Computer Networks

Programming assignments #3

資管三 B10705002 吳亞宸

如何編譯、執行 client 端程式

解壓縮以後，在資料夾底下輸入 `make`，就可以產生 `server` 和 `client` 這兩個可直接執行的檔案。若要直接輸入編譯，則編譯的方式為 `g++ -o client client.cpp -lssl -lcrypto -pthread` 和 `g++ -o server server.cpp -lssl -lcrypto -pthread`。執行 `server` 程式的指令為 `./server <server port>`，而執行 `client` 程式的指令為 `./client <server ip> <server port>`。

程式執行環境

程式執行環境為 Windows 子系統 Linux 版（WSL2），版本為 5.10.102.1。使用的 openssl 版本為 OpenSSL 1.1.1f 31 Mar 2020。

安全傳輸的實作與方法

在 server 端的 main function 開始時，先透過 `RSA *generateRSAKey()` 去生成一個 public private key pair。Client 端也會在開始執行時，透過此 function 去建立 client 端的 public private key pair。

```
RSA *generateRSAKey()
{
    RSA *rsa_key = RSA_new();
    BIGNUM *e = BN_new();
    BN_set_word(e, RSA_F4);

    if (RSA_generate_key_ex(rsa_key, 2048, e, NULL) != 1)
    {
        std::cerr << "Error generating RSA key pair.\n";
        RSA_free(rsa_key);
        BN_free(e);
        return nullptr;
    }
```

```
    BN_free(e);
    return rsa_key;
}
```

透過 `X509 *generateSelfSignedCertificate(RSA *rsa_key)` 這個 function 去建立一個 self-signed certificate。

```
X509 *generateSelfSignedCertificate(RSA *rsa_key)
{
    X509 *cert = X509_new();

    if (cert == nullptr)
    {
        std::cerr << "Error creating X.509 certificate.\n";
        return nullptr;
    }

    EVP_PKEY *pkey = EVP_PKEY_new();
    EVP_PKEY_assign_RSA(pkey, rsa_key);

    X509_set_version(cert, 2);
    ASN1_INTEGER_set(X509_get_serialNumber(cert), 1);
    X509_gmtime_adj(X509_get_notBefore(cert), 0);
    X509_gmtime_adj(X509_get_notAfter(cert), 31536000L); // 1 year validity
    X509_set_pubkey(cert, pkey);
    X509_sign(cert, pkey, EVP_sha256());

    EVP_PKEY_free(pkey);
    return cert;
}
```

接著 client 和 server 用 `ssl_read` 和 `ssl_write` 傳輸訊息。在登入以後，server 回傳 list 會同時將 public key 用 string 的方式傳給 client。轉成 string 和轉回 RSA 的 function 如下

```
std::string getPublicKeyAsString(RSA *publicKey)
{
    BIO *bio = BIO_new(BIO_s_mem());
    PEM_write_bio_RSA_PUBKEY(bio, publicKey);

    char *ptr;
    size_t len = BIO_get_mem_data(bio, &ptr);
```

```
    std::string result(ptr, len);

    BIO_free(bio);

    // std::cout << "Result\n"<< result << "\nResult";
    return result;
}
RSA *getPublicKeyFromString(const std::string &keyString)
{
    BIO *bio = BIO_new_mem_buf(keyString.c_str(), -1);
    RSA *publicKey = PEM_read_bio_RSA_PUBKEY(bio, nullptr, nullptr, nullptr);
    BIO_free(bio);

    return publicKey;
}
```

在登入以後，client 端傳給 server 的任何訊息都會先透過 server 的 public key 加密。而 p2p 的過程中，p2p 中的 client 會先用明文傳送 "Key" 給另一方，對方會回傳自己 public key 的 string。收到 peer 的公鑰以後，即可將交易的訊息加密傳給對方。對方收到以後再透過 server 端的公鑰加密後，用 `ssl_write` 將交易資訊傳送給 server。

參考資料

https://github.com/davidleitw/socket

https://shengyu7697.github.io/cpp-linux-tcp-socket/

https://stackoverflow.com/questions/59096604/difference-between-rsa-sign-and-evp-digestsignx

https://codereview.stackexchange.com/questions/205478/a-simple-c-client-that-sends-data-over-tls-using-openssl