Security considerations

User authentication

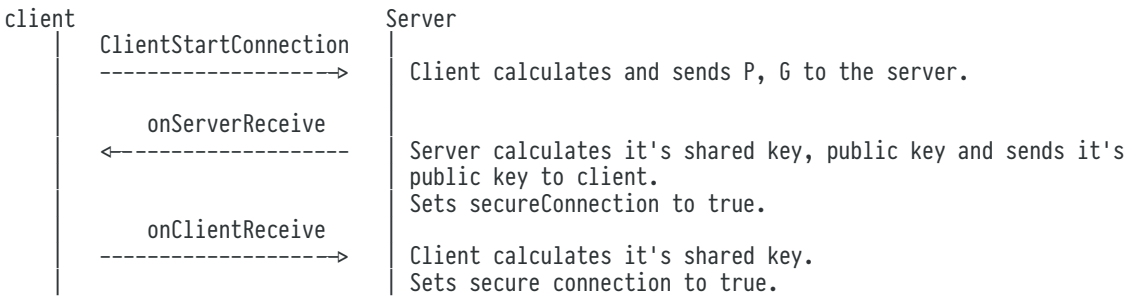
We support user authentication with username and password.

DoS protection

To prevent DoS attacks we make sure the send `Client-Hello-Handshake` message is always larger than the `Server-Hello-Handshake` message. We fill up the `Client-Hello-Handshake` message with random bits so it is always 1000 bits long.

Multi-way handshake

For the key-exchange we use the Diffie-Hellman key-exchange.



Diffie-Hellman algorithm relies on *discrete logarithm problem*. It is very hard for computers to solve discrete logarithm thus it is a good candidate against brute force attacks. This implementation works like this: - Client and server have to have the common values `P` and `G`. `P` is a prime number and `G` is the primitive root of that number. As connections are initiated by client, both `P` and `G` is calculated by client. Before sending `P` and `G`, client generates a private key and calculates it's public key. `P , G and client's public key` are sent to server by the client at the beginning of the connection.

- Server receives `P , G , and client's public key`. It selects a private key for itself and calculates it own public key. At this calculation, server also successfully calculates the `shared key` that the encryption will be based on.
- Client receives server's public key and calculates the `shared key`. At this point both server and the client have the same shared key which has never transmitted through insecure channels.
- Exposed API of Encrypt and Decrypt uses that shared key to encrypt and decrypt messages.

More info can be found at: Wiki Link

Encryption

We encrypt all messages with a asymmetric encryption (Implementation). Because we forgot to enable the encryption again, all messages get send unencrypted.

Man-in-the-Middle attacks

We are vulnerable against Man-in-the-Middle attacks. We do not use any server authentication mechanisms at the moment (e.g. certificates).