University of Wollongong
Singapore Institute of Management

School of Computing and Information Technology (SCIT)

Lecturers: Distinguished Professor Willy Susilo and Mr. Japit Sionggo

CSCI361 - Session 1 2025

Assignment 2 (15 marks)

Due: Sunday, 23 February 2025, 09.00pm Singapore time. Submission via Moodle only.

Aim: To gain a basic familiarity with public key cryptography and its applications.

Standard Requirements for Assignments

- Submission must be made via Moodle. No other submission method will attract any marks.
- Submission via email will result in getting ZERO.
- Follow the directions given by the tutor (Mr. Sionggo).
- At the top of your code, you will need to specify the version of the programming language that you use.
- Students are to give batch / make files for compilation.
- Students are to place all compilation and instructions on how to run the program inside a `README.TXT` file. The markers will refer to this file when marking.
- Submission filenames are to be the same as the ones given in the assignment specifications, do not use your own filenames.
- **Plaigarised assignments will receive 0 marks immediately.**
- **DO NOT** post this assignment to any forum, or else you will receive 0 marks immediately.
- Penalty for the late assignment is 25% per day.

1 Task One. Least Common Multiple (2 marks)

Least Common Multiple (LCM) of two integers is the smallest integer that is a multiple of both numbers. If we know the GCD of the two integers, then LCM can be computed as follows:

$$LCM(a, b) = \frac{a \times b}{GCD(a, b)}$$

Write a program that given a and b as input, it outputs the LCM and GCD between those two numbers.

2 Task Two. Implementation of Extended GCD Algorithm (2 marks)

In this section, you are to implement the Extended GCD algorithm. When the program is run, it will ask for two variables a and b, and the task is to find α and β where $\alpha a + \beta b = gcd(a, b)$. You will need to output each row of the Extended GCD algorithm as shown in the lecture notes.

Here is an example of the screenshot:

```
$gcd
Enter a = 39
Enter b = 11
We are to find gcd(39, 11) using the Extended Eucledian algorithm.
The contents of the variables are as follows:
 n1 n2 r   q a1 b1 a2 b2
 39 11 6 3 1 0 0 1
 11 6 5 1 0 1 1 -3
 6 5 1 1 1 -3 -1 4
 5 1 0 5 -1 4 2 -7

Summary:
 gcd(39, 11) = 1
 39 * 2 + 11 * -7  = 1
```

3 Task Three. Implementation of Trapdoor Knapsack Encryption Scheme (2 marks)

In this section, you are to implement a trapdoor knapsack encryption scheme. When the program is run, it first asks for the size of the super-increasing knapsack, then the user is asked to enter the value of each a_i in the private key. Then, the user is asked to enter a modulus, follows by a multiplier. You will need to check whether condition of the multipllier is satisfied. Then, the public key will be generated and shown. Now, a set of message is being asked, and the ciphertext will need to be

displayed. Finally, a ciphertext will need to be asked and the correct decryption of the ciphertext will need to be displayed. Implement this part as knapsack.cpp/knapsack.java/knapsack.py.

4 Task Four. Collision Finding of Hash Functions (2 marks)

SHA-1 is a commonly used hash function. It produces 160-bit hash value. You can download the source code for SHA-1 from the web, but you need to state where you get the original code from. As an example, see this github: https://github.com/clibs/sha1

We learnt in the lecture that a good hash function should be collision-resistant, meaning that it is difficult to find two messages, m and m', where $m \neq m'$, such that $H(m) = H(m')$. In this task, we assume a simplified version of SHA-1, named SSHA-1, is used for hashing. SSHA-1 only **outputs 34 bits, which are composed from the first 10 bits and the last 24 bits of SHA-1** when hashing a message.

Your task is to find a pair of integers (x, x') such that $x \neq x'$ but the SSHA-1 hash values of the following two messages are the same.

Mario owes [FIRSTNAME] x dollars

Mario owes [FIRSTNAME] x' dollars

You should replace [FIRSTNAME] with your first name. Write a C++, JAVA or Python program to accomplish the task. Your program should output the two messages, their hash values (should be the same), and the number of trials your program has made before it finds the collision.

5 Task Five. Implementing Ring Signature of 2 users (2 marks)

In this task, you are to implement a ring signature for 2 users, as described in the lecture notes. The input files are the following:

- publickey.txt
- message.txt

The file publickey.txt has four lines, which indicates: e_1, n_1, e_2, n_2 from RSA algorithm. The message.txt contains a string of characters, which needs to be signed. You need to implement two programs: sign and verify. The sign program will sign the message (from message.txt) and read the public keys from publickey.txt. It will ask for one input, which is user 1 or user 2, who is the signer, and the program will ask for that user's private key. Then, the sign program will output `signature.txt`.

The verify program will take an input of publickey.txt, message.txt and signature.txt and it will output `True` or `False` to show the verification of the ring signature.

The symmetric encryption should use the AES algorithm. You can import the AES algorithm from the existing library or use any implementation of AES algorithm (with 10 rounds) to do this.

6 Task Six. Diffie-Hellman Key Exchange and Man-in-the middle attack implementation (3 marks)

You are to implement a Diffie-Hellman key exchange algorithm here. The input should receive the public parameter. Then, generate a secret key for each user randomly, and then show the common key (of course, you will need to generate the generator, etc.).

In the second step, implement the man-in-the-middle-attack, and finally, implement one solution that solves this problem. You need to demonstrate that a man in the middle attack will no longer work after you patch the original algorithm.
Implement the protocol using C++/Java/Python.

7 Task Seven: Analysis of RSA algorithm (2 marks)

Alice decides to use RSA with the public key N. In order to guard against transmission errors, Alice has Bob encrypt his message twice, once using the encryption exponent e_1 and once using the encryption exponent e_2 (where $e_1 \neq e_2$). Eve intercepts two encrypted messages c_1 and c_2. Assuming Eve knows N (since it is public) and the two encryption exponents e_1 and e_2, then:

- Can Eve compute the plaintext without knowing the private key of Alice? Explain your answer using mathematical explanation. (2 marks)
- If Eve can find the plaintext, what is the issue that has occured here? If Eve cannot find the plaintext, then is there any other information that needs to be given to Eve so Eve can find the plaintext? Explain your answer using mathematics formula. (2 marks)

Submission

You need to submit one ZIP file and upload it to Moodle. In this ZIP file, you need to create six subdirectories, in which each subdirectory will have the answer to each task. For each programming task, write a README file that explains the compiler setting. Ideally, you should make a Makefile for each of the tasks.

DISCLAIMER: This assignment contains an intellectual property that is owned by the University of Wollongong. Please do not use it without the permission from the University of Wollongong. If you have any questions, please contact the author on `wsusilo@uow.edu.au`.

DISCLAIMER: By submitting your assignment, you **DECLARE** that the assignment is your own work and you did not obtain it from any third party or even purchase it from someone else or ask someone else to do it for you. If you violate this rule, then you may end up failing the subject entirely.

Notes about Class Test: Class Test is scheduled on 15 February 2025 (Saturday) for full time students, and 18 February 2025 (Tuesday) for part time students, during the respective tutorial sessions.