

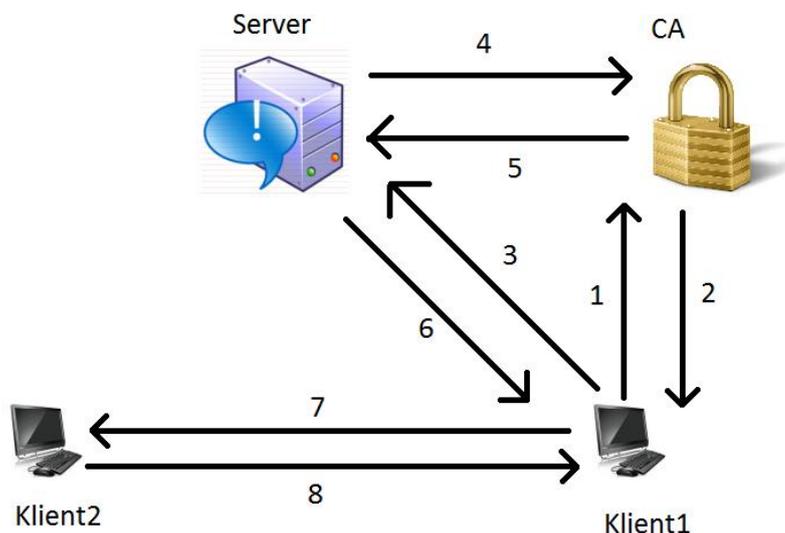
Projekt PB173

Postup při komunikaci:

Klient nejdříve vygeneruje sady klíčů, veřejny a soukromy pro Aes. Dále se napojí na CA a zaregistruje si u ni svůj veřejný klíč pod svým uživatelským jménem.

Klient dále pošle request-login na server zašifrovaný veřejným klíčem serveru, aby overil autoritu serveru. Nasledně obdrží od serveru zašifrovaný seznam online uživatelů. Toto musí dešifrovat svým soukromým klíčem. Vybere si toho správného a odešle mu communication-request zašifrovaný jeho veřejným klíčem.

Tento a předchozí krok zároveň slouží jako autorizace uživatele = nesprávný uživatel nedešifruje správně a jsou mu data kničemu. Pokud druhý klient potvrdí komunikaci, první klient vygeneruje a odešle symetrický klíč. Tím zahájí spojení. Nadále komunikují a komunikace je šifrována symetricky. Kdykoli bude klient potřebovat odešle abort-request který ukončí spojení mezi klienty a symetrický klíč se stává neplatným.



Naše aplikace se bude skládat ze tří částí, klientské, serverové a certifikační autority.

1. Klient vygeneruje svoji sadu veřejný/soukromí klíč. Své údaje a veřejný klíč pošle CA zašifrovaný jejím veřejným klíčem. Ta je přidá do svého seznamu.
2. Ta mu odešle potvrzení.
3. Klient odešle login request serveru, zašifrovaný jeho veřejným klíčem .
4. Server verifikuje request u certifikační autority.
5. Pokud je klient schválen obdrží přidá se do seznamu online uživatelů a následně je mu odeslán zašifrovaný jeho veřejným klíčem.
6. Klient si vybere ze seznamu s kým chce komunikovat a odešle mu communication request.
7. Klient2 obdrží communication request, verifikuje ho u CA, následně přijme nebo odmítne komunikaci.
8. Pokud přijme tak odešle communication response.
9. Dále pokračují v symetricky šifrované komunikaci, pomocí dohodnutého klíče.

Použité algoritmy: AES 128bitu, RSA 1024bitu, SHA-2

Server API:

```
/**
 * Encrypt/Decrypt data
 *
 * @param key rsa key
 * @param data data for encryption
 * @param start starting offset
 * @param length length of encrypted data
 */
void encrypt(unsigned char* key, unsigned char* data, int start, int length)
void decrypt(unsigned char* key, unsigned char* data, int start, int length)
```

```
/**
 * Generate 1024bit RSA keys
 *
 * @param public_key buffer for publickey
 * @param private_key buffer for private key
 */
void generate_keys(unsigned char* public_key, unsigned char* private_key)
```

```
/**
 * Contact CA and verify user
 *
 * @param username user name
 * @param key user publickey
 */
bool verify_user(char* username, unsigned char* key)
```

```
/**
 * Login user
 *
 * @param username user name
 * @param key user publickey
 * @param data string encrypted with user private key
 */
bool login(char* username, unsigned char* key, unsigned char* data)
```

```
/**
 * Logout current user
 */
void logout()
```

```
/**
 * Get online list
 */
char* get_online_list()
```

```
/**
 * Send communication request to online user
 *
 * @param username user name
 */
bool send_communication_request(char* username)
```

```
/**
 * Send data to online user
 *
 * @param username user name
 * @param data data for user
 */
bool send_data_to_user(char* username, unsigned char* data)
```

Client API:

```
/**
 * Generate both RSA 1028bit keys
 *
 * @param publickey
 * @param private key
 */
void generateRSAKeys(unsigned char* public_key, unsigned char* private_key);
```

```
/**
 * Generate AES 128bit key
 *
 * @param publickey
 * @param private key
 */
void generateAESKey(unsigned char* key);
```

```
/**
 * Encrypt data with Aes algorithm
 *
 * @param key
 * @param data for encryption
 * @param begin of block data
 * @param length of block data
```

```

*/
int encryptAES(unsigned char* key, unsigned char* data, int start, int offset);

/**
 * Decrypt data with Aes algorithm
 *
 * @param key
 * @param data for encryption
 * @param begin of block data
 * @param length of block data
 */
int decryptAES(unsigned char* key, unsigned char* data, int start, int offset);

/**
 * Encrypt data with Aes algorithm
 *
 * @param public key
 * @param data for encryption
 * @param begin of block data
 * @param length of block data
 */
int encryptRSA(unsigned char* key, unsigned char* data, int start, int offset);

/**
 * Encrypt data with Aes algorithm
 *
 * @param public key
 * @param data for encryption
 * @param begin of block data
 * @param length of block data
 */
int decryptRSA(unsigned char* key, unsigned char* data, int start, int offset);

/**
 * Send login request to server
 *
 * @param user name
 */
void login(const char* username);

/**
 * Send logout request to server
 */
void logout();

/**
 * Send Certificate to CA
 *

```

```
* @param publickey
* @param user name
* @param email
*/
void setCertificate(unsigned char* key, const char* username, const char* email);
```

```
/**
* Get Certificate of selected user from CA
*
* @param user name
*/
void getCertificate(const char* username);
```

```
/**
* Send request to client to start connection
*
* @param Ip address of client
*/
void requestConnection(string ip);
```

```
/**
* Send encrypted AES key to client and establish Connection
*
* @param Ip address of client
* @param publickey
*/
void establishConnection(string ip, unsigned char* key);
```

```
/**
* Abort connection with clients
*
* @param Ip address of client
*/
void closeConnection(string ip);
```

```
/**
* Encrypt data with Aes algorithm
*
* @param Ip address
* @param Block of data
*/
void sendData(string ip, unsigned char* data);
```

CA API:

```
/**
* add new user into system
```

```
*
* @param name
* @param email
*/
void registerNewUser(unsigned char* name, unsigned char* email, unsigned char* publicKey);

/**
* verify if user whose request is send have access to private key of specific user
*
* @param request
*/
bool verifyUser(unsigned char* request);

metoda ověří jestli daný uživatel skutečně vlastní soukromý klíč, toho za koho se vydává.
request bude zašifrovaný soukromým klíčem uživatele, jehož identitu ověřujeme a bude tvaru:
fixníPočetRandomBitů-Jmeno-RegistrovanáEmailAdresa-FixníPočetRandomBitů

/**
* send public key of specific user
*
* @param name
*/
void getPublicKey(unsigned char* name);
```