System design

# 1. Requirements

- Server is able to:

+ Sync file with client with encryption

+ Have a mechanism to prevent DDos

+ Detect new file stored

+ Have a file sending mechanism using queue like MQTT

+ Store encrypted file

+ Exchange new AES key everyday

+ Have simple GUI providing the functions:

+ Decrypt stored files to view

+ Delete decrypted files

+ restore previous working state after restarting

- Client is able to:

+ Sync file with server with encryption

+ Store encrypted file

+ Exchange new AES key everyday

# 2. Modules’ function in server

## 2.1. Transmission

- Send file

- Receive file

- Put file to queue if cannot send

- Send key

- Receive key

- Form packet following format

- Hash utility

## 2.2. ED

## 2.3. StorageModule

# 3. Modules’ function in server

# 4. Technology description

## 4.1. Message exchanging protocol

|  |  |  |
| --- | --- | --- |
| **From byte** | **To byte** | **Description** |
| 0 | 63 | Bytes to prevent DDos |
| 64 | 64 | Byte marking type of packet |
| 65 | 76 | Timestamp in form: YYYYMMDDHHmm |
| 77 | infinity | Encrypted data |

|  |  |  |
| --- | --- | --- |
| **Type of packet** | **Byte value** | **Description** |
| Exchanging data | 0 | Packet to exchange data |
| ExchangeACK | 1 |  |
| Update resource | 2 | Packet to update resources |
| UpdateACK | 3utils |  |

## 4.1. Exchanging new resources everyday

At specified time of the day, server and client exchange the following:

- new AES key for latter day

- new secret string for DDos prevention

- timestamp for exchanging resources the latter day

## 4.2. Mechanism to prevent DDos

In each packet exchanged, there are first 64 bytes reserved for preventing DDos. These 64 bytes are SHA 256 hash value (in hexadecimal form) of a secret string exchanged in Updating resources session.

The secret string is renewed every day. It occurs at Updating resources session.