INTRODUCTION TO SSL/TLS PROTOCOL

Secure Sockets Layer (SSL) is a security protocol that provides privacy, authentication, and integrity to Internet communications, and was first developed in 1995 by NetScape. SSL eventually evolved into Transport Layer Security (TLS) as an update in 1999.

Websites using SSL or TLS have http**S** in their URLs instead of http

- Encrypts data
- When intercepted, displays garbled characters, nearly impossible to decrypt
- Has authentication handshake process between communicating devices
- Signs data to verify it has not been tampered with

PLAINTEXT did not conceal information and data, like a credit card number on an online shopping website.

SSL is not up to date. It is important to note that many browsers use TLS, but refer to the protocol as SSL due to how easily recognizable it is.

Websites using TLS have an **SSL Certificate**

- Website's public key is used for encryption and authentication with user
- Website's private key decrypts data encrypted with the public key.

SSL STEPS

1. Client initiates a secure connection: When a user types in a URL that begins with "https" instead of "http," the web browser sends a request to the server to establish a secure connection.
2. Server sends its SSL certificate: The server sends its SSL certificate, which includes a public key, to the client.
3. Client verifies the SSL certificate: The client verifies the SSL certificate to ensure that it is valid and issued by a trusted Certificate Authority (CA).
4. Client generates a symmetric key: The client generates a symmetric key, which will be used to encrypt and decrypt data exchanged between the client and server.
5. Client encrypts the symmetric key: The client encrypts the symmetric key using the server's public key from the SSL certificate and sends it to the server.
6. Server decrypts the symmetric key: The server decrypts the symmetric key using its private key.
7. Secure communication: Once the symmetric key is exchanged, the client and server use it to encrypt and decrypt data exchanged between them. This ensures that any data

transmitted between the client and server is secure and cannot be intercepted or tampered with by anyone else.

SYMMETRIC KEY

- Same key for encryption and decryption
- Same key by sender and receiver, in other words
- SYMMETRIC KEY ALGORITHM
    - plaintext message is encrypted into ciphertext using the symmetric key, and then the ciphertext is decrypted back into plaintext using the same key
    - this key is shared by sender and receiver but not by anyone else, keeping the connection secure
    - faster than asymmetric key algorithms (like RSA, using diff keys for encryption and decryption)
    - problem is how to share the key between sender and receiver
        - shared during initial communication using asymmetric key algorithm like RSA

ASYMMETRIC KEY

- public key cryptography
- Uses a pair: one public and one private key
- You can encrypt a message using a user's public key that they decrypt with their related private key
- If a message is encrypted with a sender's private key, it is decrypted with the related public key
- MOST SECURE (users never share their own private key)
    - Not possible for key to be discovered since it is never transmitted
- plaintext → ciphertext → plaintext
- Used to authenticate digital signatures

RSA

- most widely used asymmetric algorithm
- Embedded in ssl and tls
- derives its security from the computational difficulty of factoring large integers that are the product of two large prime numbers
- EASY TO MULTIPLY, DIFFICULT TO FACTOR
    - Can not easily factor the product of two very large primes

    - keys are typically 1024 or 2048 bits long, but experts believe 1024-bit keys will be broken soon, which is why government and industry are moving to a minimum key length of 2048-bits

Alternative: ECC

- **gaining favor with many security experts as an alternative to RSA**
- elliptic curve theory/EC equation
- attacker must compute an elliptic curve discrete logarithm, which is significantly more difficult problem than factoring (more security means keys can be smaller than those of RSA)

Behind RSA

- Idea first proposed in 1977 by professors at stanford
- numbers raised to specific powers to produce decryption keys
- Then..
- RSA invented in 1977 by rivest, shamir, and adelman at mit(rsa)

THE TLS HANDSHAKE (the initial asymmetric sharing of keys)

Client 1: g, n, x.

g and n can be exposed. **x has to be private**

client 1 sends hello to client 2

hello contains g, n

and **g^x % n = eq1**

(DIFFICULT/IMPOSSIBLE TO EXTRACT X)

Client 2: receives **eq1,** g, and n.

creates private number y

client 2 creates eq2 = eq1^y % n = g^(xy) % n

eq2 becomes the input for the cipher algorithm client 2 sends hello with g^y % n

hard to extract y

client 1 does similar process and ends up with

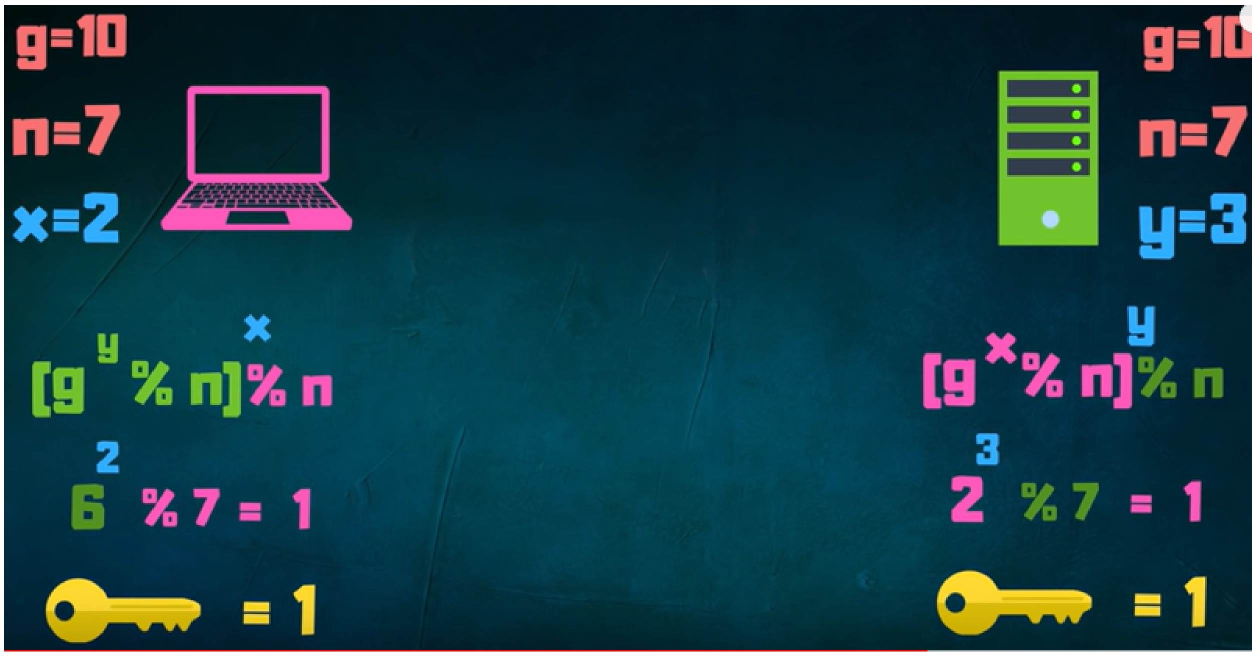
g^xy % n

- SAME INPUT: YOU HAVE SUCCESSFULLY SHARED THE KEY FOR THE ALGORITHM

This exchange is called Diffie-Hellman

The larger n is, the higher the security

Example:



The numbers are exchanged based on the formulas, and the same key is reached. However, private values of x=2 and y=3 are never directly shared