19Z701 - CRYPTOGRAPHY

Secure Sockets Layer(SSL)

DHEEKSHITHA R	- 22Z216
O KEERTHI	- 22Z243
PRAMODINI P	- 22Z244
S AKASH	- 22Z255
SANJITHA R	- 22Z259
SREERAGHAVAN R	- 22Z261

Contents

Introduction and Overview

- DHEEKSHITHA R (22Z216)

What is SSL ?

- SSL or Secure Sockets Layer, is an Internet security protocol that encrypts data to keep it safe.
- It was created by Netscape in 1995 to ensure privacy, authentication, and data integrity in online communications.
- SSL is the older version of what we now call TLS (Transport Layer Security).
- Websites using SSL/TLS have "HTTPS" in their URL instead of "HTTP"

Where SSL is Used

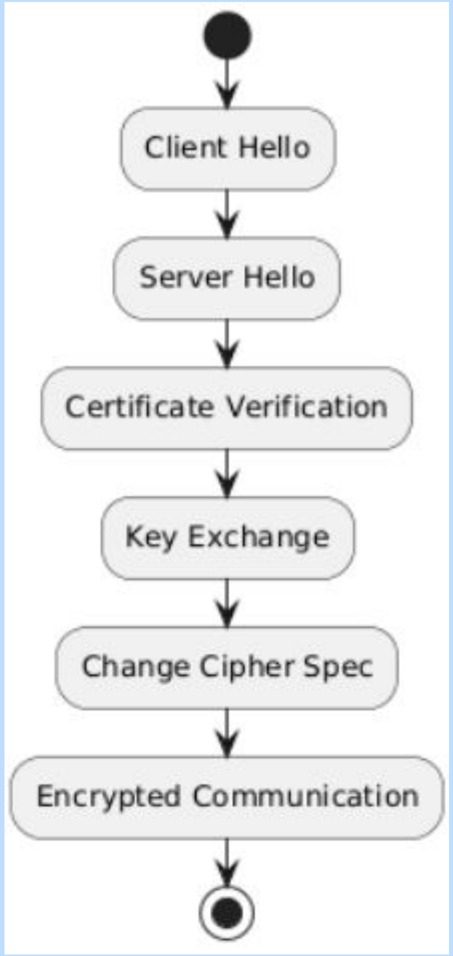
- **Websites (HTTPS):** Protects login forms, personal info, and transactions.
- **Emails:** Secure mail transfer (e.g., SMTPS, POP3S, IMAPS).
- **Online Banking & Payments:** Safeguards financial transactions.
- **Messaging Apps:** Ensures privacy in chats and calls.
- **APIs & IoT Devices:** Protects communication between connected systems.

Working of SSL

Encryption: SSL encrypts data transmitted over the web, ensuring privacy. If someone intercepts the data, they will see only a jumble of characters that is nearly impossible to decode.

Authentication: SSL starts an authentication process called a handshake between two devices to confirm their identities, making sure both parties are who they claim to be.

Data Integrity: SSL digitally signs data to ensure it hasn't been tampered with, verifying that the data received is exactly what was sent by the sender.



Importance of SSL

Originally, data on the web was transmitted in plaintext, making it easy for anyone who intercepted the message to read it.

By encrypting data between a user and a web server, SSL ensures that anyone who intercepts the data sees only a scrambled mess of characters.

Additionally, SSL helps prevent cyber attacks by:

- **Authenticating Web Servers:** Ensuring that users are connecting to the legitimate website, not a fake one set up by attackers.
- **Preventing Data Tampering:** Acting like a tamper-proof seal, SSL ensures that the data sent and received hasn't been altered during transit.

Confidentiality

Data Integrity

Authentication

Trust & Credibility

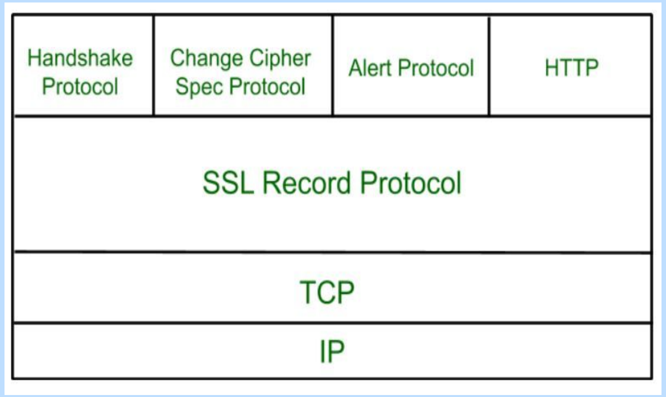
Secure Socket Layer Protocols

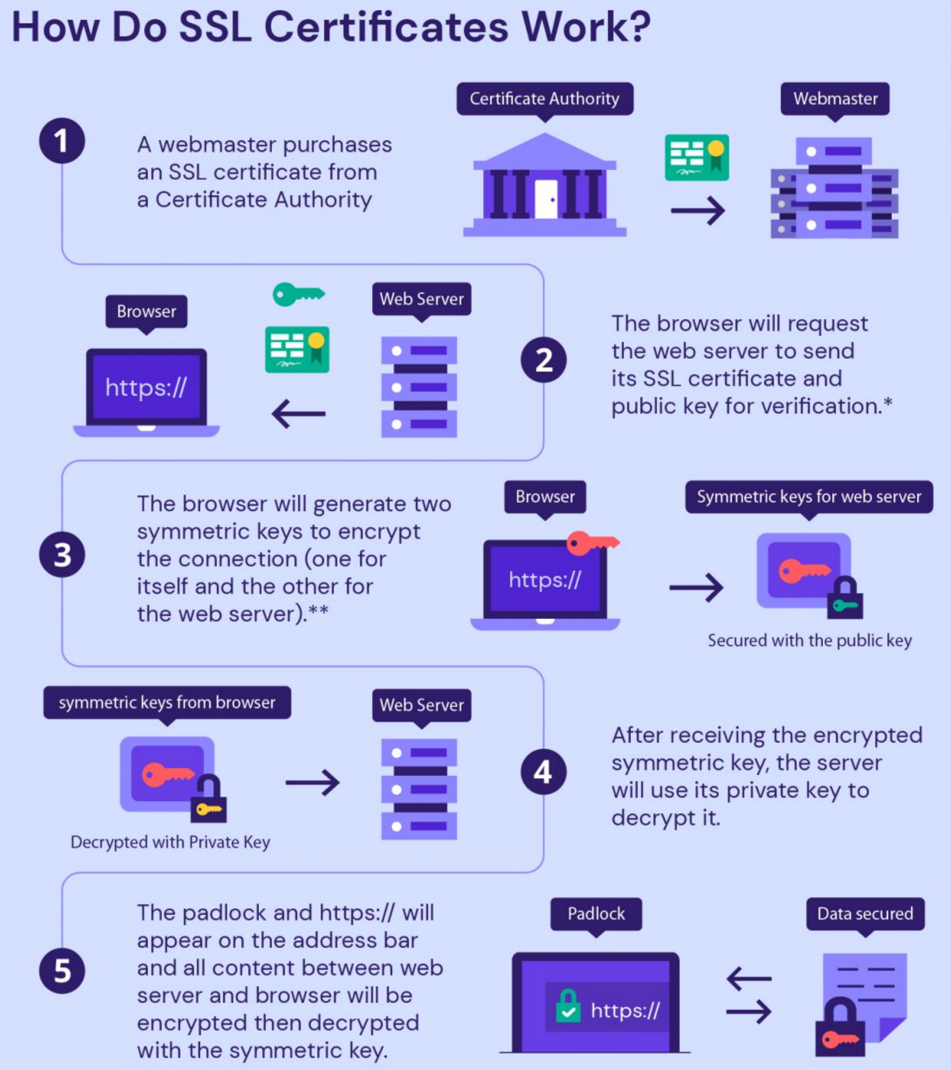
- **SSL Record Protocol:** Ensures secure data transmission by fragmenting, compressing, and encrypting messages before sending them.

- **Handshake Protocol:** Establishes the secure session by authenticating parties and negotiating encryption algorithms and keys.

- **Change-Cipher Spec Protocol:** Signals the transition to the newly negotiated encryption and hashing algorithms during communication.

- **Alert Protocol:** Sends notifications about errors or session status, such as warnings or connection terminations.





How Do SSL Certificates Work?

1 A webmaster purchases an SSL certificate from a Certificate Authority

2 The browser will request the web server to send its SSL certificate and public key for verification.*

3 The browser will generate two symmetric keys to encrypt the connection (one for itself and the other for the web server).**

4 After receiving the encrypted symmetric key, the server will use its private key to decrypt it.

5 The padlock and https:// will appear on the address bar and all content between web server and browser will be encrypted then decrypted with the symmetric key.

Certificate Authority
Webmaster
Browser
https://
Web Server
Browser
https://
Symmetric keys for web server
Secured with the public key
symmetric keys from browser
Decrypted with Private Key
Web Server
Padlock
https://
Data secured

SSL Certificate

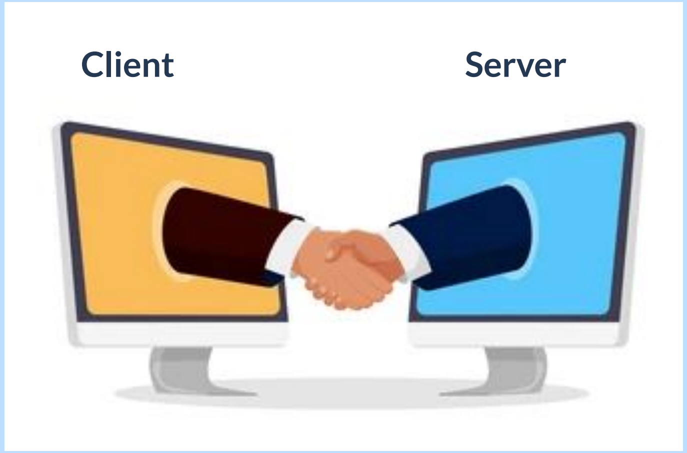
SSL [Secure Sockets Layer] certificate is a digital certificate used to secure and verify the identity of a website or an online service. The certificate is issued by a trusted third-party called a Certificate Authority [CA], who verifies the identity of the website or service before issuing the certificate.

SSL Handshake Process

- SANJITHA R (22Z259)

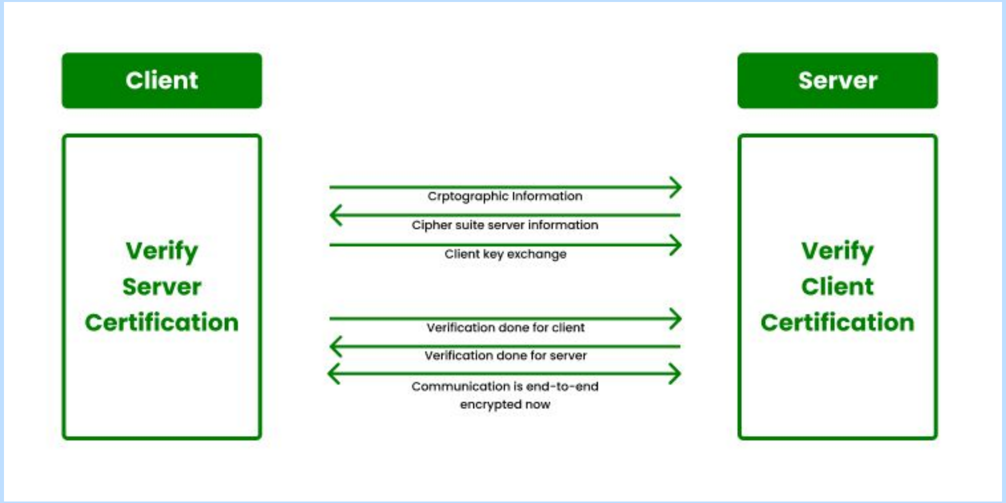
What is SSL Handshake Process?

- Establishes a secure connection between client & server
- Ensures: Authentication, Encryption, Data Integrity
- Process happens before actual data transfer
- Protects sensitive data like passwords, cards, and personal info
- Forms the foundation of secure internet communication (HTTPS)



Steps in SSL Handshake Process

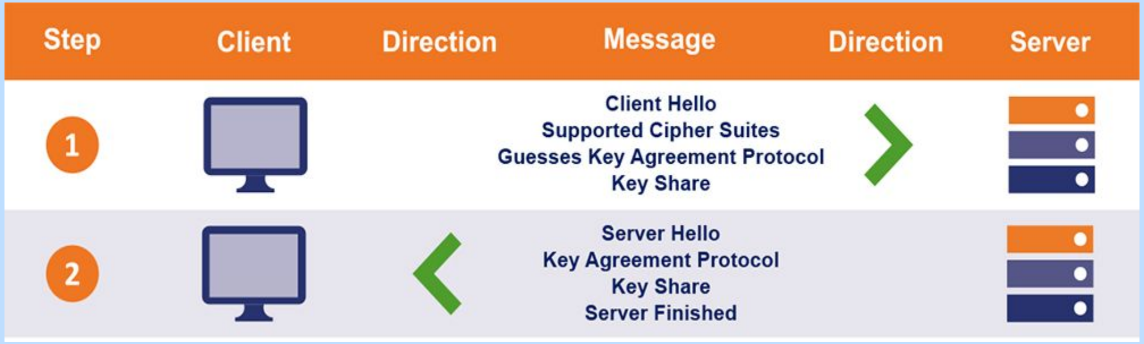
1. Client Hello -> Server Hello
2. Certificate Exchange
3. Session Key Establishment


Client
Server
Verify Server Certification
Verify Client Certification
Crptographic Information
Cipher suite server information
Client key exchange
Verification done for client
Verification done for server
Communication is end-to-end encrypted now

Step 1: Client Hello -> Server Hello

Client Sends:
- Supported cipher suites (types of encryption)
- SSL/TLS version
- Random number

Server Replies With:
- Chosen cipher suite
- Its own random number



Step	Client	Direction	Message	Direction	Server
1			Client Hello Supported Cipher Suites Guesses Key Agreement Protocol Key Share	>	
2		<	Server Hello Key Agreement Protocol Key Share Server Finished		

Step 2: Certificate Exchange

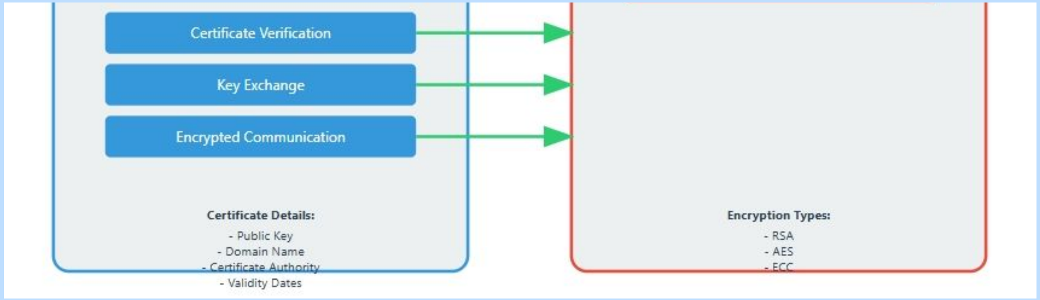
3
Certificate

Server Sends:

- Digital certificate (contains public key)
- Signed by a trusted Certificate Authority (CA)

Client Verifies:

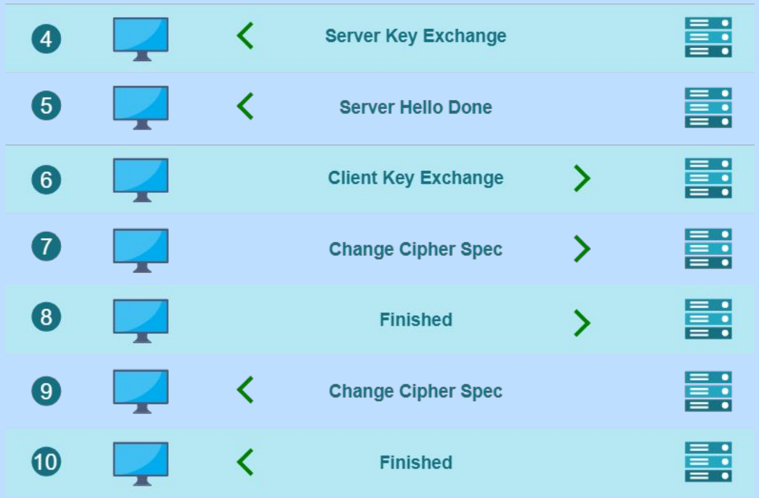
- Validity, expiration and authenticity



Step 3: Session Key Establishment

Process:

1. Client generates a **pre-master secret.**
2. Encrypts it using the **server's public key.**
3. Server decrypts it with its **private key.**
4. Both sides derive the same **session key.**



This **session key** is used for symmetric encryption during communication.

Real - Life Example

Imagine accessing https://www.bankexample.com from a browser:

1. **Client Hello:** The browser says, “Hello Bank! I support TLS 1.3 and these cipher suites: AES-256-GCM, CHACHA20-POLY1305. Here’s a random number to start[Eg. XXX].”

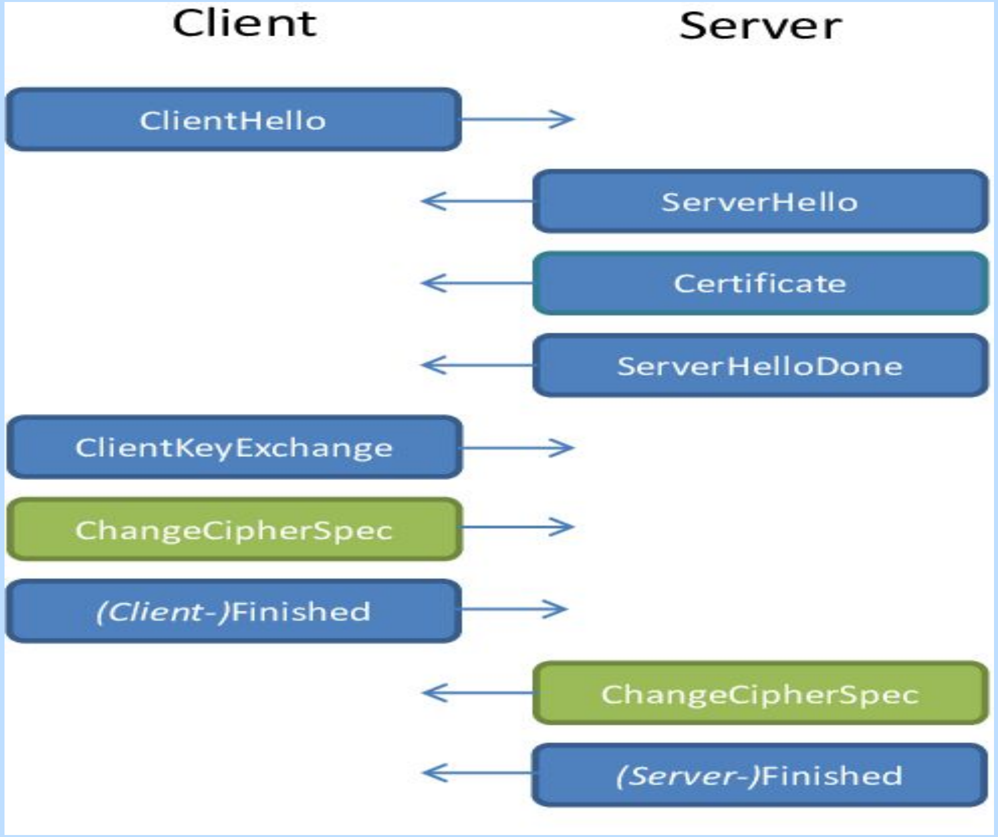
2. **Server Hello:** Bank’s server replies, “I choose TLS 1.3 with AES-256-GCM. Here’s my random number[Eg. YYY].”

3. **Certificate Exchange:** Bank sends its SSL certificate signed by a trusted CA.The browser verifies the certificate to confirm the server is genuine.

4. **Key Exchange:** Your browser and the server securely generate a shared session key (using RSA/ECDHE).

5. **Finished Messages:** Both sides confirm the handshake succeeded. Now all your online banking traffic is encrypted with the session key, keeping your password and account data safe.

Real - Life Example



DEMO

Session Management in SSL

- SREERAGHAVAN R (22Z261)

The Need

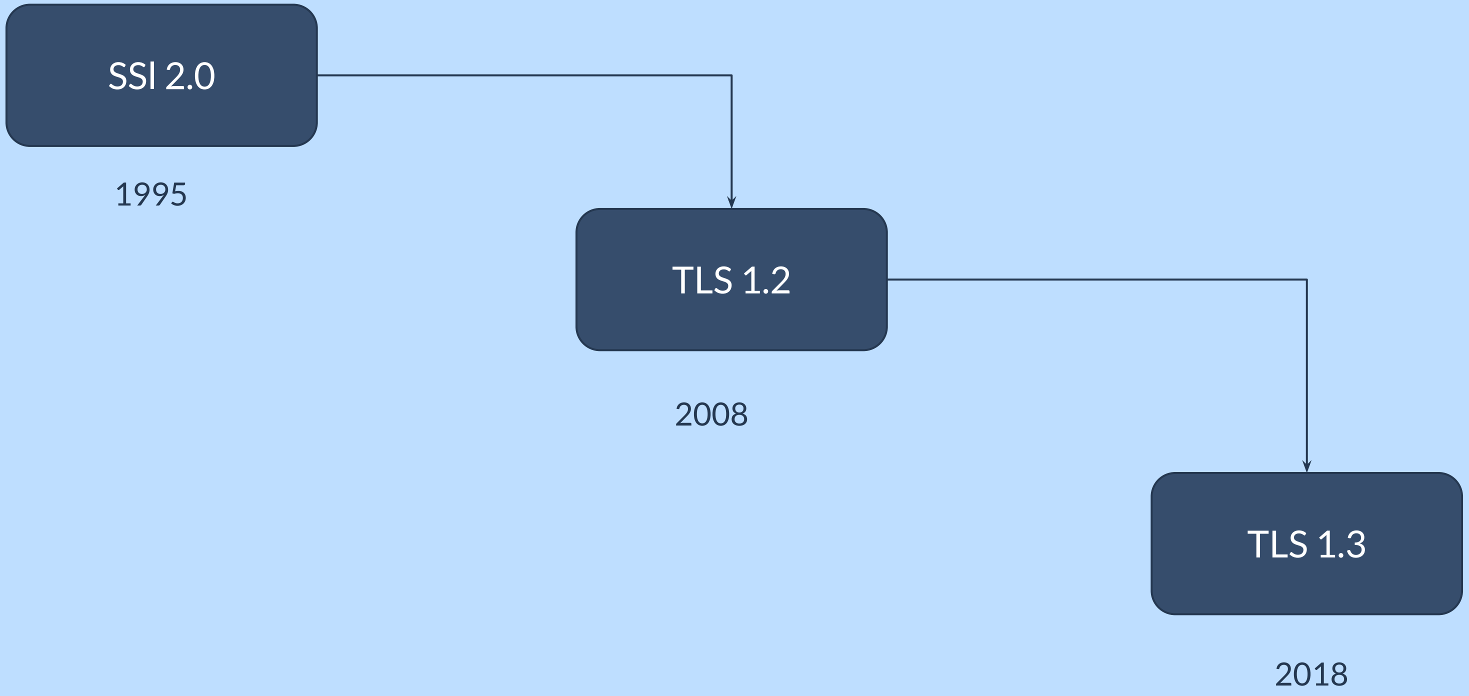
- SSL Handshake is a heavy process and takes ~5 RTT
- This scales terribly
- Only useful for human usage (Still noticeable)
- Redundant actions and messages

Session Contents

Each session has:

- A Session ID
- A cipher suite (encryption and MAC algorithms)
- A master secret (the main key material)
- Random values exchanged by client and server
- The compression method (rarely used now)
- The certificate and keys used for authentication

History



Session IDs

- Introduced in SSl 2.0
- Store session data on the server and send session id to client
- Client uses session id to restore session
- Highly centralized

Session Tickets

- Introduced in TLS 1.2
- Uses PKI to store the session state
- Store the session state in a blob (Ticket)
- Client sends the ticket to the server
- Server uses its private key to retrieve the session state
- Server sends session state to client and session resumes

Pre shared keys

- Introduced in TLS 1.3
- An advancement of the Session Ticket idea
- Store a symmetric shared key based on handshake
- Allows for 0-RTT handshakes

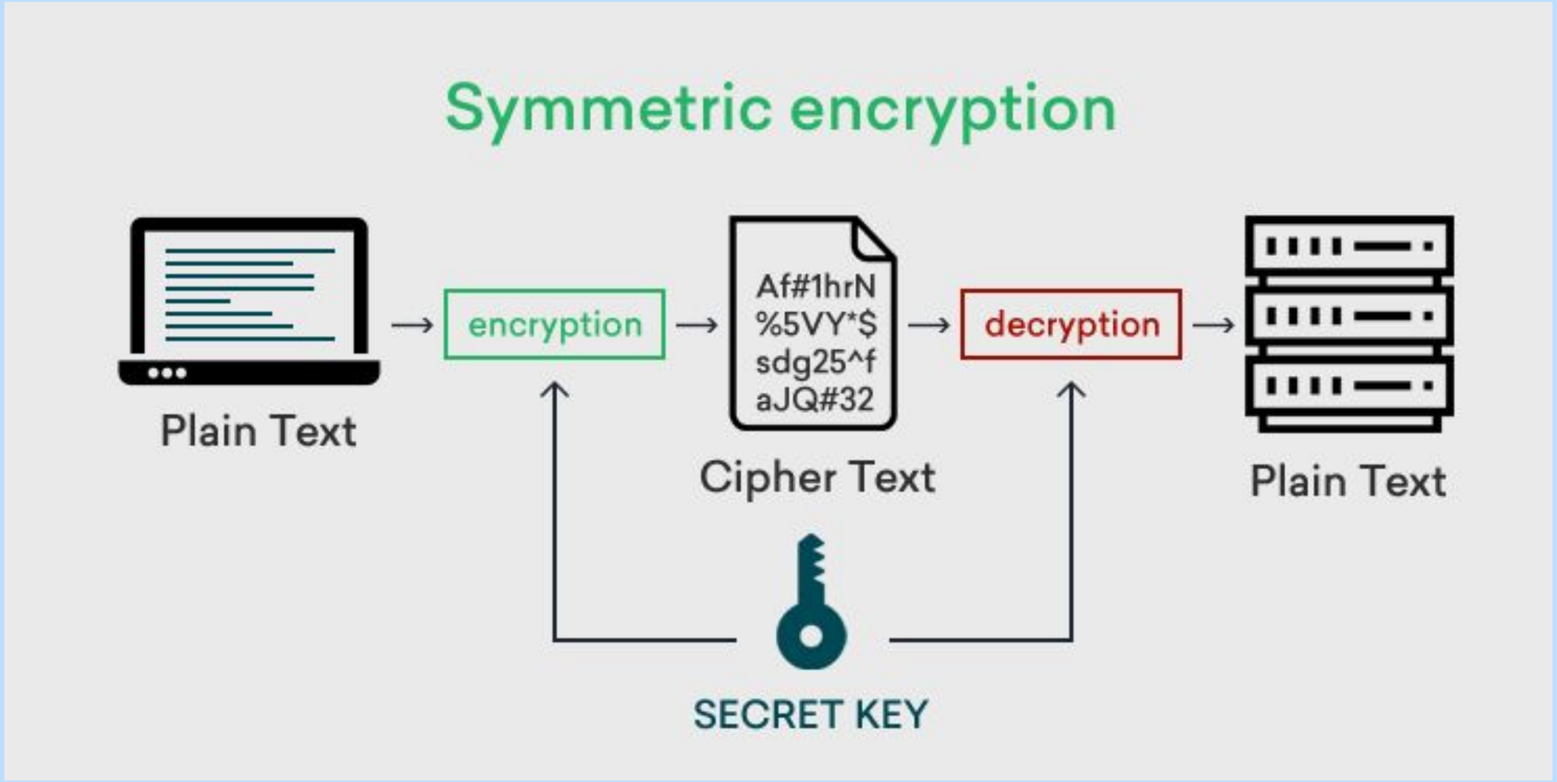
DEMO

Cryptographic Techniques in SSL

- PRAMODINI P (22Z244)

How SSL Keeps Internet Communication Secure

Symmetric Encryption



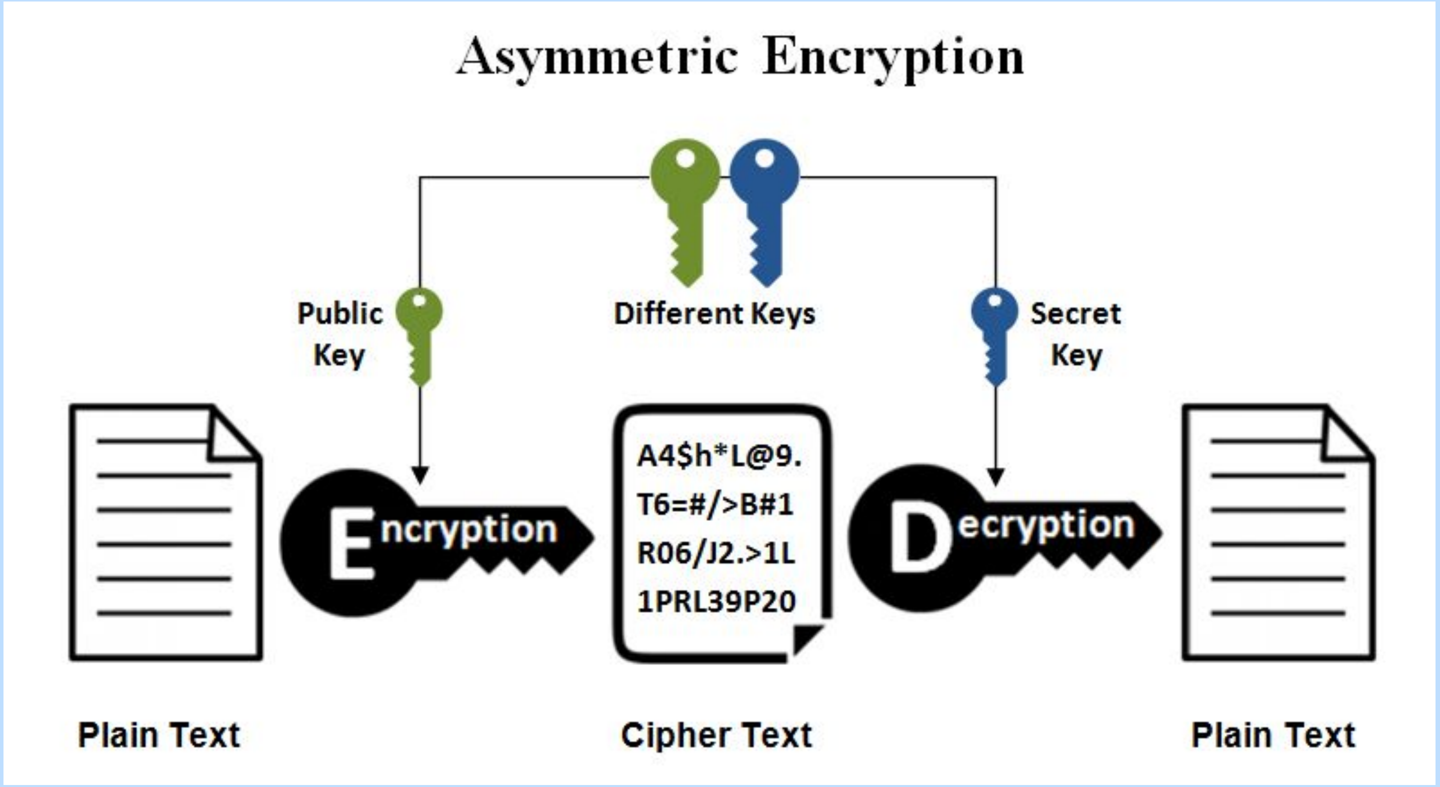
Symmetric Encryption

Both sender and receiver use the **same secret key** for encryption and decryption. It's fast and efficient , perfect for securing large amounts of data once the session key is established.

- One shared key for both ends
- Fast and low computational cost
- **Main challenge:** securely sharing the key

Example:
AES (Advanced Encryption Standard) used for encrypting data after the SSL handshake.

Asymmetric Encryption



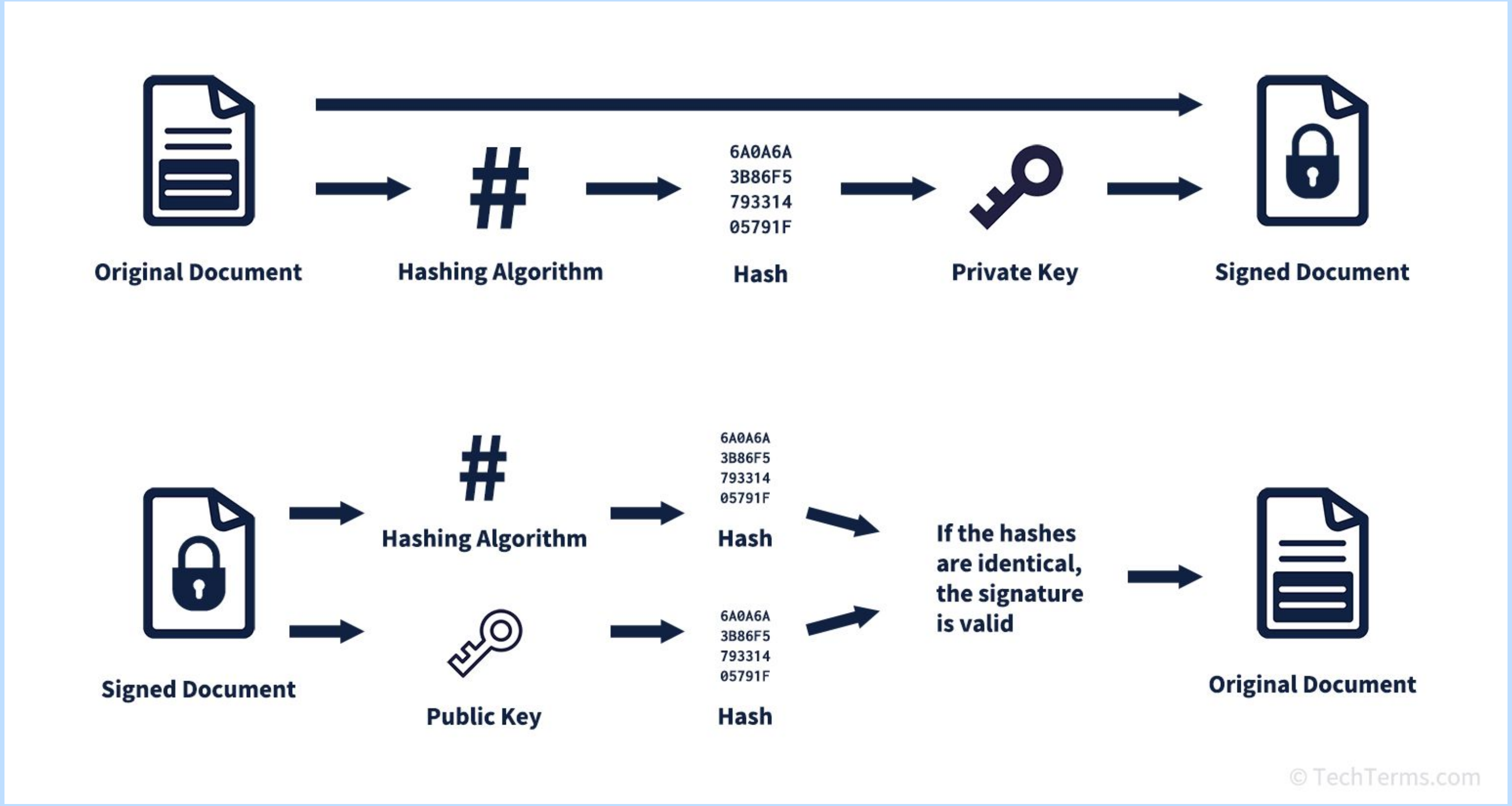
Asymmetric Encryption

Uses a **pair of keys**, one **public** and one **private** to establish secure communication without sharing secrets beforehand.It solves the key distribution problem in symmetric encryption.

- **Public key:** shared openly
- **Private key:** kept secret by the owner
- Enables secure key exchange and authentication

Example:
RSA algorithm used during the SSL handshake phase.

Digital Signatures



Digital Signatures

Digital signatures verify **who sent the message** and ensure it hasn't been altered. They use the sender's **private key to sign** and the receiver's **public key to verify**.

- Confirms sender identity
- Ensures data integrity
- Prevents denial of sending (non-repudiation)

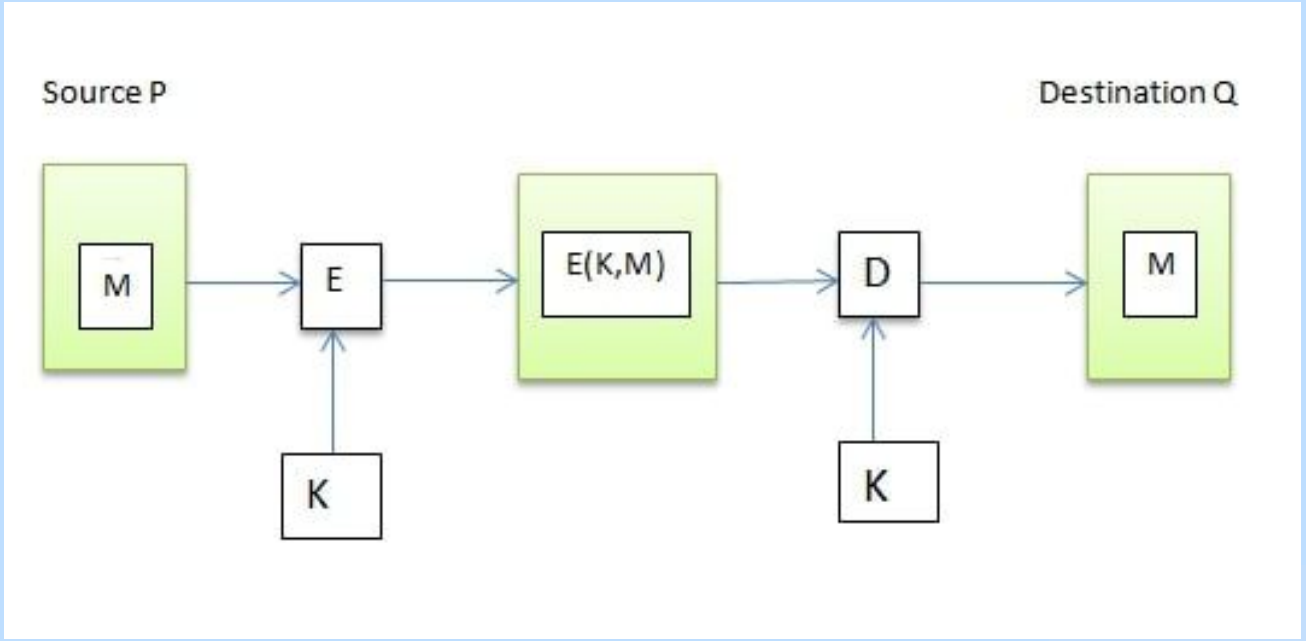
Example:
SSL certificates are digitally signed by trusted Certificate Authorities (CAs).

Message Authentication Code

A MAC combines a secret key with a message to produce a short code that ensures data integrity and authenticity. If the message is altered during transmission, the MAC value will no longer match, immediately revealing tampering

- Detects unauthorized message changes
- Provides integrity and authentication
- Commonly used in symmetric encryption protocols like AES-CMAC and CBC-MAC

Message Authentication Code



SSL in Action

Handshake Phase (Asymmetric Encryption)

Establishes trust using public/private keys.

Securely exchanges a session key.

Session Phase (Symmetric Encryption)

Provides fast and private data transfer using the shared key.

Authentication & Integrity

Digital Signatures → verify the sender's identity.

HMAC → ensures data is not altered during transfer.

Layered Defense System

Ensures confidentiality, authentication, and integrity.

Protects every SSL connection from interception and tampering.

SSL vs TLS

- S AKASH (22Z255)

Security and Encryption Algorithms

SSL uses **RC4** and **3DES** which are susceptible to attacks.

TLS utilizes stronger **AES-GCM** and **ChaCha20-Poly1305**, offering robust security

TLS also introduces better key exchange algorithms like ECDHE and ECC, enabling perfect forward secrecy, making previous sessions secure even if a key is compromised.

Handshake Process and Protocol Negotiation

SSL handshake has more steps, making it slower and less efficient. Moreover it’s explicit

TLS handshake removes unnecessary steps, speeds up the process, and negotiates encryption settings at the start of the connection.

Authentication and Compatibility

SSL supports basic server authentication,

TLS enables both client and server authentication. This increases security, and compatibility with modern web applications.

TLS is designed to be backward-compatible with SSL for transition purposes, but modern browsers lack support for SSL.

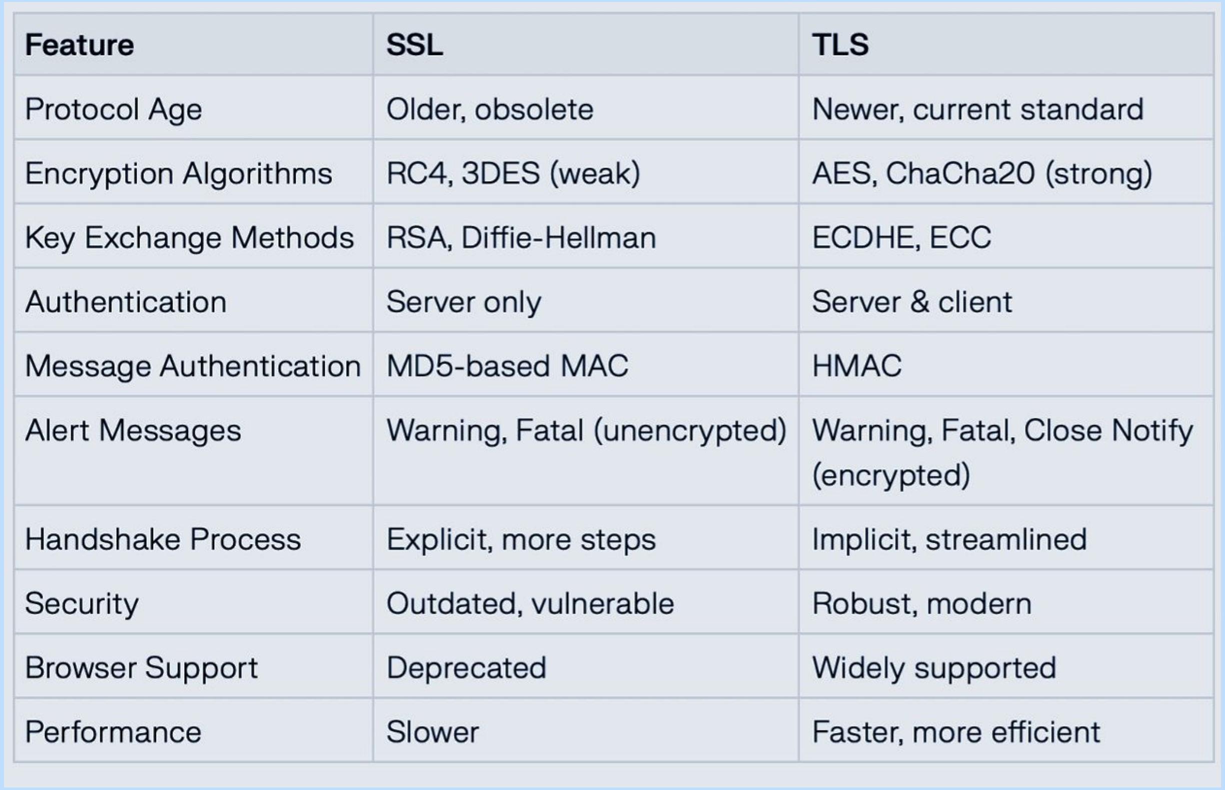
Message Authentication and Alerts

SSL relies MD5 for message authentication codes (MACs).

TLS uses HMAC, a more secure cryptographic method.

SSL alerts are unencrypted and limited to two types, whereas TLS encrypts alerts and adds extra message types such as “close notify” for better communication.

Recap



Feature	SSL	TLS
Protocol Age	Older, obsolete	Newer, current standard
Encryption Algorithms	RC4, 3DES (weak)	AES, ChaCha20 (strong)
Key Exchange Methods	RSA, Diffie-Hellman	ECDHE, ECC
Authentication	Server only	Server & client
Message Authentication	MD5-based MAC	HMAC
Alert Messages	Warning, Fatal (unencrypted)	Warning, Fatal, Close Notify (encrypted)
Handshake Process	Explicit, more steps	Implicit, streamlined
Security	Outdated, vulnerable	Robust, modern
Browser Support	Deprecated	Widely supported
Performance	Slower	Faster, more efficient

Vulnerabilities and Applications of SSL

- O KEERTHI (22Z243)

SSL Vulnerabilities: Overview

Even SSL/TLS can be attacked if misconfigured or outdated.

Key weaknesses often arise from:

→ Old protocol versions (SSL 2.0 / 3.0)

→ Weak cipher suites

→ Implementation bugs (e.g., OpenSSL flaws)

Common SSL Attacks

1. POODLE Attack (2014)

- Exploits fallback to SSL 3.0.
- Allows decryption of secure cookies.

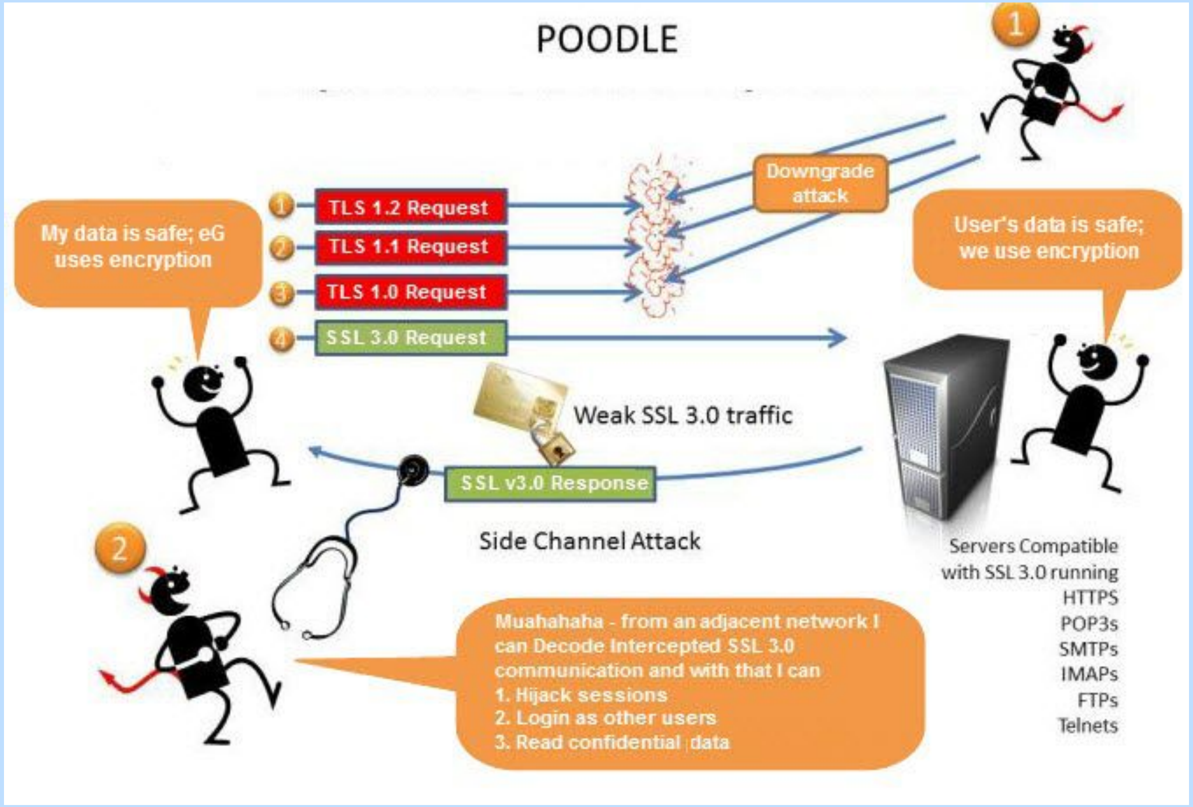
2. BEAST Attack (2011)

- Targets TLS 1.0 using CBC mode vulnerability.
- Enables attackers to decrypt HTTPS traffic.

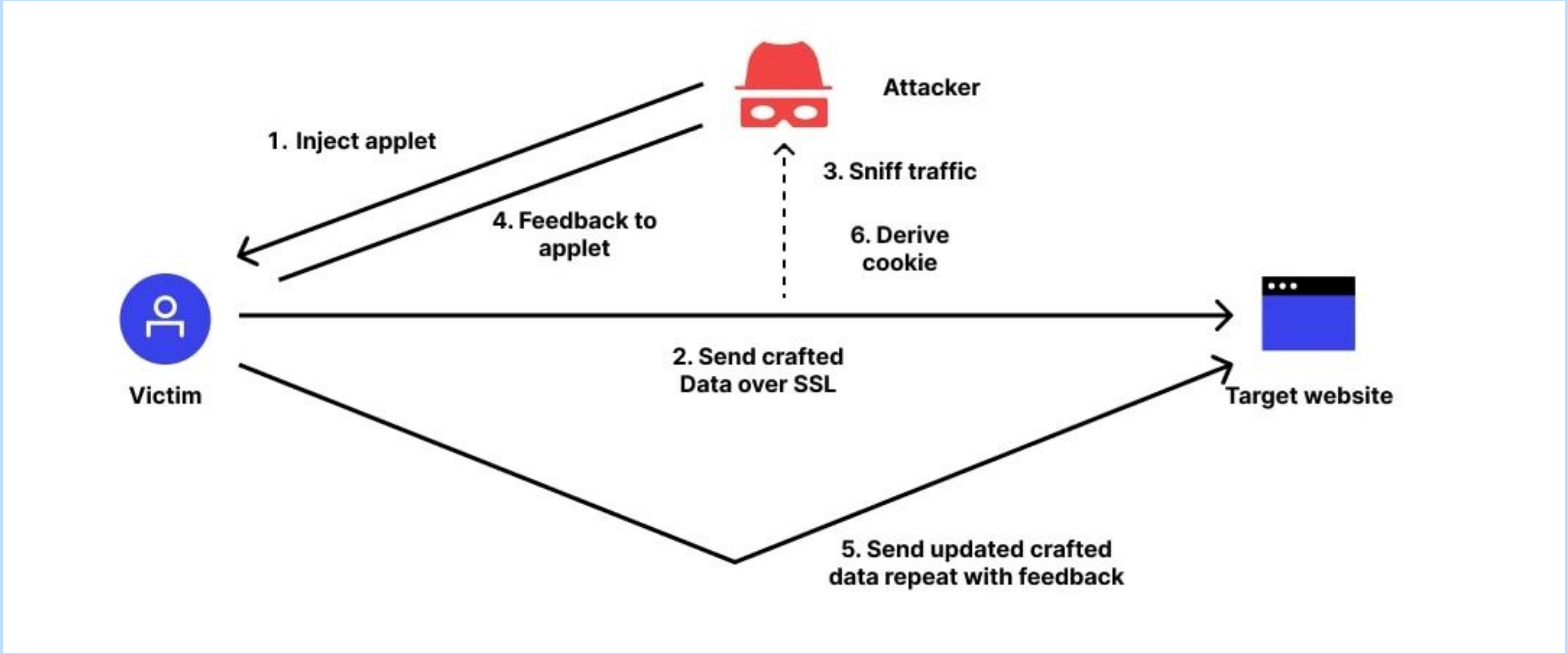
3. Heartbleed (2014)

- Bug in OpenSSL's heartbeat extension.
- Exposes server memory and sensitive keys.

Common SSL Attacks


POODLE
1
Downgrade attack
1 TLS 1.2 Request
2 TLS 1.1 Request
3 TLS 1.0 Request
4 SSL 3.0 Request
My data is safe; eG uses encryption
User's data is safe; we use encryption
Weak SSL 3.0 traffic
SSL v3.0 Response
Side Channel Attack
2
Muahahaha - from an adjacent network I can Decode Intercepted SSL 3.0 communication and with that I can
1. Hijack sessions
2. Login as other users
3. Read confidential data
Servers Compatible with SSL 3.0 running
HTTPS
POP3s
SMTPs
IMAPs
FTPs
Telnets

Common SSL Attacks



Common SSL Attacks

- In August 2014, Community Health Systems (a major US-for-profit hospital chain) was reported to have its systems breached via Heartbleed: hackers stole the private keys of their systems and compromised about **4.5 million patient records**.

- Another example: The Canada Revenue Agency (CRA) reported that about 900 social insurance numbers were stolen over a 6 hour period via exploitation of Heartbleed

Mechanics of one attack instance:

- Attacker sends a “heartbeat” request to the vulnerable server saying “here’s a payload of size 65 535 bytes” but only supplies a small amount.

- Server blindly reads up to 65 535 bytes from memory following the payload buffer and returns it. That memory might contain passwords, private keys, session cookies.

- Because the heartbeat exchange often isn’t logged as a standard request, the attack can leave **no trace**, making forensic detection harder.

Mitigation Strategies

1. **Disable outdated SSL versions** (SSL 2.0, SSL 3.0).
2. **Use strong ciphers** (AES, SHA-256, forward secrecy).
3. **Update OpenSSL libraries** regularly.
4. **Apply frequent security patches** and monitor logs.

Real-World Applications of SSL

1. HTTPS (Web Security)

- Ensures encrypted connection between browser & server.
- Prevents eavesdropping and data tampering.

2. Secure Email Protocols

- **SMTPS / IMAPS / POP3S** use SSL/TLS to encrypt mail transfer.
- Protects credentials and message content.

3. Online Banking & E-Commerce

- Safeguards user credentials, payment details, and transactions.
- Builds user trust and prevents data breaches.

Thank you!