Rahul Anand – 301319328, Andy Ng - 301221017 Dec 8, 2020

CMPT 471 Project: End-to-End Encrypted Messaging

Problem

The problem that we chose to solve for our project was to provide an encrypted chat application where clients can communicate with each other and send encrypted messages that cannot be read by another host on the network or the server itself.

We implemented the basics of the Signal Protocol to achieve the end-to-end encryption. The signal protocol is made up of 4 parts.

- Extended Triple Diffie-Hellman (X3DH) key agreement protocol
- Double Ratchet algorithm
- Pre-keys
- Curve25519, AES, and HMAC-SHA256 as cryptographic primitive

Contributions: Andy worked on tools.py, chatClientDuplex.py, and chatServerDuplex.py, the test cases and the test cases video, and helped edit the report. Rahul worked on chatClientDuplex.py, chatServerDuplex.py and wrote most of the report.

Implemented Features

All the above parts are implemented (except for the double ratchet algorithm) and were implemented with help from the Pycryptodome python library. This library was used for the creation of keys, the key derivation function, and for importing cryptographic primitives such as AES-128 to encrypt messages, and thus we implemented the Signal protocol using these tools.

Our implementation works as follows:

1. When the server is created, it will generate a random large prime number *p* (modulus) and *g* (base). *P* and *g* will be sent to clients who connect to the server.
2. When a client connects, the client will generate its own random number *a,* and will compute its private key as being g^{ab}. It will then calculate its public key $g^a mod p$, which is sent to the server.
3. The public keys for every client are used to calculate a shared secret key using Diffie-Hellman key exchange, this time using public key and private key in the formula. This key allows for encrypted connection between clients and is processed through a key derivation function (PBKDF2) for further protection.
4. When a client wants to send a message to another client, the message will be encrypted with AES-128.
5. When a client receives a message, it is decrypted using AES cipher and the unique userKey. This hides the message from the server but allows clients to see the message.

The server is always listening for connections from clients. Once a client wants to connect the above steps are performed. Since we are updating our shared secret keys every time a new client joins the server, we can support more than 2 clients. We used the Pycryptodome mainly for the following purposes:

- Using an AES cipher in our encryption (using functions such as encrypt_and_digest and decrypt)
- Calling a key derivation function (PBKDF2)
- Generating random and large prime numbers (number.getPrime)

Issues

One issue with our project currently is the inability to close the server gracefully.

Learnt Lessons

Prior to the project, we had assumed that encryption was a straightforward process, where a client will send some keys, calculate some keys, and encrypted messaging can be achieved. However, there is more that can be done to ensure the safety of our encrypted messages, such as how the Signal protocol utilizes the double ratchet algorithm. Although we did not have time to implement such an algorithm, we see why it is important to the protocol. Essentially, in the double ratchet algorithm, whenever we send a message, we encrypt the message with a new key, thus if an attacker manages to break one message being sent, the other messages are still protected as they are using a different key. Although it is very unlikely that an attacker is able decipher the key, the Signal protocol still takes this precaution.