Why HTTPS?

It is used for three main reasons:

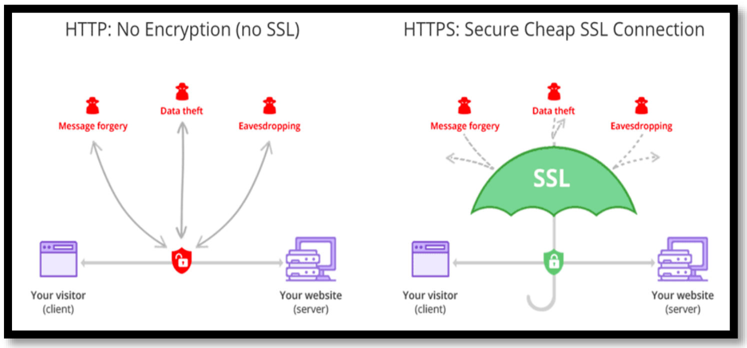

Fig 1: HTTP vs HTTPS

- *Confidentiality*. This protects the communication between two parties from others within a public medium such as the Internet. For example, without HTTPS, someone running a Wi-Fi access point could see private information that was being handled on our website.
- *Integrity*. This makes sure information reaches its destined party in full and unaltered. For example, an attacker could alter the contents of our webpage by reducing the quality of our images to save bandwidth or change the content of articles we read. HTTPS ensures that the website can't be modified.
- *Authentication*. This ensures that the website is what it claims to be. HTTPS ensures that our website that says **saaki.com** is **saaki.com**. Some certificates even check the legal identity behind that website.

How it's done:

The four components of HTTPS require encryption:

- *The initial key exchange*: This uses asymmetric (private and public key) algorithms.
- *The identity certification*: (the HTTPS certificate, issued by a certification authority) This uses asymmetric (private and public key) algorithms.
- *The actual message encryption*: This uses symmetric (pre-shared secret) algorithms.
- *The message digesting:* This uses cryptographic hashing algorithms.

For example, the recommended setting ***ECDHE-RSA-AES256-GCM-SHA384*** means that the key will be exchanged using the Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) key exchange algorithm; the CA signed the certificate using the Rivest-Shamir-Adleman (RSA) algorithm; the symmetric message encryption will use the Advanced Encryption Standard (AES) cipher, with a 256-bit key and GCM mode of operation; and message integrity will be verified using the SHA secure hashing algorithm, using 384-bit digests. Thus, we are, at any point not using outdated crypto.

Configuration choices we will be using:

CIPHER SUITES

Deciding the cipher suites to use is a balance between compatibility and security:

- Compatibility with older browsers needs the server to support older cipher suites.
- However, many older cipher suites are no longer considered secure.

OpenSSL lists the supported combinations in order of cryptographic strength, with the most secure at the top and the weakest at the bottom. It is designed in this way because, during the initial handshake between the client and the server, the combination to be used is negotiated until a match is found that is supported by both parties.

KEY TYPES

We can choose from one of the following key types:

Elliptic Curve Cryptography (ECC) certificates are faster and use less CPU than the RSA certificates, which is particularly important for our mobile clients. A 256-bit ECC key is considered enough. Rivest Shamir Adleman (RSA) certificates are slower but compatible with a wider variety of older servers. RSA keys are larger, so a 2048-bit RSA key is considered enough for the security.

PROCEDURES

To obtain an HTTPS certificate, perform the following steps:

- Create a private and public key pair, and prepare a Certificate Signing Request (CSR), including information about our organization and the public key.
- Contact a certification authority and request an HTTPS certificate, based on the CSR.
- Obtain the signed HTTPS certificate and install it on our NGINX web server.

Conclusion:

It is very important for our company to use the HTTPS on our webserver and website as business is expanding and dealing with large customer database. At any point we never comprise on security and thus we aren’t using any outdated crypto which is considered less secure and redirecting traffic for all the customers from HTTP to HTTPS.