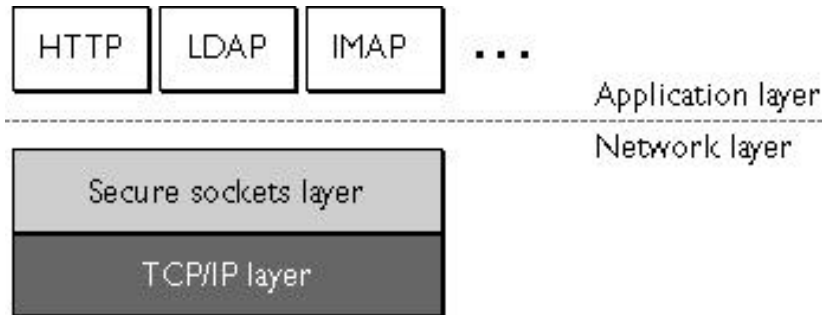


HiPACE SSL Stack

The SSL Stack runs above TCP/IP and below higher-level protocols such as HTTP or IMAP. It uses TCP/IP on behalf of the higher-level protocols, and in the process allows the SSL stack to



authenticate itself to an SSL-enabled client (example a web browser), allows the client to authenticate itself to the server, and allows both machines to establish an encrypted connection.

The SSL stack includes two sub-protocols: the SSL record protocol and the SSL handshake protocol. The SSL record protocol defines the format used to transmit data. The SSL handshake protocol involves using the SSL record protocol to exchange a series of messages between the SSL Stack and an SSL-enabled client (web browser) when they first establish an SSL connection. This exchange of messages is designed to facilitate the following actions:

- Authenticate the server to the client.
- Allow the client and server to select the cryptographic algorithms, or ciphers, that they both support.
- Optionally authenticate the client to the server.
- Use public-key encryption techniques to generate shared secrets.
- Establish an encrypted SSL connection.

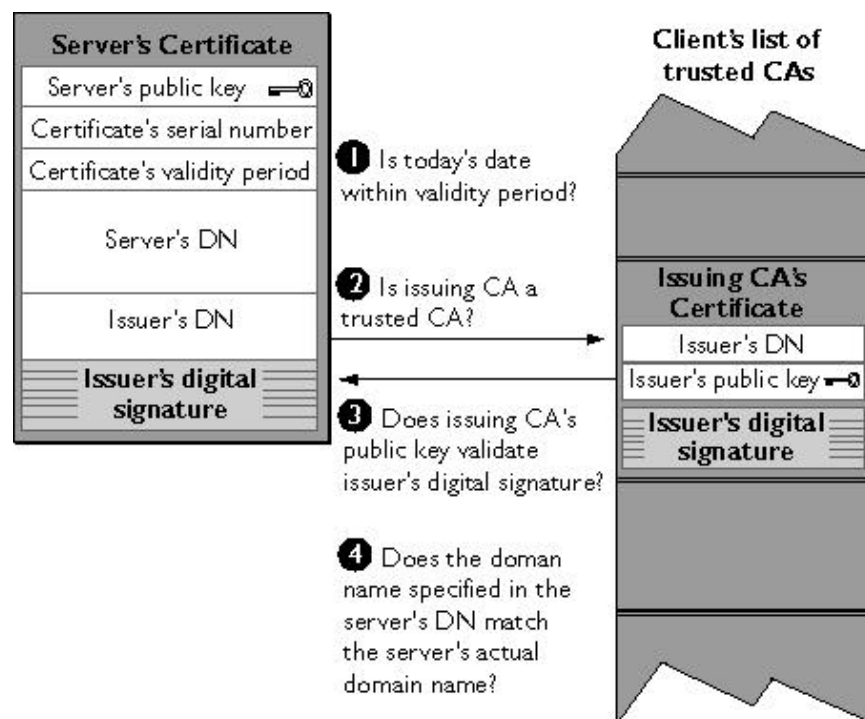
The SSL Handshake

The SSL Stack uses a combination of public-key and symmetric key encryption. Symmetric key encryption is much faster than public-key encryption, but public-key encryption provides better authentication techniques. An SSL session always begins with an exchange of messages called the **SSL handshake**. The handshake allows the server to authenticate itself to the client using public-key techniques, then allows the client and the server to cooperate in the creation of symmetric keys used for rapid encryption, decryption, and tamper detection during the session that follows. Optionally, the handshake also allows the client to authenticate itself to the server.

1. The client sends the server the client's SSL version number, cipher settings, randomly generated data, and other information the server needs to communicate with the client using SSL.
2. The server sends the client the server's SSL version number, cipher settings, randomly generated data, and other information the client needs to communicate with the server over SSL. The server also sends its own certificate and, if the client is requesting a server resource that requires client authentication, requests the client's certificate.
3. The client uses some of the information sent by the server to authenticate the server. If the server cannot be authenticated, the user is warned of the problem and informed that an encrypted and authenticated connection cannot be established. If the server can be successfully authenticated, the client goes on to the next step.

4. Using all data generated in the handshake so far, the client (with the cooperation of the server, depending on the cipher being used) creates the **premaster secret** for the session, encrypts it with the server's public key, and sends the encrypted premaster secret to the server.
5. If the server has requested client authentication (an optional step in the handshake), the client also signs another piece of data that is unique to this handshake and known by both the client and server. In this case the client sends both the signed data and the client's own certificate to the server along with the encrypted premaster secret.
6. If the server has requested client authentication, the server attempts to authenticate the client. If the client cannot be authenticated, the session is terminated. If the client can be successfully authenticated, the server uses its private key to decrypt the premaster secret, then performs a series of steps (which the client also performs, starting from the same premaster secret) to generate the **master secret**.
7. Both the client and the server use the master secret to generate the **session keys**, which are symmetric keys used to encrypt and decrypt information exchanged during the SSL session and to verify its integrity--that is, to detect any changes in the data between the time it was sent and the time it is received over the SSL connection.
8. The client sends a message to the server informing it that future messages from the client will be encrypted with the session key. It then sends a separate (encrypted) message indicating that the client portion of the handshake is finished.
9. The server sends a message to the client informing it that future messages from the server will be encrypted with the session key. It then sends a separate (encrypted) message indicating that the server portion of the handshake is finished.
10. The SSL handshake is now complete, and the SSL session has begun. The client and the server use the session keys to encrypt and decrypt the data they send to each other and to validate its integrity.

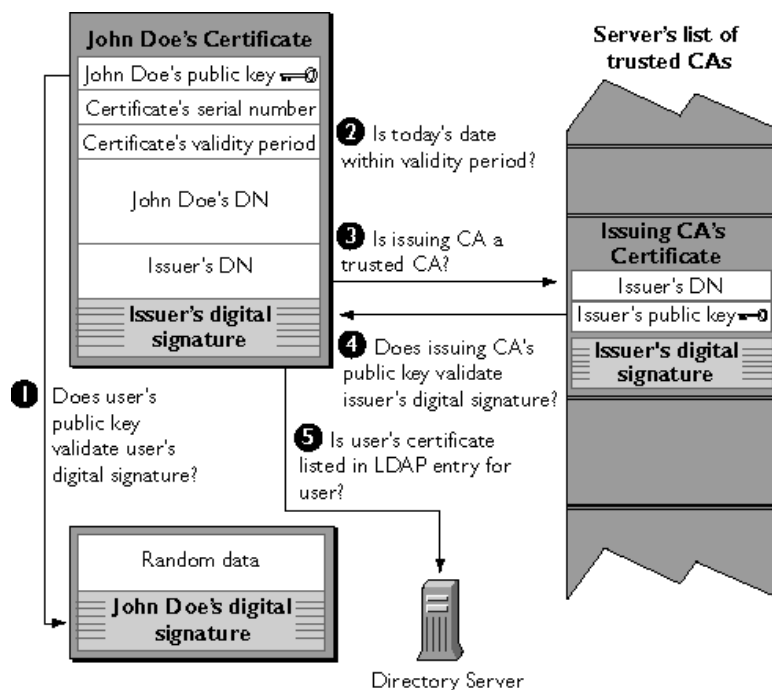
The Server Authentication



The browser goes through these steps to authenticate a server's identity:

1. **Is today's date within the validity period?** The client checks the server certificate's validity period. If the current date and time are outside of that range, the authentication process won't go any further. If the current date and time are within the certificate's validity period, the client goes on to next step.
2. **Is the issuing CA a trusted CA?** Each SSL-enabled client maintains a list of trusted CA certificates. This list determines which server certificates the client will accept. If the distinguished name (DN) of the issuing CA matches the DN of a CA on the client's list of trusted CAs, the answer to this question is yes, and the client goes on to the next step. If the issuing CA is not on the list, the server will not be authenticated unless the client can verify a certificate chain ending in a CA that is on the list.
3. **Does the issuing CA's public key validate the issuer's digital signature?** The client uses the public key from the CA's certificate (which it found in its list of trusted CAs in step 2) to validate the CA's digital signature on the server certificate being presented. If the information in the server certificate has changed since it was signed by the CA or if the CA certificate's public key doesn't correspond to the private key used by the CA to sign the server certificate, the client won't authenticate the server's identity. If the CA's digital signature can be validated, the server treats the user's certificate as a valid "letter of introduction" from that CA and proceeds. At this point, the client has determined that the server certificate is valid.
4. **Does the domain name in the server's certificate match the domain name of the server itself?** This step confirms that the server is actually located at the same network address specified by the domain name in the server certificate. Clients must perform this step and must refuse to authenticate the server or establish a connection if the domain names don't match. If the server's actual domain name matches the domain name in the server certificate, the client goes on to the next step.
5. **The server is authenticated.** The client proceeds with the SSL handshake. If the client doesn't get to this step for any reason, the server identified by the certificate cannot be authenticated, and the user will be warned of the problem and informed that an encrypted and authenticated connection cannot be established.

Client Authentication



The SSL Stack goes through these steps to authenticate a user's identity:

1. **Does the user's public key validate the user's digital signature?** The server checks that the user's digital signature can be validated with the public key in the certificate. If so, the server has established that the public key asserted to belong to John Doe matches the private key used to create the signature and that the data has not been tampered with since it was signed.
2. **Is today's date within the validity period?** The server checks the certificate's validity period. If the current date and time are outside of that range, the authentication process won't go any further. If the current date and time are within the certificate's validity period, the server goes on to next step.
3. **Is the issuing CA a trusted CA?** The SSL Stack maintains a list of trusted CA certificates. This list determines which certificates the server will accept. If the DN of the issuing CA matches the DN of a CA on the server's list of trusted CAs, the answer to this question is yes, and the server goes on to the next step. If the issuing CA is not on the list, the client will not be authenticated unless the server can verify a certificate chain ending in a CA that is on the list. Administrators can control which certificates are trusted or not trusted within their organizations by controlling the lists of CA certificates maintained by clients and servers.
4. **Does the issuing CA's public key validate the issuer's digital signature?** The server uses the public key from the CA's certificate to validate the CA's digital signature on the certificate being presented. If the information in the certificate has changed since it was signed by the CA or if the public key in the CA certificate doesn't correspond to the private key used by the CA to sign the certificate, the server won't authenticate the user's identity. If the CA's digital signature can be validated, the server treats the user's certificate as a valid "letter of introduction" from that CA and proceeds. At this point, the SSL stack considers the client authenticated and proceed with the connection as described in point #6.
5. **Is the user's certificate listed in the LDAP entry for the user?** This optional step provides one way for a system administrator to revoke a user's certificate even if it passes the tests in all the other steps. The stack can automatically remove a revoked certificate from the user's entry in the LDAP directory. All servers that are set up to perform this step will then refuse to authenticate that certificate or establish a connection. If the user's certificate in the directory is identical to the user's certificate presented in the SSL handshake, the server goes on to step 6.
6. **Is the authenticated client authorized to access the requested resources?** The server checks what resources the client is permitted to access according to the server's access control lists (ACLs) and establishes a connection with appropriate access. If the server doesn't get to step 6 for any reason, the user identified by the certificate cannot be authenticated, and the user is not allowed to access any server resources that require authentication.