SENG2250 System and Network Security
School of Information and Physical Sciences
Semester 2, 2022

Assignment 3 (100 marks, 25%) - Due: 30 October, 23:59

Aims

This assignment aims to establish a basic familiarity with network security topics via analysing, designing, and implementing solutions.

Questions

Part1: X.509 Hierarchy (36 marks)

Refer to the course Cavans Assignment 3 group for the quiz-style task "Assignment 3 – Part 1".

Part2: Programming Task (64 marks)

A client and a server are planning to do data exchange. They decide to use a simplified SSL handshake (see Figure 1) to establish a secure channel (session key) then exchange data. The simplified SSL handshake removes the messages for alert, change cipher spec, certificate, etc.

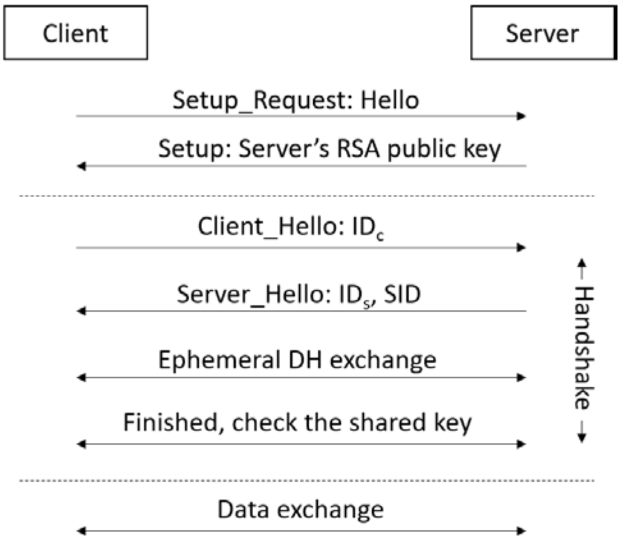

Figure 1. Secure data exchange.

ID_C: client ID; ID_S: server ID; SID: session ID;

Your task: implement the above mechanism in Java (alternatively C++/Python). The following components are mandatory for implementation.

- Fast modular exponentiation (**8 marks**)
- RSA signature scheme. (**12 marks**)
  - RSA key generation: randomly generate two primes p, q (for 2048-bit RSA). Set the public key as the fixed $e = 65537$. Server's RSA public key will be sent to the client in the Steup message. Assume this message can be securely delivered, no security protection is needed. Note that a client DOES NOT have its RSA keys.
  - RSA signature generation: using SHA256 for message digest computation.
  - RSA signature verification: using SHA256 for message digest computation.
  - The underlying hash function is SHA256. You can use it from the Java library.
  - Key generation needs to be implemented using (Java) BigInteger.
  - RSA signature generation and verification need to be implemented using your own fast modular exponentiation method.
- Diffie-Hellman key exchange (**8 marks**)
  - Use the parameters p, g from the System Parameters section.
- The DH key exchange should be secure against man-in-the-middle attacks. (**8 marks**)
- HMAC (**8 marks**)
  - Use SHA256 as the underlying hash function.
  - Use the DH key (e.g., $k = g^{xy}$) to generate the authentication key k', such that $k' = H(k)$, where $H()$ is the SHA256 hash function.
  - HMAC is calculated as (refer to lecture 2)
$$H(k', m) = H((k' \oplus opad)||H((k' \oplus ipad)||m))$$
- CBC mode (**12 marks**)
  - Assume a message is always a multiple of 16-byte, i.e. no padding needed.
- Data exchange (**8 marks**)
  - When a shared session key is created, they use 192-bit AES encryption with **CBC** and **HMAC** to protect data confidentiality and integrity, respectively.
  - Demonstrate at least **two** message exchanges, where each message is exactly 64 bytes.

You may refer to the FAQ for more information.

Compilation

- Please provide a readme.txt file for compilation and execution instructions.
- Uncompilable or unexecutable program may receive zero marks.

Input/Output

- Print (to standard input/output) all messages exchanged between the client and server.
- Use a proper output format to demonstrate the message exchange.

System Parameters

Hash function: you should use SHA256 whenever a hash function is needed.

Diffie-Hellman Key Exchange parameters (p, g)

p =
1780119054785422665282375624501599901452321563691206742732744503144428657887370207706126952521234630795671567847784664499706507709207278570500096683881440341297452211718185060472311500393010799593580673953487170663198022620197149665241350609459137075949565146728556906067941358375427073717274295513433206952 39

g =
1740682075324020951858119801235234365386044907945613509784958310405999534884558231478515974089409507253077970949157594923683005742524387610370844734671801488761181030830437549851909834726015504946913294880833954923138500003616464826446084923040787218189599990564960977693680177492737089620066891879567442107 30

Notes

- Your implementation MUST be able to handle large numbers. Otherwise, **<u>12 marks</u>** will be deducted.
    - Java
      https://docs.oracle.com/javase/7/docs/api/java/math/BigInteger.html
    - C++ users should use NTL library.
      https://www.shoup.net/ntl/doc/tour-examples.html
- Your implementation MUST use socket programming. Otherwise, **12 marks** will be deducted.
    - Java tutorial

      https://docs.oracle.com/javase/tutorial/networking/sockets/
    - C manual (This can be used with C++ with a few modifications)

      http://man7.org/linux/man-pages/man2/socket.2.html
    - C++ tutorial (uses boost, you would want build tool to manage that, such as https://cmake.org/)

      https://theboostcpplibraries.com/boost.asio-network-programming
    - Python example and documentation

      https://docs.python.org/3/library/socket.html#example

FAQ

1. What is about the “Setup_Request: Hello” message?
   It is just the text “Hello” that initiates the setup phase.
2. Can I use modpow() (or some function like that from the library) for the fast modular exponentiation computation?
   No. You need to implement the function based on the pseudocode in Lab 2.
3. What is an identity like IDs?
   It is a random character/number string of your choice, e.g., IDs=1234abcd.
4. Which is the shared session key for (CBC-AES192) encryption and HMAC?
   It is k'.
5. Can I use the “CBC” encryption mode from the library?
   No. You need to implement CBC encryption and decryption processes.
6. What should I send for the data exchange demonstration?
   Anything, as long as exact 64 bytes of each message.
7. Can I use the external cryptography library?
   Yes, but you have to implement the required components.
8. Can I reuse the code from the labs?
   Yes, you can.

Submission

All assignments must be submitted via Canvas. If you submit more than once, then only the latest will be graded. Your submission should be one ZIP file containing:

- All source code files
- A readme.txt for compilation/execution instructions.
- Submit your answers to **Part 1** separately on Canvas.

The mark for an assessment item submitted after the designated time on the due date, without an approved extension of time, will be reduced by 10% of the possible maximum mark for that assessment item for each day or part day that the assessment item is late. Note: this applies equally to week and weekend days.

Plagiarism

A plagiarised assignment will receive ZERO marks (and be penalised according to the university rules).