means of assuring the recipient of a message that the contents of the message have not been tampered with during transit. SSL achieves this by applying a message digest to the data before transmitting it. A message digest is a function that takes an arbitrary length message and outputs a fixed-length string that is characteristic of the message.

An important property of the message digest is that it is extremely difficult to reverse. Simply appending a digest of the message to itself before sending it is not enough to guarantee integrity. An attacker can change the message and then change the digest accordingly. Encoding the message digest with the sender's private key creates a Message Authentication Code (MAC), the message integrity algorithm, which the recipient can then decode using the sender's public key. SSL supports two different algorithms for a MAC: Message Digest 5 (MD5) and Secure Hash Algorithm (SHA).

This integrity scheme, however, does not work if the sender's private key is compromised. The attacker can now forge the sender's MACs. Message integrity also depends heavily on the protection of private keys. This process is known as digital signing.

RSA key pairs are effective for signing the MAC. However, it may be advantageous to separate the functions of key exchange and signing. The Digital Signature Algorithm (DSA) is an SSL algorithm that is used strictly for digital signatures but not for key exchange.

DSA was standardized as FIPS-186, which is the Digital Signature Standard (DSS). DSA and DSS can be used interchangeably. DSS uses the same crypto-math as Diffie-Hellman and requires parameters similar to Diffie-Hellman to generate keys. Additionally, DSS is restricted for use only with the Secure Hash Algorithm 1 (SHA-1) message digest.

SSL Module Cryptography Capabilities

Table 1-1 provides information on the SSL cryptography capabilities of the SSL module.

Table 1-1 SSL Module SSL Cryptography Capabilities

SSL Cryptography Function	Functions Supported by the SSL Module
SSL versions	SSL version 3.0 and Transport Layer Security (TLS) version 1.0
Public key exchange and key agreement algorithms	• **RSA** - 512-bit, 768-bit, 1024-bit, and 2048-bit (key exchange and key agreement algorithm) • **DSA** - 512-bit, 768-bit, and 1024-bit (certificate signing algorithm) • **Diffie-Hellman** - 512-bit, 768-bit, 1024-bit, and 2048-bit (key agreement algorithm)
Encryption types	• Data Encryption Standard (DES) • Triple-Strength Data Encryption Standard (3DES) • RC4 See Table 4-1 in Chapter 4, Configuring SSL Termination for a list of supported cipher suites and key encryption types.
Hash types	• SSL MAC-MD5 • SSL MAC-SHA1 See Table 4-1 in Chapter 4, Configuring SSL Termination for a list of supported cipher suites and hash types.

Table 1-1 SSL Module SSL Cryptography Capabilities (continued)

SSL Cryptography Function	Functions Supported by the SSL Module
Digital certificates	The SSL module supports all major digital certificates from Certificate Authorities (CAs), including those listed below: • VeriSign • Entrust • Netscape iPlanet • Windows 2000 Certificate Server • Thawte • Equifax • Genuity

Overview of the SSL Module Functions in the CSS

The CSS 11503 and CSS 11506 support multiple SSL modules; a maximum of two in a CSS 11503 and a maximum of four in a CSS 11506. The CSS 11501 supports a single integrated SSL module.

The SSL module is responsible for all user authentication, public/private key generation, certificate management, and packet encryption and decryption functions between the client and the server. It is dependent on the Switch Module to provide the interface for processing network traffic and the Switch Control Module (SCM) to send and receive configuration information.

The CSS stores all certificates and keys on the SCM disk. The CSS supports a maximum of 256 certificates and 256 key-pairs per SSL module, which equals approximately 3 MB of storage space on the disk. The CSS stores all certificate- and key-related files in a secure location on the disk. When processing connections, the CSS loads the certificates and keys into volatile memory on the SSL module for faster access.

No network traffic is sent to an SSL module from the SCM until an SSL content rule is activated to:

- Define where the content physically resides
- Specify where to direct the request for content (which service)
- Specify which load-balancing method to use

An SSL proxy list determines the flow of information to and from an SSL module. An entry in the proxy list defines the flow from a client to an SSL module. An entry also defines a flow from an SSL module to a back-end SSL server. To define how an SSL module processes SSL requests for content, add an SSL proxy list to an SSL service. For more detailed information on the SSL module functions, see the "Processing of SSL Flows by the SSL Module" section in Chapter 8, Examples of CSS SSL Configurations.

The SSL module provides the following major SSL features:

- SSL Termination
- Client Authentication
- Back-End SSL
- SSL Initiation

SSL Termination

When you create an entry in a proxy list to define the flow between an SSL module and a client, the module operates as a virtual SSL server by adding security services between a web browser (the client) and the HTTP connection (the server). All inbound SSL flows from a client terminate at an SSL module in the CSS.

Once the connection is terminated, the SSL module decrypts the data and sends the data as clear text to the CSS for a decision on load balancing. The CSS transmits the data as clear text to an HTTP server. For more information about SSL termination in the CSS, see Chapter 4, Configuring SSL Termination.

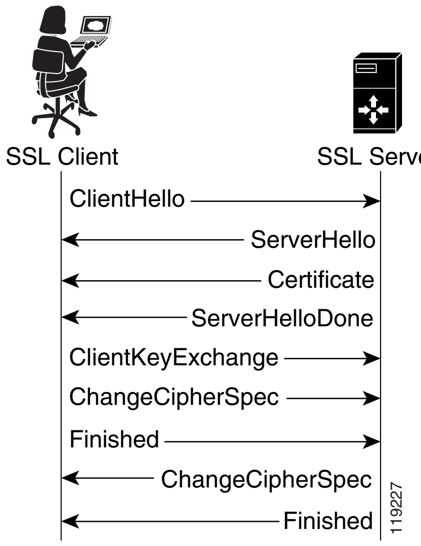
Client Authentication

When client authentication occurs on the CSS, the CSS verifies that the:

- Client sending the certificate has a corresponding private key
- Client certificate is signed by a known CA
- Certificate has not expired
- Signature is valid
- Issuing CA has not revoked the certificate if a Certificate Revocation List (CRL) is configured on the CSS

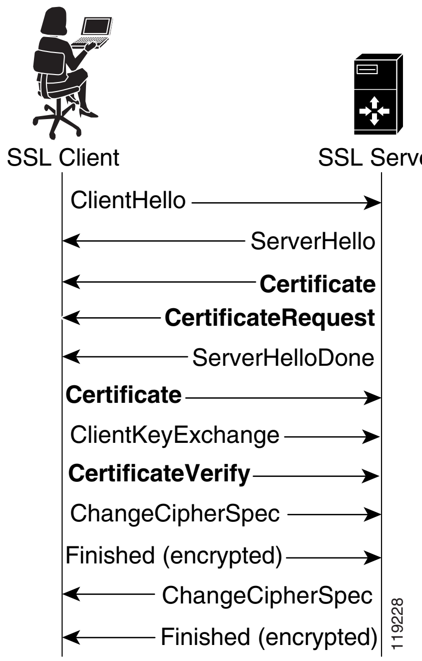
During a typical SSL handshake between a client and a server, the client does not send a certificate as shown in Figure 1-1.

Figure 1-1 SSL Handshake Without Client Authentication



For a client to send a certificate, the server must include a certificate request (CertificateRequest) message in the handshake as shown in Figure 1-2. The request message includes which types of certificates the server accepts. However, this message does not identify certificate authorities.

Figure 1-2 SSL Handshake With Client Authentication



After the server sends the ServerHelloDone message, the client responds with its certificate (Certificate) and key exchange. Then the client sends a CertificateVerify message that contains a digest of all the handshake messages from the server and was signed using the client public key. The server decrypts the message using the client public key ensuring that the client possesses the correct private key.

The CertificateVerify message does not check the authenticity of the certificate. However, it does check that the public portion of the client private key matches what is embedded in the certificate. This ensures that the client possesses the keypair that used to generate the certificate, and is not passing someone else’s certificate. However, the CSS can check whether the issuer signature is authentic.

An X.509 certificate includes a signature that is generated by signing a message digest of the entire certificate object using the private key of the CA. A CA certificate contains the CA public key that verifies the digital signature of the client certificate. If the server has a CA certificate and thus the public key of the CA, it can verify that the client certificate was signed by the CA. The CSS allows you to configure up to four CA certificates per virtual SSL server.

When a CA revokes a client certificate, the CA adds the certificate to a published list called the Certificate Revocation List (CRL). The CA publicizes this list and updates it periodically. Clients and servers can access this list through HTTP to validate a certificate. The CSS allows you to configure a CRL record that defines how and when to retrieve a CRL onto the CSS. After the CSS retrieves the CRL, the virtual SSL server can use the downloaded CRL to check the validity of all client certificates.

For information on configuring client authentication on a CSS virtual SSL server, including enabling client authentication, verifying CA certificate authenticity, configuring a CRL record, and assigning it to a virtual SSL server, see Chapter 4, Configuring SSL Termination.

Back-End SSL

A back-end SSL server entry in an SSL proxy list defines the flow from the SSL module to the back-end SSL server. After receiving encrypted data from a client, the SSL module, acting as a virtual client by preserving the originating client's IP address, encrypts the clear text data used for load balancing the flow and initiates the SSL connection to the back-end server.

On the outbound flow from the CSS, the SSL module responds in the reverse direction and sends the encrypted data from the server back to the client. For more information about back-end SSL in the CSS, see Chapter 5, Configuring Back-End SSL.

SSL Initiation

SSL initiation enables the CSS to receive clear text from a client and then to originate an SSL seesion with an SSL server and join the client connection with the SSL back-end connection. The SSL server can either be a CSS configured for SSL termination (virtual SSL server) or a real back-end SSL server (Web server).

On the outbound flow, the CSS decrypts the SSL data from the server and sends clear text back to the client. For more information about SSL initiation in the CSS, see Chapter 6, Configuring SSL Initiation.

For more detailed information on the SSL module functions, see the "Processing of SSL Flows by the SSL Module" section in Chapter 8, Examples of CSS SSL Configurations.