Secure Sockets Layer

Secure Sockets Layer (SSL) is a standard technique for transmitting documents securely across a network. SSL technology, created by Netscape, establishes a secure connection between a Web server and a browser, ensuring private and secure data transmission. SSL communicates using the Transport Control Protocol (TCP).

The term "socket" in SSL refers to the method of sending data via a network between a client and a server.

A Web server requires an SSL certificate to establish a secure SSL connection while using SSL for safe Internet transactions. SSL encrypts network connection segments atop the transport layer, a network connection component above the program layer.

SSL is based on an asymmetric cryptographic process in which a Web browser generates both a public and a private (secret) key. A certificate signing request is a data file that contains the public key (CSR). Only the recipient receives the private key.

How Does SSL Work?

SSL encrypts data communicated across the web to guarantee a high level of privacy. Anyone attempting to intercept this data will meet a jumbled mess of characters nearly hard to decrypt.

SSL begins an authentication process known as a handshake between two communicating devices to confirm that both devices are who they say they are.

SSL also digitally certifies data to ensure data integrity, ensuring that it has not been tampered with before reaching its intended receiver.

SSL has gone through multiple incarnations, each one more secure than the last. TLS (Transport Layer Security) was introduced in 1999, replacing SSL.

**How SSL works**

Encryption is necessary in order to communicate securely over the internet: if your data isn't encrypted, anyone can examine your packets and read confidential information. The safest method of encryption is called *asymmetrical cryptography;*this requires two cryptographic *keys*— pieces of information, usually very large numbers — to work properly, one public and one private. The mathematics here are complex, but in essence, you can use the public key to *encrypt*the data, but need the private key to *decrypt*it. The two keys are related to each other by some complex mathematical formula that is difficult to reverse-engineer by brute force. Think of the public key as information about the location of a locked mailbox with a slot on the front, and the private key as the key that unlocks the mailbox. Anyone who knows where the mailbox is can put a message in it; but for anyone else to read it, they need the private key.

Because asymmetrical cryptography involves these difficult mathematical problems, it takes a lot of computing resources, so much so that if you used it to encrypt all the information in a communications session, your computer and connection would grind to a halt. TLS gets around this problem by only using asymmetrical cryptography at the very beginning of a communications session to encrypt the conversation the server and client have to agree on a single *session key* that they'll both use to encrypt their packets from that point forward. Encryption using a shared key is called *symmetrical cryptography,*and it's much less computationally intensive than asymmetric cryptography. Because that session key was established using asymmetrical cryptography, the communication session as a whole is much more secure than it otherwise would be.

The process by which that sessions key is agreed upon is called a *handshake,* since it's the moment when the two communicating computers introduce themselves to each other, and it's at the heart of the TLS protocol.

Objectives of SSL

The goals of SSL are as follows −

* *Data integrity* − Information is safe from tampering. The SSL Record Protocol, SSL Handshake Protocol, SSL Change CipherSpec Protocol, and SSL Alert Protocol maintain data privacy.
* *Client-server authentication* − The SSL protocol authenticates the client and server using standard cryptographic procedures.
* SSL is the forerunner of Transport Layer Security (TLS), a cryptographic technology for secure data transfer over the Internet.

How to Obtain an SSL/TLS Certificate?

Are you ready to protect your website? The following is the fundamental approach for requesting a publicly trusted SSL/TLS website certificate −

* The individual or organization requesting the certificate generates a pair of public and private keys, which should be stored on the server being protected.
* A certificate signing request is generated using the public key, the domain name(s) to be protected, and (for OV and EV certificates) organizational information about the company requesting the certificate (CSR).
* A publicly trusted CA receives the CSR (such as SSL.com). The CA verifies the information in the CSR and generates a signed certificate that the requester can install on their web server.

**SSL handshake process**

The handshake process is quite complex, and there are a number of variations allowed by the protocol. The following steps provide a broad outline that should give you a sense of how it works.

1. The client contacts the server and requests a secure connection. The server replies with the list of *cipher suites* — algorithmic toolkits of creating encrypted connections — that it knows how to use. The client compares this against its own list of supported cipher suites, selects one, and lets the server know that they'll both be using it.
2. The server then provides its *digital certificate,*an electronic document issued by a third-party authority confirming the server's identity. We'll discuss digital certificates in more detail in a moment, but for now the most important thing you need to know about them is that they contain the server's public cryptographic key. Once the client receives the certificate, it confirms the certificate's authenticity.
3. Using the server's public key, the client and server establish a session key that both will use for the rest of the session to encrypt communication. There are several techniques for doing this. The client may use the public key to encrypt a random number that's then sent to the server to decrypt, and both parties then use that number to establish the session key. Alternately, the two parties may use what's called a [Diffie–Hellman key exchange](https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange) to establish the session key.

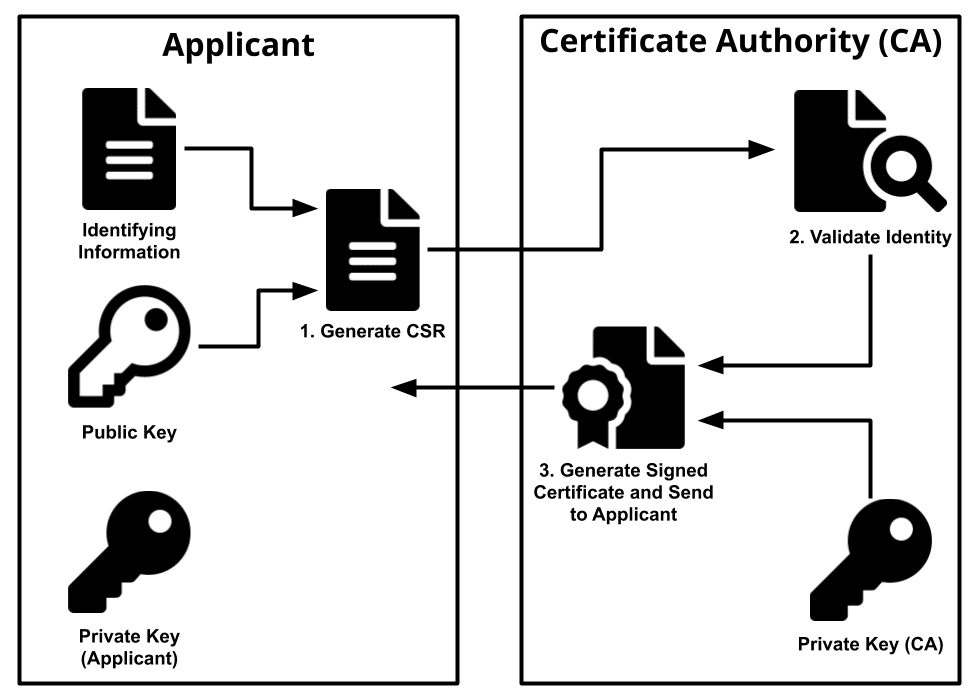
**SSL Certificate**

**SSL stands for Secure Sockets Layer, a global standard security technology that enables encrypted communication between a web browser and a web server. It is utilized by millions1 of online businesses and individuals to decrease the risk of sensitive information (e.g., credit card numbers, usernames, passwords, emails, etc.) from being stolen or tampered with by hackers and identity thieves. In essence, SSL allows for a private “conversation” just between the two intended parties.**

**To create this secure connection, an SSL certificate (also referred to as a “digital certificate”) is installed on a web server and serves two functions:**

* **It authenticates the identity of the website (this guarantees visitors that they’re not on a bogus site)**
* **It encrypts the data that’s being transmitted**

What is a CA?

* **What Is a Certificate Authority (CA)?**
* A **certificate authority (CA)**, also sometimes referred to as a **certification authority**, is a company or organization that acts to validate the identities of entities (such as websites, email addresses, companies, or individual persons) and bind them to cryptographic keys through the issuance of electronic documents known as **digital certificates**.  
    
  A digital certificate provides:  
  **Authentication**, by serving as a credential to validate the identity of the entity that it is issued to.  
  **Encryption,**for secure communication over insecure networks such as the Internet.  
  **Integrity** of documents **signed** with the certificate so that they cannot be altered by a third party in transit.
* [](https://d1smxttentwwqu.cloudfront.net/wp-content/uploads/2019/07/ca-diagram-b.png?_gl=1*7g63ez*_gcl_au*MzMwNTM1NzQ2LjE2ODYxMTA3NDc.)

Typically, an applicant for a digital certificate will generate a **key pair** consisting of a **private key** and a **public key**, along with a [**certificate signing request (CSR)**](https://www.ssl.com/faqs/what-is-a-csr/). A CSR is an encoded text file that includes the public key and other information that will be included in the certificate (e.g. domain name, organization, email address, etc.). Key pair and CSR generation are usually done on the server or workstation where the certificate will be installed, and the type of information included in the CSR varies depending on the validation level and intended use of the certificate. **Unlike the public key, the applicant’s private key is kept secure and should never be shown to the CA (or anyone else).**

After generating the CSR, the applicant sends it to a CA, who independently verifies that the information it contains is correct and, if so, digitally signs the certificate with an issuing private key and sends it to the applicant.

When the signed certificate is presented to a third party (such as when that person accesses the certificate-holder’s website), the recipient can cryptographically confirm the CA’s digital signature via the CA’s public key. Additionally, the recipient can use the certificate to confirm that signed content was sent by someone in possession of the corresponding private key, and that the information has not been altered since it was signed. A key part of this aspect of the certificate is something called a chain of trust.

**What is a trust anchor?**

The root [certificate authority (CA)](https://www.ssl.com/faqs/what-is-a-certificate-authority/) serves as the *trust anchor* in a chain of trust. The validity of this trust anchor is vital to the integrity of the chain as a whole. If the CA is *publicly trusted* (like SSL.com), the root CA certificates are included by major software companies in their browser and operating system software. This inclusion ensures that certificates in a chain of trust leading back to any of the CA’s root certificates will be trusted by the software.

**What is an intermediate certificate?**

The root CA or trust anchor has the ability to sign and issue *intermediate certificates*. Intermediate certificates (also known as *intermediate*, *subordinate*, or *issuing CAs*) provide a flexible structure for conferring the validity of the trust anchor to additional intermediate and end-entity certificates in the chain. In this sense, intermediate certificates serve an administrative function; each intermediate can be used for a specific purpose — such as issuing SSL/TLS or code signing certificates — and can even be used to [confer](https://www.ssl.com/article/subordinate-cas-and-why-you-might-need-one/)the root CA’s trust to other organizations.  
  
Intermediate certificates also provide a buffer between the end-entity certificate and the root CA, protecting the private root key from compromise. For publicly trusted CAs (including SSL.com), the CA/Browser forum’s [Baseline Requirements](https://cabforum.org/baseline-requirements-documents/) actually prohibit issuing end-entity certificates directly from the root CA, which must be kept securely offline. This means that any publicly trusted certificate’s chain of trust will include at least one intermediate certificate.

**What is an end-entity certificate?**

The *end-entity certificate* is the final link in the chain of trust. The end-entity certificate (sometimes known as a *leaf certificate* or *subscriber certificate*), serves to confer the root CA’s trust, via any intermediates in the chain, to an entity such as a website, company, [government](https://www.ssl.com/article/pki-and-digital-certificates-for-government/), or individual person.  
  
An end-entity certificate differs from a trust anchor or intermediate certificate in that it cannot issue additional certificates. It is, in a sense, the final link as far as the chain is concerned. The example below shows the end-entity SSL/TLS certificate from SSL.com’s website: