CSCI 3330 Project 1

RSA: An asymmetric cryptosystem for cybersecurity

Introduction: Asymmetric cryptosystems are essential for cybersecurity in today's internet age. An asymmetric cryptosystem consists of a public key available for anyone to encrypt a message. However, it requires the private key to decrypt the ciphered text. To ensure security, it is computationally hard to find the private key from a provided public key. In this project, you need to develop a prototype of RSA, a broadly used asymmetric crypto scheme in real world, with simple applications.

Project requirements: This is a group project. You should work in a group of 3 people in this class. Your software solution should be able to perform the following tasks securely:

- Generate both public and private keys by the owner;
- Encrypt a text message for anyone with the public key;
- Decrypt the ciphered message by whom with the private key;
- Generate a digital signature for the owner; and
- Authenticate the originality of a digital signature for these with the public key.

Project objective: Through this project, you should be able to

- Analyze the requirements of an asymmetric cryptosystem and specify the I/O for each requirement above;
- Identify mathematical and algorithmic solutions to meet the requirements effectively and ***efficiently***;
- Design an integrated prototype of RSA cryptosystem that can carry out each of the above tasks, and produce expected output from specified input;
- Implement the design into a software solution effectively with a commonly available integrated development environment (IDLE) in a professional manner;
- Verify that the implementation meets all objectives of the design; and
- Gain collaborative learning and working experience as a member or leader in a group through effective communication and collaboration.

Problem analysis and algorithm identification: To meet the above requirements and objectives, you need to answer each of the following sub-problems and identify effective and efficient algorithmic solutions:

1. What do you need to start with generating keys of RSA? What are the requirements of the seeds (p and q)? How do you meet the requirements?
2. With the seeds p and q, how do you generate a public key? What is the mathematical requirement of a public key? What is the most efficient available algorithm to ensure the requirement?
3. How do you encrypt a message with RSA public key? What is the most efficient algorithm to carry out the task?
4. To decrypt a message ciphered with a RSA public key, you need the private key. What is mathematical requirements for the private key? What is the most efficient algorithm

available to find the private key (extended Euclid's GCD)? When the extended Euclid's GCD returns a negative number, you cannot use it directly. How do you fix it?

5. How do you decipher a cyphered text with the private key? What is the most efficient algorithm available to carry out the operation?
6. Which RSA key should you use to sign a document digitally? How do you sign it?
7. What do you need to verify a digital signature? How do you authenticate it? NOTE: This project is only a prototype. One-way hashing fingerprints is not required.

Design your software solution: After selecting algorithms to meet specified requirements on each of the above subtasks, you should design your software solution prior to implementation.

- Design functional units: It is suggested to design software solution for each subtask as a functional unit with specified I/O. (An input of a function can be an output of another functional unit. In other words, it is not necessarily from keyboard.) For instance, an interactive user menu can be a unit for a user to select a specific task to perform. Key generation can be another functional unit.
- Solution integration: Assemble functional units in a flowchart as an integrated software solution.
- Backend and frontend design: The project involves both backend and frontend. While the backend carries out RSA related operations, the frontend takes care if interactive I/O between a user and the system for specified applications.

Software implementation:

- Python 3 is strongly recommended for its simplicity.
- You may use any code scripts introduced in this course with proper documentation to demonstrate your understanding.
- However, direct copy of any RSA implementation is prohibited because that defeats the purpose of this project.

Testing: To ensure your implementation meets the requirements, you should well test your implemented software. This includes:

- Unit test: to verify if each unit carries out the design objective correctly. You may use small prime numbers as seeds in unit tests when need hand calculated verification;
- Integration test: to check if your implementation meets the specified project requirements fully. Large prime numbers should be used in integration test. The user interface during integration test can be as the following:
    1. Ask the user to identify if he/she is a general public user, or the owner of the keys.
    2. If he/she is a public user, then he/she only has access to the public key. Apply the public key, he/she may
        a. Encrypt a message with the public key, and make the cyphered text for the owner to decipher it; or
        b. Authenticate the digital signature.
    3. Otherwise, the owner has both public and private keys. With them, the user may
        a. Decipher the ciphered text from a public user; or
        b. Generate a digital signature and make it available to others to verify.

4. The software implementation should carry out the user selected function correctly. After completing the task, the software should ask if the user has another task. The program exits only when the user decides to quit.

Sample I/O:

```
RSA keys have been generated.
Please select your user type:
        1. A public user
        2. The owner of the keys
        3. Exit program
Enter your choice: 1
As a public user, what would you like to do?
        1. Send an encrypted message
        2. Authenticate a digital signature
        3. Exit

Enter your choice: 1

Enter a message: topSecret
Message encrypted and sent.
As a public user, what would you like to do?
        1. Send an encrypted message
        2. Authenticate a digital signature
        3. Exit

Enter your choice: 2
There are no signature to authenticate.
As a public user, what would you like to do?
        1. Send an encrypted message
        2. Authenticate a digital signature
        3. Exit

Enter your choice: 3
Please select your user type:
        1. A public user
        2. The owner of the keys
        3. Exit program
Enter your choice: 2
As the owner of the keys, what would you like to do?
        1. Decrypt a received message
        2. Digitally sign a message
        3. Show the keys
```

```
        4. Generating a new set of the keys
        5. Exit

Enter your choice: 1
The following messages are available:
        1. (length = 9)
Enter your choice: 1
Decrypted message: TOPSECRET
As the owner of the keys, what would you like to do?
        1. Decrypt a received message
        2. Digitally sign a message
        3. Show the keys
        4. Generating a new set of the keys
        5. Exit
Enter your choice: 2

Enter a message: mySignature
Message signed and sent.
As the owner of the keys, what would you like to do?
        1. Decrypt a received message
        2. Digitally sign a message
        3. Show the keys
        4. Generating a new set of the keys
        5. Exit
Enter your choice: 5
Please select your user type:
        1. A public user
        2. The owner of the keys
        3. Exit program
Enter your choice: 1

As a public user, what would you like to do?
        1. Send an encrypted message
        2. Authenticate a digital signature
        3. Exit

Enter your choice: 2
The following messages are available:
        1. mySignature

Enter your choice: 1
```

```
Signature is valid.
As a public user, what would you like to do?
        1. Send an encrypted message
        2. Authenticate a digital signature
        3. Exit

Enter your choice: 3
Please select your user type:
        1. A public user
        2. The owner of the keys
        3. Exit program

Enter your choice: 3
Bye for now!
```

WHAT TO SUBMIT: Each group should submit

1. A single copy of documented source code; and
2. A single copy of the project report. *In your project report,* ***you MUST specify individual contributions of each groupmate***.

Here is a sample outline of the report (You may structure your report in a different way, however, the listed information should be provided in a well-organized manner.)

- **Introduction:** What are the requirements of the project? What are the *responsibilities and contributions of each groupmate*?
- **Problem analysis and identification of algorithms:** Specify the requirements of related subtasks, and identify efficient algorithms available to meet the requirements.
- **Solution design:** Dividing the project into manageable subtasks as functional units. Specify I/O's for functional unit. Arithmetic ways to carry out the O from the I of each unit. The design of integrated software solution that assembles functional units (a flowchart is suggested.)
- **Implementation:** Specify software development environment and tools**.** Specify the contributions of each groupmate in the implementation. Code segments are not required. I will check the entire program anyway.
- **Testing:** Unit and integration tests with I/O samples. Bugs or problem identified in testing and corrections made if any. *It is strongly suggested to include the steps of your integration test with the I/O* (screenshots can be helpful). I may reference the steps to verify your program.
- **Summary:** Summarize your learning outcomes through completing this project.

HOW IT WILL BE EVALUATED: I will assess the learning outcomes 1 and 2 as follows:

1.a) Be able to specify the requirements of a complex computing problem through analyzing the problem; and further,

1.b) Be able to identify mathematical and computational solutions to satisfy the requirements of the computing problem effectively.

2.a) Be able to implement algorithmic solutions into a software solution with an integrated development environment (IDLE) effectively; and further

2.b) Be able to evaluate and verify if the software solution satisfies the requirements of the computing problem.

The rubrics with performance indicators are attached below to assign a grade for your group.

		RUBRIC for SO 1			
		Unsatisfactory	**Developing**	**Satisfactory**	**Exemplary**
Performance Indicator		0 59	60 79	80 89	90 100
1.a	Analyze a complex computing problem (specifically, the RSA cryptosystem being assessed in this course)	Unable to correctly analyze and specify critical requirements of RSA and associated applications.	Be able to analyze and specify some requirements of RSA; however, the specified requirements may be incomplete and/or contain conceptual or logical mistakes.	Be able to analyze and specify critical requirements of RSA and associated applications mostly; and be able to present the specified requirements with appropriate justification mostly.	Be able to analyze the RSA cryptosystem and associated applications and to specify all of the requirements without any ambiguity; and be able to present the specified requirements very clearly with solid justification.
1.b	Apply principles of computing to identify solutions (specifically, for the RSA cryptosystem being assessed in this course)	Unable to identify available theoretical, computational, or algorithmic solutions to meet the specified requirements of RSA.	Be able to identify available theoretical and computational solutions to meet some of the specified requirements of RSA; and/or be able to identify some algorithmic solutions that carry out the specified requirements.	Be able to identify theoretical and computational solutions from abstract algebra and number theory to meet the specified requirements of RSA; and be able to identify effective algorithmic solutions that carry out specified requirements though some of them may or may not be the most efficient ones.	Be able to identify theoretical and computational solutions from abstract algebra and number theory to meet all of the specified requirements of RSA precisely; and be able to identify and select the most efficient algorithmic solution(s) among available ones to carry out each of the requirements through asymptotic complexity analysis.

		RUBRIC for SO 2			
		Unsatisfactory	**Developing**	**Satisfactory**	**Exemplary**
Performance Indicator (R)		0 59	60 79	80 89	90 100
2.a	**Implement an appropriate solution for the design** (RSA)	Unable to implement algorithms that carry out the design correctly with appropriate data structures; The implementation is unorganized and/or poorly documented, hence, hard to follow; and/or The implementation contains bugs that cause unexpected termination.	Be able to implement algorithms that carry out the design correctly with appropriate data structures mostly; The implementation is somewhat organized with documentation, however, further clarifications are needed obviously; and/or The user interface and I/O management should be more user friendly.	Be able to implement algorithms that carry out all requirements of the design correctly with appropriate data structures and without runtime error; The implementation is organized with good documentation for others to follow; and/or The implementation is friendly with proper I/O interface.	Be able to implement algorithms that carry out all requirements of the design with effective/efficient data structures and software development tools available; The implementation is very well organized and easy to follow with clear documentation; and The implementation provides a very user friendly I/O interface.
2.b	**Evaluate if the solution meets the given set of requirements**	The software solution is not well evaluated; and/or The evaluations are not well justified.	The software solution is evaluated, however, not all of design objectives are verified; The evaluations do not consider possible scenarios; and/or The evaluations are not documented well with proper justifications.	The software solution is tested to verify if all design objectives are correctly met; Different scenarios are considered in solution evaluation; and The evaluations are documented with appropriate justifications mostly.	The software solution is very well tested and verified, such that, all design objectives are completely satisfied; Different scenarios are thoroughly considered in solution evaluation for robustness; and The evaluations are very well documented and justified.