How HTTPS Works

Let me explain how HTTPS works, including SSL, certificates, the encryption process, and the differences between TLS 1.2 and TLS 1.3, in a clear and step-by-step way.

What is HTTPS?

HTTPS stands for **Hypertext Transfer Protocol Secure**. It's the secure version of HTTP, the protocol used to transfer data between your browser and a website. The "secure"part comes from adding a layer of encryption and authentication, provided by **SSL (Secure Sockets Layer)** or its modern replacement, **TLS (Transport Layer Security)**. So, HTTPS is essentially HTTP running over TLS (or historically SSL) to keep your data safe.

How Does HTTPS Work? The TLS Handshake

When you visit a website using HTTPS (e.g., `https://example.com`), your browser and the website's server establish a secure connection before exchanging any data. This process is called the **TLS handshake**. Here's how it works in simple steps:

1. **Client Hello**:
    - Your browser sends a message to the server saying, "Hello! Here's the TLS version I support (e.g., TLS 1.3), the encryption algorithms (cipher suites) I can use, and a random string of bytes (client random)."
2. **Server Hello**:
    - The server replies, "Hello back! I'll use this TLS version and this cipher suite from your list. Here's my random string (server random)."
3. **Certificate**:
    - The server sends its **SSL certificate**, which includes its **public key** and is signed by a trusted **Certificate Authority (CA)**. This certificate proves the server's identity.
4. **Client Key Exchange**:
    - Your browser checks the certificate to ensure it's valid and signed by a trusted CA. If it passes, the browser generates a **pre-master secret**, encrypts it with the server's public key, and sends it to the server.
5. **Session Keys**:

- Both the browser and server use the client random, server random, and pre-master secret to independently generate the same **session key**. This key is used to encrypt and decrypt all data during the session.

6. **Finished**:
    - Both sides send a “finished”message, encrypted with the session key, to confirm the secure connection is ready.

Once the handshake is complete, all data (like web pages, forms, or files) is encrypted with the session key, making it unreadable to anyone who might intercept it.

What Are SSL Certificates and How Do They Work?

An **SSL certificate** is a digital document that proves a website’s identity and enables encryption. Here’s what you need to know:

- **Contents**: The certificate includes the website’s domain name, its public key, and a digital signature from a **Certificate Authority (CA)**.
- **Purpose**: It ensures the server is legitimate (e.g., you’re really connecting to `example.com`, not a fake site) and provides the public key for encryption.
- **Verification**: Your browser checks:
    1. Is the certificate valid (not expired or revoked)?
    2. Is it signed by a trusted CA? (Browsers have a built-in list of trusted CAs, like DigiCert or Let’s Encrypt.)
- If the checks pass, the browser trusts the server and proceeds with the handshake.

The CA acts like a trusted third party vouching for the website. Without this, attackers could pretend to be any site and steal your data.

The Encryption Algorithm

The encryption in HTTPS relies on a combination of **asymmetric** and **symmetric cryptography**:

1. **Asymmetric Cryptography** (during the handshake):
    - Uses a **public key** (shared openly) and a **private key** (kept secret by the server).
    - The browser encrypts the pre-master secret with the server’s public key. Only the server, with its private key, can decrypt it.

- Example algorithms: RSA or Elliptic Curve Cryptography (ECC).

2. **Symmetric Cryptography** (for the session):
   - Once the session key is created, both sides use it to encrypt and decrypt data.
   - This is faster than asymmetric encryption and ideal for large data transfers.
   - Example algorithm: AES (Advanced Encryption Standard).

The handshake uses asymmetric encryption to securely share the session key, then symmetric encryption takes over for efficiency.

Differences Between TLS 1.2 and TLS 1.3

TLS 1.2 and **TLS 1.3** are versions of the TLS protocol, with TLS 1.3 being the newer, improved version. Here are the key differences:

Feature	TLS 1.2	TLS 1.3
Handshake Speed	Multiple round trips between client and server, adding latency.	Fewer round trips, often just one, for faster connections. Can send data immediately ("0-RTT") on reconnections.
Cipher Suites	Supports older, less secure encryption algorithms (e.g., SHA-1).	Removes outdated cipher suites, only uses modern, secure ones (e.g., AES-GCM).
Privacy	Some handshake data (like certificates) is sent in plaintext.	Almost the entire handshake is encrypted, hiding more details from eavesdroppers.
0-RTT Resumption	Not available.	Allows instant data sending on reconnecting, but risks replay attacks if not secured properly.
Security	Vulnerable to certain attacks due to older features.	Stronger by design, removing weak options.

In short, TLS 1.3 is **faster**, **more secure**, and **more private** than TLS 1.2, making it the standard for modern HTTPS.

Putting It All Together

HTTPS secures web communication by: 1. Using the **TLS handshake** to set up an encrypted connection. 2. Relying on **SSL certificates** from trusted CAs to verify the server's identity. 3. Combining **asymmetric cryptography** (for the handshake) and **symmetric cryptography** (for data transfer) to protect your data. 4. Evolving with protocols like **TLS 1.3**, which improves on TLS 1.2 with better speed, security, and privacy.

That's the core of how HTTPS keeps your browsing safe! Let me know if you'd like me to dive deeper into any part.