University of Wollongong
Singapore Institute of Management
School of Computing and Information Technology (SCIT)
Lecturers: Professor Willy Susilo and Mr. Japit Sionggo

CSCI361- Session 3 2022
Cryptography and Secure Applications

Assignment 2 (15 marks)

Due on 28 August 2022, 10pm Singapore time. Submission via Moodle only.

The aims of this assignment are to understand some public key cryptosystems and their mathematical components, and secret sharing scheme.

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.
- As usual, for all programming tasks, you need to take the screen capture of the sample run of your program and put it in a pdf file.
- Submission filenames are to be the same as the ones given in the assignment specifications; do not use your own filenames.
- Plagiarized assignments will receive 0 marks immediately.
- DO NOT post this assignment to any forum, or else you will receive 0 marks immediately.
- Java Version - JDK 6 update 17 or above (Using Windows)
- Penalty for the late assignment is 25% per day.

1 Task 1. Cryptanalysis (2 marks)

Let Enc be an encryption algorithm based on a one-way function F, and r a shared secret key between the sender and the receiver. Enc works as follows:

1. Compute $K = F(r)||F(2r)||F(3r)||F(4r)||..;$
2. For a message M, the ciphertext is computed as $M \oplus K$ (i.e., one-time pad).

Assume that an eavesdropper knows the function F but does not know the secret key r.

a) Suppose F is defined as $F(x) = g^x \ mod\, p$ where p is a 1024-bit prime number and g is a generator of Z_p^*. Assume that the discrete logarithm problem cannot be solved in the group Z_p^*. The secret key r is chosen randomly from Z_{p-1}. Show that the eavesdropper can decrypt the whole message easily once after obtaining $g, p,$ and the first 1024 bits of plaintext-ciphertext pair.

b) Suppose F is an RSA function, that is $F(x) = x^e \ mod\, n$ where n is 1024-bit long. Assume the RSA problem cannot be solved in the group Z_n^*, and r is chosen randomly from Z_n^*. Show that the eavesdropper can decrypt the message easily once after obtaining n, e, and the first 1024 bits of plaintext-ciphertext pair.

Write your answer in a file called Task1.pdf. You need to show all the key steps in order to obtain full marks. Submit Task1.pdf together with the other tasks in this assignment to Moodle.

2 Task 2. Shamir Secret Sharing (3 marks)

In this section, you are to design and implement (t, n) Shamir's secret sharing scheme as described in the lecture in C++, Java or Python. Your program should be called SSS.cpp, SSS.java, or SSS.py and it comprises at least the following functionalities:

- Share Generation: to generate shares in the (t, n) Shamir's secret sharing scheme. As a hint, you should at least accept the secret, t, n and the modulus in the parameter.
- Share Reconstruction: to reconstruct the secret from the given shares in the (t, n) Shamir's secret sharing scheme.

In this part, you need to explain your design first in file Task2.pdf to explain the logic of your functions. As usual, you need to take the screen capture of the sample run of your program and put it in Task2.pdf as well.

3 Task 3. ElGamal Signature with SHA-1 (3 marks)

In this section, you are to implement an ElGamal signature scheme, where the message will need to be first hashed with the SHA-1 algorithm. The algorithm for SHA-1 is available online, for example at:

https://github.com/vog/sha1

In your report for this section, you will need to quote any algorithm where you download online and cite it accordingly, instead of stating that it is written by you.

The program is divided into three parts:

- Keygen: this part is to generate the private and public key for ElGamal algorithm and store it in a file called $keyfile.txt$.
- Sign: this part is to sign a text file given as an input. In this mode, the program will ask for the name of the text file, then read the private key from $keyfile.txt$ for signing. Prior to signing the file, the program will compute the hash of the file using SHA-1, then sign it to produce $sig.txt$.
- Verify: this part is used to verify the signature in $sig.txt$ for the file, using the public key from $keyfile.txt$.

You need to structure your program using the possible inputs given as the parameter of the program. This part needs to be written in Java, C++, or Python. This part will need to be submitted with the filename ElGamalsign (with .cpp, or java, or py respectively), together with Task3.pdf containing your report detailing how to use the program.

4 Task 4. Design of One-Time Signature Schemes (3 marks)

A fail-stop signature scheme provides some extra protection to the system. An unbounded adversary is an adversary who can solve a computationally hard problem, such as discrete logarithm problem and factorization problem.

Please refer to the following scheme:
Key Generation
Public Key:

$$y_1 = g^{a1} y^{a2} \ (mod\ p)$$
$$y_2 = g^{b1} y^{b2} \ (mod\ p)$$

g denotes a generator of Z_p^* and y is a random element from Z_p^*. The private key is (a_1, a_2, b_1, b_2).

Signing

$$\sigma_1 = a_1 m + b_1 \ (mod\ q)$$
$$\sigma_2 = a_2 m + b_2 \ (mod\ q)$$

where $q|p-1$. The signature on m is (σ_1, σ_2).

The notation $q|p-1$ means that q is a multiple of $p-1$.

Verification

To verify (m, σ_1, σ_2), one does the following.

$$y_1^m y_2 \overset{?}{=} g^{\sigma_1} y^{\sigma_2} \ (mod\ p).$$

If the equation holds, then the signature is accepted. Otherwise, the signature is rejected.

Write a C++, Java or Python program to accomplish the task. You need to take the screen capture of the sample run of your program and put it in a file named Task4.pdf. You need to submit both your source code and the report (Task4.pdf).

5 Task 5. 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.

You may implement you program using C++, Java, or Python. You need to take the screen capture of the sample run of your program and put it in a file named Task5.pdf. You need to submit both your source code and the report (Task5.pdf).

6 Task 6. Various Questions (2 marks)

Answer the following questions. You do not need to implement any program for these tasks. These tasks are pen-and-paper exercises. Please show all your workings for Task 6.1 to 6.4. **Answers without showing the workings, receive no mark**.

1. Assume that the size of message space (domain) for a given hash function is 2^{50}. Also, assume that we want the chance of the adversary finding a collision to be at most 2^{-30}. What is the size of the hash (in bits) required? (0.5 marks)

2. Sign and verify the message $m = 11$ using the RSA signature when $p = 59$, $q = 47$, and $e = 15$. (0.5 marks)

3. Demonstrate that the RSA signature with the parameters given in Q2 is forgeable under chosen message attack with two messages $m_1 = 2$ and $m_2 = 3$. (0.5 marks)

4. Adam and Bob share the same modulus $n = 21$ for RSA, and encryption exponents $e_a = 5$ and $e_b = 4$ with $gcd(e_a, e_b) = 1$. Charlie sends them the same message m encrypted with e_a and e_b respectively, resulting in the ciphertexts $c_a = 14$ and $c_b = 7$. Eve intercepts both c_a and c_b, and applies common modulus attack to recover the message m. What is the message m?

Write your answer in a file called Task6.pdf. You need to show all the key steps in order to obtain full marks. Submit Task6.pdf together with the other tasks in this assignment to Moodle.

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.txt file that explains the compiler setting.

DISCLAIMER: By submitting your assignment, you DECLARE that the assignment is your own works, 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.