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

Programming assignments should be done using .NET Core 3.1. While you can write your code on a Windows or Mac machine, the autograder will test your code on Ubuntu 18.04. It is highly recommended to create a Virtual Machine for writing your assignments.

Information on how to install the .NET Core SDK on Ubuntu can be found on Microsoft's website: [Install .NET Core SDK or .NET Core Runtime on Debian]()

You can create a new project using this command: dotnet new console --output project_name

You can also run your project by going into your project directory and using: dotnet run

Directions

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 Project 3 (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.

A full example is provided here. In this example, the same message was used for the plaintext and the ciphertext (after encrypting it):

```
Command: dotnet run 254 1223 251 1339
66536047120374145538916787981868004206438539248910734713495276883724693574434582104900978079701174539167102706725422582788481727619546235440508214694579 1756026041
```

```
Expected output:
1756026041,665360471203741455389167879818680042064385392489107347134952768837246935744345821049009780797011745391671027067254225827884817276195462354405082146945 79
```

Submission Directions for Project Deliverables

Compress your project folder into a .zip archive and submit that file. The autograder will run the following commands on your archive for testing:

```
unzip submission_name.zip

cd submission_name

dotnet run
```

Try these commands before submitting to ensure your submission is properly set up. You can add the command line arguments from the provided examples to verify your program works correctly.

Important Note: Your folder name must match the submission name. For example, if Coursera asks you to submit a file named P4.zip, running the unzip command as shown should produce a folder named P4. If you do not follow this convention, the autograder will be unable to find your project.

Project 4 "submission_name": P4