CS 409: Introduction to Cryptography

November 27, 2024

Chalk & Talk: Transport Layer Security

Instructor: Sruthi Sekar **Submitted by:** Pinak Mahapatra and Karrthik Radhakrishnan

1 Transport Layer Security

Transport Layer Security (TLS) is a critical security protocol widely adopted to ensure privacy, integrity, and data security for communications over the Internet. It evolved from its predecessor, Secure Sockets Layer (SSL), and is now the standard for encrypting data in transit. TLS works by establishing a secure channel between two communicating parties, such as a web browser and a web server, ensuring that sensitive data like login credentials, financial information, and personal details are protected from unauthorized access and tampering during transmission.

The protocol achieves this through a combination of cryptographic techniques, including symmetric encryption for data confidentiality, public key cryptography for secure key exchange, and message authentication codes (MACs) for data integrity. A widely recognized application of TLS is encrypting the connection between web browsers and websites, enabling the "HTTPS" prefix in URLs that signifies a secure communication channel.

Beyond securing web traffic, TLS is also integral to encrypting other types of communication, such as email (via protocols like SMTP, IMAP, and POP3), messaging services, and voice-over-IP (VoIP) systems, protecting them from eavesdropping and interception. Furthermore, TLS plays a key role in securing APIs, file transfers, and many other internet-based interactions, making it an indispensable part of modern cybersecurity frameworks. Its flexibility, efficiency, and strong encryption mechanisms have made TLS a cornerstone of trust in online communication.

2 TCP Three-Way Handshake

The **TCP handshake** is a process used in the Transmission Control Protocol (TCP) to establish a reliable connection between a client and a server before data transfer begins. It ensures both sides are synchronized and ready to communicate. The process is called the **three-way handshake** because it involves three steps:

2.1 Steps of the Three-Way Handshake

SYN (Synchronize): The client sends a `SYN` (synchronize) packet to the server. This packet indicates that the client wants to establish a connection and includes an initial sequence number (`ISN`), which is used to track the data sent from the client.

SYN-ACK (Synchronize-Acknowledge): The server responds with a `SYN-ACK` packet. This packet acknowledges the client's `SYN` request and includes the server's own initial sequence number.

ACK (Acknowledge): The client sends an `ACK` packet back to the server. This acknowledges the server's `SYN-ACK` response, completing the handshake and establishing the connection.

The process is illustrated as follows:

$$\textbf{Step 1:}\quad \text{Client} \xrightarrow{\text{SYN (Sequence = X)}} \text{Server}$$

$$\textbf{Step 2:}\quad \text{Server} \xrightarrow{\text{SYN-ACK (Sequence = Y, Acknowledgment = X+1)}} \text{Client}$$

$$\textbf{Step 3:}\quad \text{Client} \xrightarrow{\text{ACK (Sequence = X+1, Acknowledgment = Y+1)}} \text{Server}$$

3 Certificate Check

A certificate check is a process used in secure communication protocols (e.g., HTTPS) to verify the authenticity and validity of a digital certificate provided by a server. The client ensures the certificate is issued by a trusted Certificate Authority (CA), checks its expiration date, and confirms that the domain name matches the certificate. This process helps establish a secure connection by ensuring the server's identity is legitimate and preventing potential attacks, such as man-in-the-middle attacks.

3.1 CLIENT Hello

The session begins by the client saying hello, providing the server with the following information:

- **Client Random Data:** A random 32-byte value generated by the client, later combined with a similar random value from the server to create a shared session key for encrypting data during the session.
- **Cipher Suites Supported by the Client:** A list of cipher suites (encryption algorithms, hash functions, and key exchange mechanisms) that the client supports, such as AES and ChaCha20 for encryption, or ECDHE for key exchange. The server selects one it also supports to establish a common encryption protocol.
- **Supported Public Key Types for Key Exchange:** A list of public key algorithms the client supports, like RSA, DSA, and ECDHE, to securely exchange keys. The server picks one for negotiating the shared key.
- **Supported Protocol Versions:** A list of TLS protocol versions the client can use, such as TLS 1.2 and TLS 1.3. The server chooses the highest version it also supports, enabling the strongest security features available.

3.2 SERVER Hello

The server says "Hello" back. The server provides information including the following:

- **Server Random Data:** A random value generated by the server, combined with the client's random data to create the shared session key.
- **Selected Cipher Suite:** A cipher suite chosen by the server from the list provided by the client.
- **Public Key for Key Exchange:** The server's public key that will be used for secure key exchange.
- **Negotiated Protocol Version:** The highest protocol version that both the client and server support, which will be used for the communication.

3.3 Server Certificate

The server sends its digital certificate, issued by a trusted Certificate Authority (CA), which contains the server's public key and identity information. Upon receiving the certificate, the client performs several validation checks to ensure its authenticity. These checks include verifying that the CA is trusted, confirming that the certificate has not expired, ensuring that the certificate matches the server's domain, and validating the signature on the certificate. If all these checks pass successfully, the client trusts the server's identity, allowing the handshake to proceed and ensuring that secure communication can take place.

4 Key Exchange

The client and server each generate their own private keys by creating random 32-byte (256-bit) data within the range of 2^0 to $2^{256} - 1$. These keys are referred to as the client's private key and the server's private key, respectively. Using the X25519 algorithm, both the client and server derive their corresponding public keys from their private keys. As a result, both parties possess a key pair: one private key, which remains confidential, and one public key, which is shared for secure communication.

4.1 X25519 Algorithm

4.1.1 Key Generation

If K_a is a client private key, then the client public key can be calculated as:

$$K'_a = K_a \cdot P$$

where P is a point on the curve where $x = 9$.

4.1.2 Background on Elliptic Curves

Elliptic curves used in cryptography are defined by a specific type of equation. For the purposes of X25519, the curve in use is called *Curve25519*, a type of elliptic curve known as a Montgomery curve. The equation for this curve differs from the classic elliptic curve equation. However, to simplify, we focus on point addition principles using the Weierstrass form:

$$y^2 = x^3 + ax + b$$

4.1.3 Point Addition on an Elliptic Curve

Adding Two Distinct Points: If $P \neq Q$, the line through P and Q intersects the curve at one other point. Reflecting this point over the x-axis gives the sum $R = P + Q = (x_3, y_3)$. The formulas are:

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2$$

$$y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1$$

Doubling a Point: If $P = Q$, the tangent line at P intersects the curve at another point. Reflecting this point over the x-axis gives $2P$. The formulas for point doubling are:

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1$$

$$y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1$$

where a is a constant defined by the curve equation.

4.1.4 Example Calculation

Scalar multiplication involves adding a point P to itself multiple times. To calculate $k \cdot P$ (e.g., for a private key k), we add P to itself k times. Assume the curve equation $y^2 = x^3 + ax + b$, base point $P = (2, 3)$, and scalar $k = 5$. We calculate $5P$ step by step:

1. Calculate $2P$ using point doubling formulas.
2. Add P to $2P$ to get $3P$.
3. Continue adding points until reaching $5P$.

4.2 Shared Key Calculation

The shared secret key can be calculated by combining the public and private keys. The server computes the shared key by combining its private key with the client's public key, while the client computes it by combining its private key with the server's public key. This gives the same final result, which is the shared secret key. Both the client and server have access to this without any information leak.

- k_a: Alice's secret key – a 256-bit (32-byte) random number.
- k_b: Bob's secret key – a 256-bit (32-byte) random number.
- P: A point on the curve, where $x = 9$.
- $k_a \cdot P$: Alice's public key.
- $k_b \cdot P$: Bob's public key.
- $k_a \cdot (k_b \cdot P) = k_b \cdot (k_a \cdot P)$: The shared secret.

Once Alice and Bob have a shared secret, they can use it to build encryption keys for a secure connection.

4.3 Server Handshake Finished

The *Server Handshake Finished* step is the final stage of the TLS handshake, ensuring secure communication between the client and server. In this step, the server sends a `Finished` message, which includes a cryptographic hash of all prior handshake messages. This serves as proof that the handshake was completed correctly and without tampering. The client validates this message using the established session keys, confirming the server's authenticity. Once verified, the client responds with its own `Finished` message, completing the handshake. At this point, both parties transition to secure communication, encrypting all further data using the shared session key.

4.4 ChangeCipherSpec Message

4.4.1 Purpose

The *ChangeCipherSpec* message informs the recipient that all further messages will be encrypted and authenticated using the new keys and cipher suite that have been negotiated during the handshake. This ensures that both parties are synchronized regarding the security parameters of the session.

4.4.2 Timing

The *ChangeCipherSpec* message is sent by both the client and the server after key exchange and authentication steps are completed, but before any application data is transmitted. The typical order of messages is as follows: the client first sends the *ChangeCipherSpec* message, followed by a *Finished* message to confirm completion of its part of the handshake. Subsequently, the server responds with its own *ChangeCipherSpec* message and follows with a *Finished* message. This sequence ensures that both parties acknowledge that the newly negotiated parameters are now in effect and that subsequent communications will be secure.

5 References

- TLS 1.3 Server Key Exchange Generation.
  Available at: `https://tls13.xargs.org/#server-key-exchange-generation`
- TCP 3-Way Handshake Process.
  Available at: `https://www.geeksforgeeks.org/tcp-3-way-handshake-process/`
- Use of GPT: ChatGPT Shared Reference.
  Available at: `https://chatgpt.com/share/672a9292-ec50-8009-b991-7ab223532d55`