CHAPTER 1

Overview

This chapter describes the Secure Sockets Layer (SSL) application-level protocol that provides encryption technology for the Internet. SSL ensures the secure transmission of data between a client and a server through a combination of privacy, authentication, and data integrity. SSL relies upon certificates and private-public key exchange pairs for this level of security.

This chapter contains the following major sections:

SSL Cryptography Overview

The Cisco Application Control Engine (ACE) module uses a special set of SSL commands to perform the SSL cryptographic functions between a client and a server. The SSL functions include server authentication, private-key and public-key generation, certificate management, and data packet encryption and decryption.

The ACE supports SSL Version 3.0 and Transport Layer Security (TLS) Version 1.0. The ACE understands and accepts an SSL Version 2.0 ClientHello message, known as a hybrid 2/3 hello message, allowing dual-version clients to communicate with the ACE. When the client indicates SSL Version 3.0 in the Version 2.0 ClientHello, the ACE understands that the client can support SSL Version 3.0 and returns a Version 3.0 ServerHello message.

Note The ACE cannot pass network traffic if the client supports only SSL Version 2.0.

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

This section provides an overview of SSL cryptography as implemented in the ACE and contains the following topics:

- SSL Public Key Infrastructure
- SSL Handshake

SSL Public Key Infrastructure

SSL provides authentication, encryption, and data integrity in a Public Key Infrastructure (PKI). PKI is a set of policies and procedures that establishes a secure information exchange between devices. Three fundamental elements characterize the PKIs used in asymmetric cryptography, each of which is described in the topics that follow:

- Confidentiality
- Authentication
- Message Integrity

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.

Confidentiality

Confidentiality ensures 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 endpoint, it must be transmitted securely to the other endpoint. The ACE supports the use of the *key exchange* mechanism for the secure transmission of a symmetric key between the ACE and its peer.

In a key exchange, one device generates the symmetric key and then encrypts it using an asymmetric encryption scheme before transmitting the key to its peer. Asymmetric encryption requires that each device has a unique key pair that consists of a public key and a private key. The two keys are mathematically related; data that is encrypted using the public key can only be decrypted using the corresponding private key, and vice versa. A device shares its public key with its peer but must keep its private key a secret.

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 website) could decrypt the stream containing the symmetric key and the entire data transfer. 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 to ensure that the private key of the receiver is the only key that can decrypt the transmission.

Authentication

Authentication ensures that one or more devices in the exchange can verify the identity of the other device. 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 a legitimate e-commerce website. Both the client and the server may need to authenticate themselves to each other before beginning the transaction. In a financial transaction between two banks, both the client and the server must be confident of the other's identity. SSL facilitates this authentication through the use of digital certificates.

Digital certificates are a form of digital identification to prove the identity of the server to the client, or optionally, the client to the server. A certificate ensures that the identification information is correct and that the public key embedded in the certificate actually belongs to that client or 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. Digital certificates contain the following information:

- Details about the owner (the certificate subject)
- Details about CA (the certificate issuer)
- Public key for the certificate's subject
- Certificate validity and expiration dates
- Privileges associated with the certificate

As the certificate issuer, the CA uses a private key to sign the certificate. Upon receiving a certificate, a client uses the issuer's public key to decrypt and verify the certificate signature. This procedure ensures that the certificate was actually issued and signed by an authorized entity.

Public key certificates support *certificate hierarchies*. A CA creates a hierarchy of subsidiary authorities that share in the responsibility of issuing signed certificates. The CA that sits at the head of the hierarchy is known as the *root authority*. Each level in the hierarchy certifies the level below it, creating a hierarchy of trusted relationships known as *certificate chaining*. This process enables an entity that is verifying a certificate to trace the CA certificates back to the root authority, if needed, to find a CA in the hierarchy that it trusts.

A certificate remains valid until it expires or is revoked by the CA. When a CA revokes a certificate, it adds the certificate to a certificate revocation list (CRL) that lists any certificates that it previously issued but no longer considers valid.

Clients or servers connected to the ACE 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).

Message Integrity

Message integrity assures the recipient of a message that the contents of the message have not been tampered with during transit. To ensure message integrity, SSL applies a message digest to the data before transmitting it. A message digest 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.

Each message exchanged between peers is protected by a message authentication code (MAC), which can be calculated by using a hash algorithm such as SHA or MD5. The MAC is a hash value of several pieces of data, including a secret value, the actual data being sent, and a sequence number. The secret value is the write session key. The sequence number is a 32-bit counter value. This data is processed by the hash algorithm to derive the MAC. Upon receipt of a message, the receiver verifies the MAC by using the read session key and the predicted sequence number and calculates the hash over the received data. If the two hash values do not match, the data stream has been modified in some way.

SSL Handshake

The client and server use the SSL handshake protocol to establish an SSL session between the two devices. During the handshake, the client and server negotiate the SSL parameters that they will use during the secure session. Figure 1-1 shows the client/server actions that occur during the SSL handshake.

Note The ACE does not replicate SSL and other terminated (proxied) connections from the active context to the standby context.

Figure 1-1 SSL Handshake

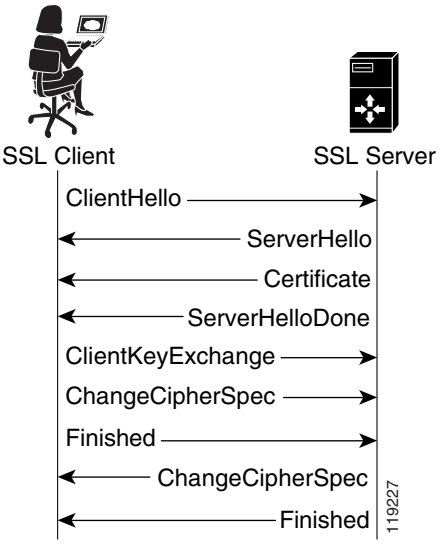


Table 1-1 describes the actions that take place between the client and the server during the SSL handshake.

Table 1-1 SSL Handshake Actions

Step	Message	Action
1	ClientHello	Client initiates the handshake by sending the ClientHello message that proposes the SSL parameters to use during the SSL session.
2	ServerHello	Server responds with the ServerHello message that contains the SSL parameters that it selects for use during the SSL session.
3	Certificate	Server sends the client its public key certificate.
4	ServerHelloDone	Server concludes its part of the SSL negotiations.
5	ClientKeyExchange	Client sends session key information that it encrypts using the server's public key.

Table 1-1 SSL Handshake Actions (continued)

Step	Message	Action
6	ChangeCipherSpec	Client instructs the server to activate the negotiated SSL parameters for all future messages that it sends.
7	Finished	Client instructs the server to verify that the SSL negotiation has been successful.
8	ChangeCipherSpec	Server instructs the client to activate the negotiated SSL parameters for all future messages that it sends.
9	Finished	Server instructs the client to verify that the SSL negotiation has been successful.

ACE SSL Capabilities

Table 1-2 provides information on the SSL capabilities of the ACE.

Table 1-2 ACE SSL Capabilities

SSL Feature	Feature Type or Specification Supported by the ACE
SSL versions	• SSL Version 3.0 and Transport Layer Security (TLS) Version 1.0 • SSL Version 2.0 ClientHello message (hybrid 2/3 hello)
Public key exchange algorithm	RSA—512 bits, 768 bits, 1024 bits, 1536 bits, 2048 bits
Encryption types	• Data Encryption Standard (DES) • Triple-Strength Data Encryption Standard (3DES) • RC4 • AES
Hash types	• SSL MAC-MD5 • SSL MAC-SHA1

Table 1-2 ACE SSL Capabilities (continued)

SSL Feature	Feature Type or Specification Supported by the ACE
Cipher suites	• RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA • RSA_WITH_DES_CBC_SHA • RSA_WITH_3DES_EDE_CBC_SHA • RSA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA • RSA_EXPORT1024_WITH_RC4_56_MD5 • RSA_EXPORT1024_WITH_DES_CBC_SHA • RSA_EXPORT1024_WITH_RC4_56_SHA • RSA_WITH_AES_128_CBC_SHA • RSA_WITH_AES_256_CBC_SHA
Digital certificates	Supports all major digital certificates from Certificate Authorities (CAs), including the following: • VeriSign • Entrust • Netscape iPlanet • Windows 2000 Certificate Server • Thawte • Equifax • Genuity
Maximum number of certificates	3800
Maximum size of a certificate file	Unlimited (cannot exceed the available capacity of the flash disk)
Maximum number of key pairs	3800

Table 1-2 ACE SSL Capabilities (continued)

SSL Feature	Feature Type or Specification Supported by the ACE
Maximum number of concurrent SSL connections	200,000
Maximum number of SSL transactions per second (TPS)	By default, the ACE supports 1000 SSL TPS. Installing an optional SSL TPS license allows you to increase the number of TPS that your ACE supports to 5000, 10000, or 15000 TPS. See the *Cisco Application Control Engine Module Administration Guide* guide for information on ACE licensing options.
Maximum amount of SSL bandwidth	4 Gbps

ACE SSL Functions

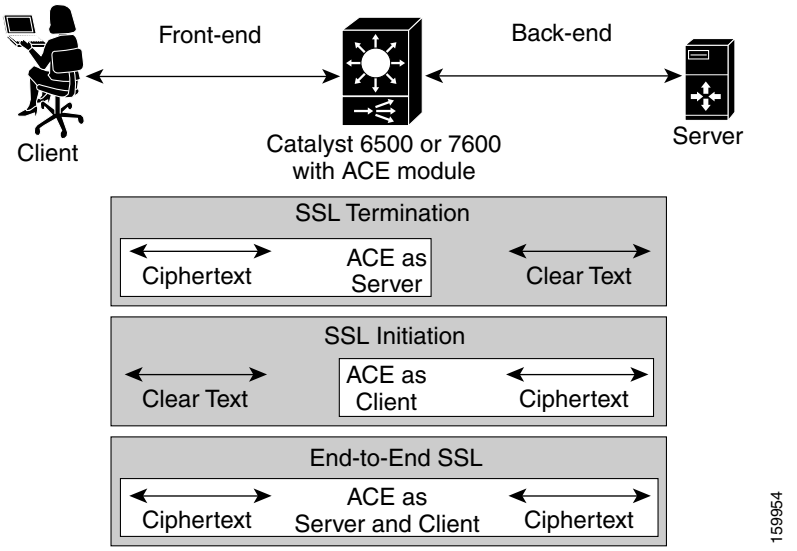
The ACE is responsible for all user authentication, public/private key generation, certificate management, and packet encryption and decryption functions between the client and the server.

You can partition the ACE into multiple contexts (virtual ACE devices). You configure each context with the certificate and key files that the context needs to establish an SSL session with its peer. The ACE creates a secure storage area in flash memory for storing the certificates and keys associated with each context that you create.

To establish and maintain an SSL session between the ACE and its peer, the ACE applies policy maps to the traffic that it receives. When the traffic characteristics match the attributes of a specific policy map, the ACE executes the actions associated with the policy map.

Depending on how you define the policy map, you can configure the ACE to act as a client or a server during an SSL session. Figure 1-2 shows the three basic SSL configurations of the ACE in which the ACE is used to encrypt and decrypt data between the client and the server.

Figure 1-2 ACE SSL Applications



The following topics provide an overview of the three ACE SSL applications:

- SSL Termination
- SSL Initiation
- End-to-End SSL

SSL Termination

SSL termination refers to configuring an ACE context for a front-end application in which the ACE operates as an SSL server that communicates with a client. When you create a Layer 3 and Layer 4 policy map to define the flow between an ACE and a client, the ACE 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 the ACE.

After the connection is terminated, the ACE decrypts the ciphertext from the client and sends the data as clear text to an HTTP server. For information about configuring the ACE for SSL termination, see Chapter 3, Configuring SSL Termination.

SSL Initiation

SSL initiation refers to configuring an ACE context for a back-end application in which the ACE operates as a client that communicates with an SSL server. When you create a Layer 7 policy map to define the flow between an ACE and an SSL server, the ACE operates as a client and initiates the SSL session between the ACE and the server. SSL initiation enables the ACE to receive clear text from a client and then to establish an SSL session with an SSL server and join the client connection with the SSL server connection. The ACE encrypts the clear text that it receives from the client and sends the data as ciphertext to an SSL server. The SSL server can either be an ACE configured for SSL termination (virtual SSL server) or a real SSL server (web server).

On the outbound flow from the SSL server, the ACE decrypts the ciphertext from the server and sends clear text back to the client.

For more information on configuring the ACE for SSL initiation, see Chapter 4, Configuring SSL Initiation.

End-to-End SSL

End-to-end SSL refers to configuring an ACE context for both SSL termination and SSL initiation. You can configure the ACE for end-to-end SSL when you have an application that requires establishing a secure SSL channel between the client, the ACE, and the SSL server. For example, a transaction between banks requires end-to-end SSL to protect the financial information exchanged between the client and the server.

End-to-end SSL also allows the ACE to insert load-balancing and security information into the data. The ACE decrypts the ciphertext it receives and inserts the load-balancing and firewall information into the clear text. The ACE then reencrypts the data and passes the ciphertext to its intended destination.

For more information on configuring the ACE for end-to-end SSL initiation, see Chapter 5, Configuring End-to-End SSL.

ACE SSL Configuration Prerequisites

Before configuring your ACE for SSL operation, you must first configure it for server load balancing (SLB). During the SLB configuration process, you create the following configuration objects:

- Layer 7 class map
- Layer 3 and Layer 4 class map
- Layer 7 policy map
- Layer 3 and Layer 4 policy map

After configuring SLB, modify the existing SLB class maps and policy maps with the SSL configuration requirements described in this guide for SSL termination, SSL initiation, or end-to end SSL.

To configure your ACE for SLB, see the *Cisco Application Control Engine Module Server Load-Balancing Configuration Guide*.

SSL Behavior with Large SSL Records and Small MSS

With a combination of a large SSL record size and a small TCP maximum segment size (MSS), ACE SSL throughput may become very slow or possibly even stall because of a zero TCP window size. This behavior can happen with any of the following SSL configurations:

- SSL termination
- SSL initiation
- End-to-end SSL (combination of SSL termination and SSL initiation)

The reason for this behavior is that the ACE needs to receive the entire buffered SSL record before it can encrypt or decrypt the data. If a local interface MSS is set too low, then a large SSL record would use up all the available buffers before the ACE could receive the entire record and the sender would stop sending additional packets because of an advertised zero TCP window size. If the **set tcp buffer-share** command value is also set too low, the problem is exacerbated and the zero window size may be seen sooner. The actual determining factor is the path maximum transmit unit (PMTU) or the smallest MTU along a packet's path.

You should never decrease the default buffer-share value (32768) when using SSL with the ACE. For cases where the default value is not sufficient, you can increase the **set tcp buffer-share** value to allow more buffers to be available to accommodate large SSL records and/or small MSSs.

For example, in the case of a 160 MSS (200 MTU), it would require 103 buffers (16,400 byte maximum SSL record size divided by 160) to create a maximum-sized SSL record. As far as TCP is concerned, 103 buffers can hold 57,736 bytes. Therefore, the buffer-share value would have to be at least 57736 for the SSL connection to continue. Different MSS and MTU sizes will require different buffer-share value settings.

For more information about the **set tcp buffer-share** command, see the *Cisco Application Control Engine Module Security Configuration Guide*.