Assignment-3

Aarunish Sinha

April 2021

1 README

Open 3 terminal windows/tabs.
In the first window run,

```
$ python3 my_ttp.py
```

In the second window run,

```
$ python3 my_server.py
```

In the third window run,

```
$ python3 my_client.py
```

2 Sample Run

```
(base) Aarunishs-MacBook-Pro:TLS-Prototype aarunishsinha$ python3 my_ttp.py
CA certificate has been found as CA/ca.crt
CA Certificate valid for 364 days
Connecting to server
Server Digital Certificate issued
Connecting to Client
Client Digital Certificate issued
```

Figure 1: Trusted Third Party

```
(base) Aarunishs-MacBook-Pro:TLS-Prototype aarunishsinha$ python3 my_server.py
Key pair generated
Connecting to the TTP
Certificate recieved
Requested Client Certificate
('ECDHE-RSA-AES128-SHA256', 'TLSv1.2', 128)
Certificate Verified
Connected to Client
```

Figure 2: Server

```
(base) Aarunishs-MacBook-Pro:TLS-Prototype aarunishsinha$ python3 my_client.py
Key pair generated
Connecting to the TTP
Certificate recieved
Requested Server Certificate
Certificate verified
Message Recieved: The OTP for transferring Rs 1,00,000 to your friend's account is 2
56345.
```

Figure 3: Client

3 Implementation

3.1 Issuing Certificates

The TTP first creates a RSA key pair and a self-signed `X509v3` certificate to establish the CA. The server and the client create their own RSA key pair and send a message to the TTP containing their common name. The TTP uses this common name and and the public key of the client/server to create a signed `X509v3` certificate.

I have used the `OpenSSL` library in python to create certificates. Since, this library only allows RSA or DSA(not ECDSA) key generation, I used RSA key pairs for all the three entities. For signing the certificates, I used `SHA256` for TTP's self-signed certificate and `SHA384` for the client and server certificates.

3.2 TLS Handshake

The server and client connect through a socket. I have used the `ssl` library for performing TLS Handshake Protocol. A `ssl.SSLContext` wrapper is created for making the socket used for communication secure and also takes care of handshake by default (since the `do_handshake_on_connect` parameter is set as `True` by default). Then socket is programmed to make certificates necessary for authentication and is own pre-defined session ID such one session cannot be reused. I have used the `TLSv1_2` protocol since the library does not support `TLSv1_3` yet. The cipher suite list includes `ECDHE-RSA-AES128-SHA256` and `ECDHE-RSA-AES256-SHA384`. The program randomly allots one of the two. `CHACHA20` was not used because all the cipher suites using `CHACHA20` followed the `TLSv1_3` protocol and `ECDSA` could not be used since the certificates are signed with `RSA` keys and hence `RSA` is needed for authentication.

On connecting both the server and client prompt the other for their respective `X509v3` certificates and verify the certificates. I have also checked the common-name on the certificated and the validity of the certificates as an added layer of authentication. `ECDHE` is used has the key exchange protocol and then a message is sent for key confirmation.

3.3 TLS Record

Another shared secret key is created which is a random string and shared through the secure socket. The client generates this secret key and sends it to the server. The server on receiving the key, creates a message authentication code using the `hmac` and `hashlib` library. I have used `SHA-1` as the hash function. The server sends this authenticated message to the client with verifies the MAC and then extracts the message and prints it on the terminal window.

4 Security

Since, signed `X509v3` certificates are used for authentication, the implementation is safe from Man-in-the-Middle attacks. The ssl sockets use a session ID by default which makes it impossible for an adversary to perform replay attacks. Downgrade attacks are not possible since I have enforced the use of `TLSv1_2`, hence cipher suites with lower TLS versions are not compatible with the ssl socket.

5 Computational Cost

The TTP uses RSA to create all the three certificates. AES is used as the encryption algorithm after the TLS handshake is performed between the Client and the Server. There are 4 operations performed using AES(symmetric) encryption/decryption, sharing of the secret key for the TLS record protocol and the actual sharing of the message.

6 Communication Cost

The communication with the TTP only involves sharing of the common-name of the client and the server. The major. communication occurs in the interaction between the client and the server involving a `client_hello` and a `server_hello`, sharing of certificates across the secure socket for authentication, deciding the cipher spec, key exchange protocol, key confirmation, sharing of the secret key for message authentication and finally the transmission of the original message.