Ira A. Fulton Schools of Engineering
Arizona State University

CSE 539: Applied Cryptography

RSA Project

Purpose

In this project, you will gain first-hand experience implementing the RSA algorithm. You will be given prime numbers that are greater than 200 bits, testing your ability to work with very large prime numbers. You will learn how to implement the Extended Euclidean algorithm and how to encrypt and decrypt using RSA.

Objectives

Students will be able to:

- Execute the proof of the RSA algorithm.
- Determine why RSA works.
- Implement the RSA algorithm.
- Test ways to work with very large prime numbers.
- Implement the Extended Euclidean algorithm.
- Apply number theory concepts to solve real-world problems.

Technology Requirements

This project requires the knowledge of the following tools and technologies:

- .NET Core 8.0
- Basic Understanding of C#
- Zybooks IDE for editing your code

Directions

Accessing ZyLabs

You will complete and submit your work through zyBooks's zyLabs. Follow the directions to correctly access the provided workspace:

1. Go to the Canvas submission space, "**Submission: RSA Project**"
2. Click the "**Load Submission…in new window**" button.
3. When ready, review the provided code and develop your work where instructed.

Project Description

Before completing this assignment, it is highly recommended that you review the RSA lecture so you understand exactly how the algorithm is constructed and how to verify that your values are correct.

As in the Diffie-Hellman and Encryption Project, you will be given prime numbers in the form (e, c). To calculate the value of the prime number, compute 2^e- c. To implement the RSA algorithm, you will need several values. These values are prime numbers p and q, e coprime to phi(n), and d such that e*d mod phi(n) = 1.

To encrypt with RSA, you need a message m and the values e and n. To decrypt, you need the encrypted message and the values d and n. In this assignment, you are required to perform both encryption and decryption.

You will be given command line arguments in the following order:

1. p_e in base 10
2. p_c in base 10
3. q_e in base 10
4. q_c in base 10
5. Ciphertext in base 10
6. Plaintext message in base 10

The value e will not be given in the command line arguments. Rather, you will use the value 2^{16}+1 = 65,537. As mentioned before, you can calculate p = 2^{p_e}- p_c. You can calculate q in the same way using q_e and q_c.

In order to calculate the value of d, you must employ the Extended Euclidean algorithm. The Extended Euclidean algorithm is covered in the lectures or can be found online. **While you may consult other resources on how the Extended Euclidean algorithm works, you may not copy**

code you find online. It is expected that you will produce your own implementation.

After calculating the value of d, you can verify that it is correct using the equation e*d mod phi(n) = 1. If this equation does not evaluate correctly, your value for d is incorrect.

Once you have calculated all of these values, decrypt the ciphertext given to you and encrypt the given plaintext. Your program should output these two values as a comma-separated pair, with the decrypted plaintext first.

Example

In this example, the same message was used for the plaintext and the ciphertext (after encrypting it):

Command:

```
dotnet run 254 1223 251 1339
6653604712037414553891678798186800420643853924891073471349527688372469
3574434582104900978079701174539167102706725422582788481727619546235440
508214694579 1756026041
```

Expected output:

```
1756026041,665360471203741455389167879818680042064385392489107347134952
76883724693574434582104900978079701174539167102706725422582788481727619
546235440508214694579
```

Submission Directions for Project Deliverables

You must submit your RSA Project deliverable through zyBooks' zyLabs. Carefully review submission directions outlined in this overview document in order to correctly earn credit for your work. Learners may not email or use other means to submit any assignment or project for review, including feedback, and grading.

1. Please use the **program.cs** file provided in your zyBooks workspace to complete your 'RSA Project' program, ensuring that you do not change the class name.
    a. Execute your code by running the below command in the terminal located at the bottom of the IDE.
        i. dotnet run 254 1223 251 1339
        66536047120374145538916787981868004206438539248910734713495276
        88372469
        35744345821049009780797011745391671027067254225827884817276195
        46235440 508214694579 1756026041

2. When you are ready to submit your completed work, click on “**Submit for grading**” located on the bottom left.

3. You will know you have completed the project when feedback appears below the workspace.

4. If needed: to resubmit the project in zyLabs

    a. Edit your work in the provided workspace.
    b. Click “**Submit for grading**” again at the bottom of the screen.

Your submission will be reviewed by the course team and then, after the due date has passed, your score will be populated from zyBooks into your course grade.

Evaluation

The auto-grader will execute your code by passing the values of "pe" "pc" "qe" "qc" "Ciphertext" "Plaintext" through the command line. The grader will compare the two comma separated values i.e the decrypted plaintext and the encrypted ciphertext generated by your code with the expected plaintext and ciphertext values. If these match, you will receive **100 points**. Otherwise, you will receive **0 points** with corresponding error messages.