SSL and Certificates

SSL / TLS and HTTPS?

What is SSL / TLS and HTTPS?

You have heard of all these terminologies and what do they all mean? How do they work together to provide a more secure way of browsing your web on the internet?

In this article, I will explain all the basics that you will need to know about SSL / TLS and how that is used to build the secure HTTP protocol that we are using in our modern day.

HTTP - Time before secure internet

In the beginning, there is the plain old HTTP, the hypertext transfer protocol. RFC 1945 explains the hypertext transfer protocol and how it is used to share HTML documents between servers and clients. When HTTP is invented it was used purely for sharing documents between users. I.e. a research paper that can be shared in HTML document to other university around the world. Security was not a concern back then because there was no need to secure any of the content if it was meant to be shared freely around the world.

However, the surge in online banking and online shopping in the 1990s increase the concern of security. When form logins can be easily intercepted and the user name and password can be recorded down because with HTTP those information are in plaintext, thus a more secure way of browsing the web was needed.

HTTPS is created, it is HTTP wrapped in SSL / TLS, which essentially encrypt the data that's transferred between the client and server, making it difficult for anyone sniffing the TCP packets in the internet hard to decode what information is shared.

SSL / TLS Why two names?

Let's get the elephant out of the room first why is there two different name for the same thing?

SSL = Secure socket layer

TLS = Transport layer security

TLS is the direct successor to SSL, as all version of SSL is now all deprecated. They are both communication protocols that encrypts the data between servers and clients.

We still refer as SSL because it is commonly understood compared to TLS acronym.

To use SSL / TLS we will need to create what's called SSL certificate which verifies the server's identity and verifies the integrity if the encryption data.

HTTPS

Now HTTPS is just HTTP but with SSL / TLS layer on top of it. Between the server and client it will use HTTP to do the communication, but the transportation of the actual TCP packet is done through SSL connection.

To use HTTPS connection the server and the client establish the SSL connection first. To do that the server and client first do a SSL handshake to find which cipher they will want to use. Then certificate from the server is validated with the client. Finally key exchange is performed to do the actual encryption of the data.

To recap quickly:

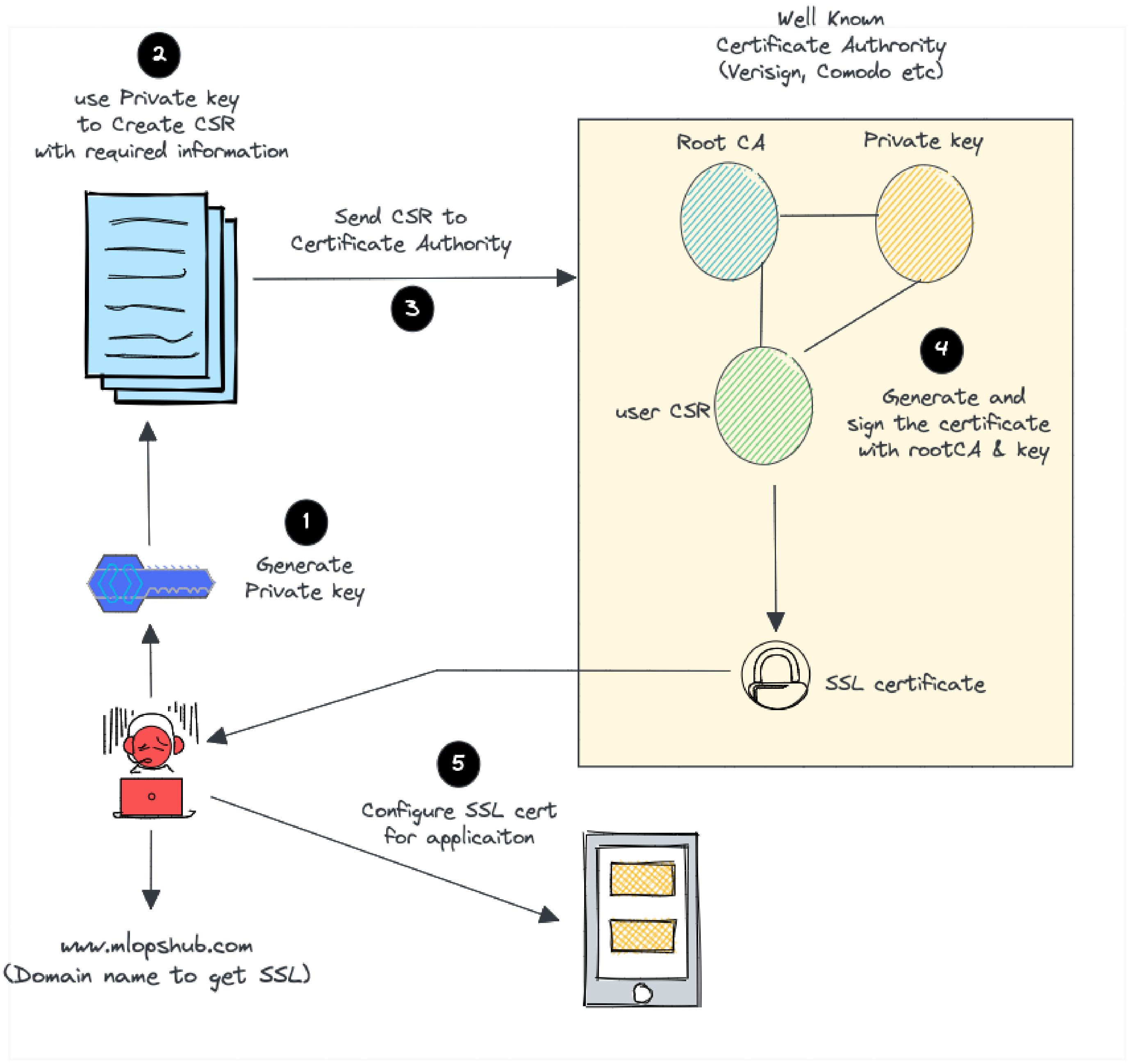
1. SSL Handshake, exchange the cipher suite information, agree on which cipher suite to use for the duration of the SSL connection
2. Certificate exchange, the server have to prove the identity to the client by sending over it's certificate. The client will then verify the certificate by going through the certificate chain of trust.
3. Key exchange, then key exchange is perform in order to use it to encrypt the data

In the next section we will go over how to generate a server certificate, sign it, and validate it.

SSL Certificate 101

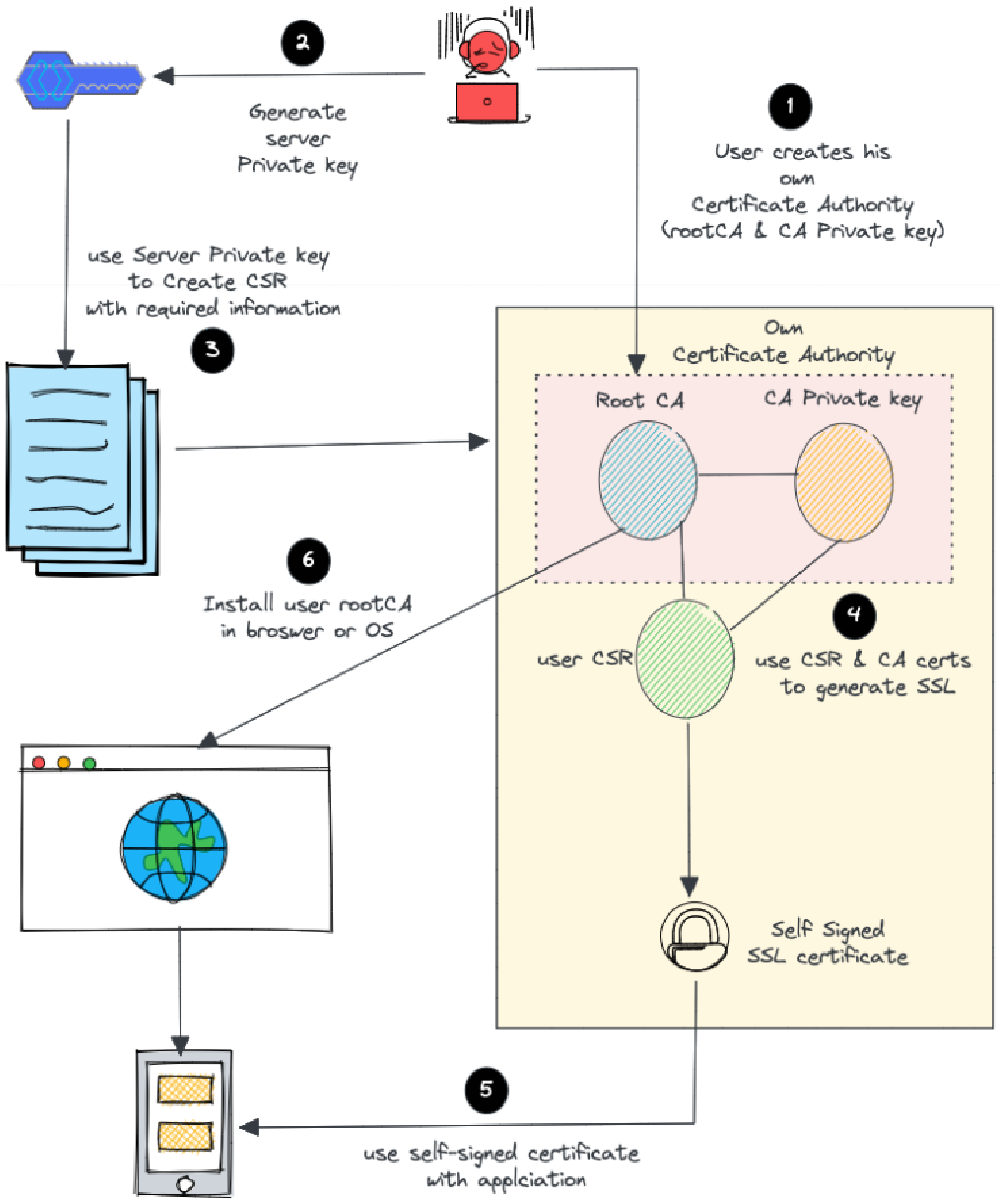
How to generate a self-signed certificate

If you wish to generate a certificate from an actual Certificate Authority then the flow is done like such:



I will only be explaining how to generate a self-signing certificate because the process is essentially the same, except that you abstract out the CA as a thirty party that will handle the request for you automatically.

To create create a self-signed certificate (a certificate that's signed by a CA that YOU own) this is the steps:



2
Generate server Private key
1
User creates his own Certificate Authority (rootCA & CA Private key)
use Server Private key to Create CSR with required information
3
Own Certificate Authority
Root CA
CA Private key
6
Install user rootCA in broswer or OS
user CSR
4
use CSR & CA certs to generate SSL
Self Signed SSL certificate
5
use self-signed certificate with applciation

1. Create your own root CA certificate and root CA private key

To get started with creating self-signed certificate you will need to first create your root CA private key and root CA certificate.

Running the following command will generate a private key and name the file as `server.key` to indicate that this is the private key for the server.

```
openssl genrsa -out root.key
```

To generate the root CA certificate run the following command:

```
openssl req -x509 -days 365 -key root.key -out root.cer
```

This will generate a root certificate with the server private key that's valid for 365 days. You can validate this certificate by running and you can see the expiration date of this certificate.

```
openssl x509 -in root.cer -text
```

> Note that you must provide a common name field in order to validate the certificate trust chain.

2. Create your server private key and certificate signing request

Go ahead and generate your server's private key with the following command.

```
openssl genrsa -out server.key
```

Now then you will need to create a certificate signing request with your server's private key.

```
openssl req -new -key server.key -out server.csr
```

> When prompted to enter in a common name field you will also need to enter it in otherwise, you cannot validate the certificate trust chain.

3. Use CSR to create server certificate

The following command will use the certificate signing request (who to sign the certificate for) and the root server's certificate information and the private key to create a server certificate signed by the root CA hosted by ourselves.

```
openssl x509 -req -in server.csr -CA root.cer -CAkey root.key -CAcreateserial -out server.crt -days 365
```

4. Validate certificate chain

Finally, to validate that the output server certificate is indeed created and signed by the root server we can run the following command:

```
openssl verify -CAfile root.cer server.crt
```

If everything is good, then you should see the output `server.crt: OK`

What is Self-Signed Certificate

All root CA certificates are self-signed in the sense that the root CA certificate is created using the certificate's own private key. Nothing special about them besides that the organization owns them are a trusted entity by the general public and have good reputation in owning and issuing out certificates.

You can also generate your own self-signed certificate.

What is PEM files

https://origin-blog.mediatemple.net/work-life/ssl-certificate-101-everything-you-need-to-know/

- What is PEM files
- What is X.509 file types
- How the hell does certificate work