TLS/SSL
SSL (Secure Sockets Layer) and its successor, **TLS (Transport Layer Security)**, are protocols for establishing authenticated and encrypted links between networked computers. Although the SSL protocol was deprecated with the release of TLS 1.0 in 1999, it is still common to refer to these related technologies as "SSL" or "SSL/TLS." The most current version is **TLS 1.3**, defined in **RFC 8446** (August 2018).

What is an SSL certificate?

An **SSL certificate** (also known as a TLS or SSL/TLS certificate) is a digital document that binds the identity of a website to a cryptographic key pair consisting of a public key and a private key. The public key, included in the certificate, allows a web browser to initiate an encrypted communication session with a web server via the TLS and HTTPS protocols. The private key is kept secure on the server, and is used to digitally sign web pages and other documents (such as images and JavaScript files).

An SSL certificate also includes identifying information about a website, including its domain name and, optionally, identifying information about the site's owner. If the web server's SSL certificate is signed by a publicly trusted certificate authority (CA), like SSL.com, digitally signed content from the server will be trusted by end users' web browsers and operating systems as authentic.

An SSL certificate is a type of X.509 certificate.

What is TLS?

TLS (Transport Layer Security), released in 1999, is the successor to the **SSL (Secure Sockets Layer)** protocol for authentication and encryption. TLS 1.3 is defined in in **RFC 8446** (August 2018).

Do I need a dedicated IP address to use SSL/TLS?

At one time it was a mandatory requirement to have a dedicated IP for each SSL certificate on a web server. This is no longer the case due to a technology called Server Name Indication (SNI). Your hosting platform will specifically have to support SNI. You can find out more information about SNI in this SSL.com article.

What port is recommended to use SSL/TLS over?

For maximum compatibility, port 443 is the standard, thus recommended, port used for secured SSL/TLS communications. However, any port can be used.

What is the current version of SSL/TLS?

TLS 1.3, defined in August 2018 by RFC 8446, is the most recent version of SSL/TLS. TLS 1.2 (RFC 5246) was defined in August 2008 and also remains in wide use. Versions of SSL/TLS prior to TLS 1.2 are considered insecure and should no longer be used.

What are the security issues with older versions of TLS?

TLS versions 1.0 and 1.1 are affected by a large number of protocol and implementation vulnerabilities that have been published by security researchers in the last two decades. Attacks like **ROBOT** affected the RSA key exchange algorithm, while **LogJam** and **WeakDH** showed that many TLS servers could be tricked into using incorrect parameters for other key exchange methods. Compromising a key exchange allows attackers to completely compromise network security and decrypt conversations.

Attacks on symmetric ciphers, such as **BEAST** or **Lucky13**, have demonstrated that various ciphers supported in TLS 1.2 and earlier, with examples including **RC4** or **CBC-mode** ciphers, are not secure.

Even signatures were affected, with **Bleichenbacher's RSA signature forgery** attack and other similar padding attacks.

Most of these attacks have been mitigated in TLS 1.2 (provided that TLS instances are configured correctly), even though TLS 1.2 is still vulnerable to downgrade attacks, such as **POODLE**, **FREAK**, or **CurveSwap**. This is due to the fact that all versions of the TLS protocol prior to 1.3 don't protect the handshake negotiation (which decides the protocol version that will be used throughout the exchange).

Go to top

Keys, Certificates, and Handshakes

SSL/TLS works by binding the identities of entities such as websites and companies to cryptographic **key pairs** via digital documents known as **X.509 certificates**. Each key pair consists of a **private key** and a **public key**. The private key is kept secure, and the public key can be widely distributed via a certificate. The special mathematical relationship between the private and public keys in a pair mean that it is possible to use the public key to encrypt a message that can only be decrypted with the private key. Furthermore, the holder of the private key can use it to **sign** other digital documents (such as web pages), and anyone with the public key can verify this signature.
For a detailed comparison of the two most widely used digital signature algorithms used in SSL/TLS, please read our article, **Comparing ECDSA vs RSA**.

If the SSL/TLS certificate itself is signed by a **publicly trusted certificate authority (CA)**, such as **SSL.com**, the certificate will be implicitly trusted by client software such as web browsers and operating systems. Publicly trusted CAs have been approved by major software suppliers to validate identities that will be trusted on their platforms. A public CA's validation and certificate issuance procedures are subject to regular, rigorous audits to maintain this trusted status. Via the **SSL/TLS handshake**, the private and public keys can be used with a publicly trusted certificate to negotiate an encrypted and authenticated communication session over the internet, even between two parties who have never met. **This simple fact is the foundation of secure web browsing and electronic commerce as it is known today.**
Not all applications of SSL/TLS require public trust. For example, a company can issue its own privately trusted certificates for internal use. For more information, please read our article on **Private vs. Public PKI**.

Go to top

SSL/TLS and Secure Web Browsing

The most common and well-known use of SSL/TLS is secure web browsing via the **HTTPS** protocol. A properly configured public HTTPS website includes an SSL/TLS certificate that is signed by a publicly trusted CA. Users visiting an HTTPS website can be assured of:

- **Authenticity.** The server presenting the certificate is in possession of the private key that matches the public key in the certificate.
- **Integrity.** Documents *signed* by the certificate (e.g. web pages) have not been altered in transit by a **man in the middle**.
- **Encryption.** Communications between the client and server are encrypted.

Because of these properties, SSL/TLS and HTTPS allow users to securely transmit confidential information such as credit card numbers, social security numbers, and login credentials over the internet, and be sure that the website they are sending them to is authentic. With an insecure HTTP website, these data are sent as plain text, readily available to any eavesdropper with access to the data stream. Furthermore, users of these unprotected websites have no trusted third-party assurance that the website they are visiting is what it claims to be.
Look for the following indicators in your browser's address bar to be sure that a website you are visiting is protected with a trusted SSL/TLS certificate (screenshot from Firefox 70.0 on macOS) :

- A closed padlock icon to the left of the URL. Depending on your browser and the type of certificate the website has installed, the padlock may be green and/or accompanied by identifying information about the company running it.

- If shown, the protocol at the beginning of the URL should be https://, not http://. Note that not all browsers display the protocol.

Modern desktop browsers also alert visitors to insecure websites that do not have an SSL/TLS certificate. The screenshot below is of an insecure website viewed in Firefox, and shows a crossed-out padlock to the left of the URL:

Go to top

Obtaining an SSL/TLS Certificate

Ready to secure your own website? The basic procedure for requesting a publicly trusted SSL/TLS website certificate is as follows:

- The person or organization requesting the certificate generates a pair of public and private keys, preferably on the server to be protected.
- The public key, along with the domain name(s) to be protected and (for OV and EV certificates) organizational information about the company requesting the certificate, is used to generate a **certificate signing request (CSR)**.
  - **Please see this FAQ for instructions on generating a keypair and CSR on many server platforms.**
- The CSR is sent to a publicly trusted CA (such as SSL.com). The CA validates the information in the CSR and generates a signed certificate that can be installed on the requester's web server.
  - **For instructions on ordering SSL/TLS certificates from SSL.com, please see this how-to.**

SSL/TLS certificates vary depending on the validation methods used and the level of trust they confer, with extended validation (EV) offering the highest level of trust. For information on the differences between the major validation methods (DV, OV, and EV), please refer to our article, **DV, OV, and EV certificates**.