THE UNIVERSITY OF NEWCASTLE AUSTRALIA

SCHOOL OF INFORMATION AND PHYSICAL SCIENCES

SENG2250 System and Network Security

Assignment 2

This assignment is to be done individually.

Due on ***Sunday,*** **17 September 23*****, 11:59pm****, electronically via the "Assignment 2" submission link in Canvas.*

Total 100 marks

Aims

The aim of this assignment is to create a secure communication channel using SSL certificates. In the second part, you will develop a basic password manager, enhancing its security through the RSA algorithm.

Guidelines

You are free to use any programming language for the tasks listed below. Additionally, you may use Large Language AI models (e.g., ChatGPT), although if you do so, please refer to the "Using Generative AI.pdf" file found on Canvas, this document provides advice to better use the toolset and formatting advice for answering questions.

Task 1: Securing a Communication Channel with SSL (45 Marks)

Summary: For this task you will set up SSL certificates with either XCA or OpenSSL, and modify the provided programs to use those certificates.

Problem statement: Jill and Chris have designed a socket-based client server program to covertly communicate with each other but keep finding that their messages are being leaked. After chatting about this issue with their friend Barry, they learnt of the security technology of SSL and how it may help with maintaining the confidentiality of their communications. So, they have employed you, an up-and-coming cyber security expert, to add SSL functionality to their program, and to set-up the SSL certificates required.

Getting started: On the canvas page you will find the starting code for this task, you have the option of either modifying the Java or Python version. The instructions for the set of programs are included in the README.md file in the folder of the respective version. You will need to install either the XCA (https://hohnstaedt.de/xca/) or OpenSSL (https://www.openssl.org/) programs to create SSL certificates. Both sets of programs include a server that belongs to Jill, and a client that belongs to Chris, there is also an interceptor/sniffer program that is the leak. When creating certificates, have Barry as the CA, and the issuer name match who it belongs to, the other data in the certificates can be anything you want. If you set up the SSL and its usage correctly, the sniffer/interceptor will be unable to recover the messages being sent between Chris and Jill.

Requirements:

- The SSL certificates are correctly set up, where Barry is the CA and Jill owns the server. **(15 Marks)**
- Client and server programs use SSL in all socket communications. **(20 Marks)**
- The interceptor/sniffer program is unable to read communications between Chris and Jill (achieved if the above two points are correct).

Reflective questions:

1. What other potential security issues are there with this program? Identify and discuss at least three. **(6 Marks)**
2. If Barry were to secretly be the leaker, would the program remain secure? Explain why or why not. **(4 Marks)**

Task 2: A Single Client Password Manager (45 Marks)

For this task, you will develop a basic password manager for a single client. This password manager will be secured with your own implementation of the RSA algorithm. You must find the correct way to use RSA, i.e., which parts of the messages to encrypt/decrypt, so that the password management server and potential interceptors are never able to see the cleartext passwords. We have provided the starting files for Java or Python based programs on the Canvas page.

The following sections provide further details for the expected pieces of the program.

RSA

You are to implement and use your own version of the RSA algorithm. On Canvas we have included a primes.txt file, which you will read p and q from to compute n, ϕ(n), e, and d. To find d, you may use the standard library modular inverse function, i.e., BigInteger.modInverse(), or pow (x, -1, n). For all other places where modular exponentiation is required, you will use your own implementation of the fast modular exponentiation algorithm.

Password Manager Client

You will extend the included client.py or Client.java file to have an interface and to correctly handle passwords. The interface should allow the user to store, retrieve, or end the program in any order at any point. The interface should gracefully handle erroneous inputs and received messages.

The client has three possible network messages that it will send to the server:
- store <website> <password>
- get <website>
- end

Where store tells the server to store the <password> for the <website>, get tells the server to send back the password for the specified <website>, and end tells the server to end the program.

The following is an example output and input for the client:

```
Welcome to the SENG2250 password manager client, you have the following options:
- store <website> <password>
- get <website>
- end

>>> store example.com correct horse battery staple
Password successfully stored.

You have the following options:
- store <website> <password>
- get <website>
- end

>>> get example.com
Your password for example.com is correct horse battery staple

You have the following options:
- store <website> <password>
- get <website>
- end

>>> end
bye.
```

Password Manager Server

You will extend the server.py or Server.java file to handle the store and get commands received from the client.

When the server receives a ‘store’ command, it will store the specified password such that it is mapped to the specified website. Afterwards it will send back the message: "Password successfully stored."

When the server receives a ‘get’ command, it will send back only the password that is mapped to the specified website.

The ‘end’ command is already handled in the code you are extending, but its specifications are that on receiving an ‘end’ command it will send back "end okay" and quit the program.

Requirements: The following are the requirements for the entirety of Task 2.

- Personal implementation of RSA, using only BigInteger in Java or the standard library in Python. Computes n, ϕ(n), e, and d, in addition to encryption and decryption. Uses a personal implementation of the fast modular exponentiation algorithm where applicable. **(10 Marks)**

- Extends the client program to have an interface that handles erroneous inputs by clearly telling the user what was wrong and taking them back to the main interface. **(5 Marks)**

- Correctly handles passwords so that interceptors and the server are never able to clearly read them. **(20 Marks)**

- Extends the server to handle the store and get commands. **(4 Marks)**

- All sent messages use the sockets and follow the exact correct format*:* **(6 Marks)**

Client: store <website> <password>	Server: Password successfully stored
Client: get <website>	Server: <password>
Client: end	Server: end okay

Task 3: Reflections **(10 Marks)**

For this task you will write two brief word reflections on Tasks 1 and 2. The two reflections will be 200-500 words each and will firstly discuss what you have learnt from extra resources, such as websites, textbooks, or large language models, to complete the tasks. The reflections will then relate what you have learnt to subjects covered in our labs and lectures. To strengthen your reflection, you may also either discuss how these tasks relate to real world applications, or how your programs resulting from these tasks may be extended to be better applicable in the real world.

Your reflection should cite the extra resources you learnt from, if any, including prompts from LLMs (such as ChatGPT). Your bibliography should follow a citation standard such as IEEE, APA, or Harvard. The bibliography will not count towards your word count.

Your answers to the two reflective questions in Task 1 should also be included in this document.

Submission Guidelines

Please submit your work in a single zip file. This file should encompass a PDF document illustrating program execution with running examples (including screenshots to validate the successful implementation of requirements) and reflections. Additionally, include two folders named "Task 1" and "Task 2". The two folders will contain all of your code and external files required to run your programs for the respective tasks, they also will contain a "README.md" file which states instructions for setting up and running your programs.

Marking Rubric

Task	Requirement	Basic	Sound	Good	Excellent
1	The SSL certificate set up	(1) An SSL certificate is provided but does not follow the correct setup	(5) SSL certificate provided and has correct information but not the correct setup	(10) Multiple SSL certificates are provided but some information or the setup is wrong	(15) Multiple SSL certificates are provided with correct information and setup
	Client and server programs SSL usage	(5) SSL sockets usage was attempted but fails to run	(10) SSL sockets start but are not used for communications or do not use the generated certificates	(15) SSL sockets correctly uses the certificates but is only used for some communications	(20) SSL sockets correctly uses the certificates and is used for all communications
	Reflective question 1	(1) Identifies one possible issue	(2) Identifies and describes one issue	(4) Identifies and describes two issues	(6) Identifies and describes three issues
	Reflective question 2	(1) States whether the program is secure	(2) Correctly states whether the program is secure	(3) Correct statement with a simple one sentence explanation	(4) Correct statement with a detailed explanation
2	RSA	(1) Implements one of the components, but makes a mistake	(4) Correctly implements two of the components	(8) Correctly implements four of the components	(10) Correctly implements all components
	Client program errors handling	(1) Only prints that there was an error but	(2) Handles only client-side errors	(4) Handles both client and server-side	(5) Gracefully handles all errors

		does not handle them		errors, but not always gracefully	
	Password handling	(5) Some password confidentiality is considered but is not effective	(10) Passwords are confidential to an interceptor or the server, but cannot be clearly retrieved by the client	(15) Passwords are confidential to only one of: an interceptor or the server	(20) Passwords are confidential to both an interceptor and the server
	Server command handling	(1) The server responds only one of the get or store commands but not correctly	(2) The server responds to both get and store commands but not correctly	(3) The server responds correctly to one of the get or store commands and the other incorrectly	(4) The server responds correctly to both get and store commands
	Message formatting	(1) For one of the sets of messages only either the client or the server follows the correct format	(2) For one of the sets of messages both the client and the server follow the correct format	(4) For two of the sets of messages both the client and the server follow the correct format	(6) For all the sets of messages both the client and the server follow the correct format
3	Reflections	(2) What is learnt from external resources is only listed	(5) What is learnt from external resources is discussed	(8) What is learnt from external resources is discussed, tasks are related back to the lecture and lab contents	(10) External resource lessons are discussed, tasks are related back to this course, and discussions of how the tasks relate to or may be extended to the real work are provided

**Note: Marks may fall in between these cases for submissions of intermediate qualities.