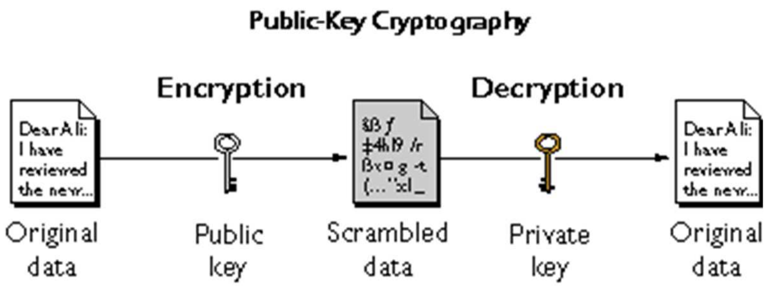
Assignment	Points	Announced	Due
#5	40	Oct-10	Oct-26

RSA

1 Overview

How are messages transferred in a secure way? How is your browser sure that it is talking to the website it should be instead of an imposter? How can we carry key-chains that tell us a password that changes every five seconds? The answer to all of these questions is asymmetrical encryption.

Asymmetric encryption (also known as public-key cryptography) allows us to take a message and turn it into gibberish using a digital key and turn it back into the original message using a different key. We can give out a public key to the world and let them encrypt messages that can only be decrypted using our private key. The public and private keys are mathematically related, but it is computationally intractable to compute a private key from only the public key.



Today we will be implementing one such asymmetrical encryption algorithm known as RSA, named for its inventors Ron Rivest, Adi Shamir and Leonard Adleman. RSA uses randomly generated prime numbers to generate a key pair, and then uses exponentiation and modulation to encrypt and decrypt messages using this key. RSA, while being supplanted by the more sophisticated SHA family of algorithms, is still commonly used for digital security. The mathematics proving RSA to be secure are beyond the scope of this course. The implementation of RSA, however, is of a reasonable level of complexity to be implemented in our lab, given the tools provided by Java's class library.

2 Learning Outcomes

By the end of this project students should be able to:

- write algorithms based on a technical problem description;
- utilize Java types.
- utilize Java's BigInteger to perform high level computation;
- utilize asymmetric encryption to encrypt and decrypt messages;
- work effectively with a partner using pair-programming;
- write an effective report that describes the students' problem-solving process.

3 Pre-Lab Instructions

Do this part before you come to lab:

- Read the Wikipedia article on the RSA algorithm: https://en.wikipedia.org/wiki/RSA_(cryptosystem) and pay special attention to the Example section.
- Read the documentation for Java's BigInteger class.
- List the methods that would be necessary to perform the calculations mentioned in the example section of the RSA Wikipedia article.

4 Submission Instructions

1. Use a basic text editor (e.g. Notepad, Notepad++, Atom…) and not an IDE for this lab assignment.
2. In addition to the lab report, submit a Java file named `RSA.java`.
3. Make sure that your code compiles and runs before submitting it.
4. Don't submit .class files nor .jar files nor archives (e.g. .zip, .rar …).
5. Include Javadoc comments for all fields, methods and the for the RSA class.

5 Lab Instructions

Do this part in lab:

Create the Java class (`RSA.java`) with three methods for:

1. Generating public and private keys. Name this method `GenerateKeys`.
2. Encrypting a number using the public key. Name this method `Encrypt`.
3. Decrypting a number using the private key. Name this method `Decrypt`.

The method `GenerateKeys` should implement steps 1 to 5 from the Wikipedia page. In the first step, two random prime numbers (p and q) with a bit length that is supplied to the `GenerateKeys` method are generated. Then, their multiplication (n) is computed. Next, the totient of this product is calculated. The totient of n is calculated as follows: $tot = \text{lcm}(p-1, q-1)$. The `BigInteger` class doesn't have a method for calculating the LCM (least common multiple) of two numbers, but it has a method for calculating the GCD (greatest common divisor) of two numbers which can be used to calculate the LCM. After that, you will generate a random number (e) that satisfies these conditions: 1) be greater than 1, 2) be smaller than the totient and 3) be coprime to the totient. For e to be coprime to tot, 1) it must be a prime number and 2) can't be a divisor for tot. Finally, calculate (d) which is the modular multiplicative inverse of e under tot. The BigInteger class provides a method for calculating the modular multiplicative inverse.

The two values (n and e) together act as the public key that is used for encryption. The method `Encrypt` should receive a BigInteger number (representing a message) and returns the result of encrypting this number (i.e. a ciphered message). If there are no values in the variables representing n and e (i.e. `n == null || e == null`), then `Encrypt` should return `null`.

The two values (n and d) together act the private key that is used for decryption. The method `Decrypt` should receive a BigInteger number (i.e. a ciphered message) and returns the result of decrypting this number (the original message). If there are no values in the variables representing n and d (i.e. `n == null || d == null`), then `Decrypt` should return `null`.

To test your RSA class, use the provided RSATester class. The RSATester class uses the RSA class to encrypt and decrypt a number and a string. If your RSA class is implemented correctly, running the RSATester class should output "Test #1 passed!" and "Test #2 passed!" in addition to the encrypted and decrypted strings.

6 Lab Report

Each pair of students will write a single lab report together and each student will turn in that same lab report on BBLearn. Submissions from each student on a pair should be identical.

Your lab report should begin with a preamble that contains:

- The lab assignment number and name
- Your name(s)
- The date
- The lab section number

It should then be followed by four numbered sections:

1. Problem Statement

In this section you should describe the problem in *your* own words. The problem statement should answer questions like:

- What are the important features of the problem?
- What are the problem requirements?

This section should also include a reasonably complete list of requirements in the assignment. Following your description of the problem, include a bulleted list of specific features to implement. If there are any specific functions, classes or numeric requirements given to you, they should be represented in this bulleted list.

2. Planning

In the second section you should describe what planning you did in order to solve the problem. You should include planning artifacts like sketches, diagrams, or pseudocode you may have used. You should also describe your planning process. List the specific data structures or techniques you plan on using, and why.

3. Implementation and Testing

In the third section you should describe how you implemented your plan. As directed by the lab instructor you should (as appropriate) include:

- a copy of your source code

- a screen shot of your running application / solution
- results from testing

4. Reflection

In the last section you should reflect on the project. Consider different things you could have done to make your solution better. This might include code organization improvements, design improvements, etc.

You should also ask yourself what were the key insights or features of your solution? Were there alternative approaches or techniques you could have employed? How would these alternatives have impacted a different solution?

5. Partner Rating

Every assignment you are required to rate your partner with a score -1, 0 or +1. This should be submitted in the comment section of the BBLearn submission, and not in the report document. If you don't want to give your partner a negative rating making sure not to use a dash before listing the number! You do not have to tell your partner the rating you assign them. A rating of 1 indicates that your partner was particularly helpful or contributed exceptional effort. A rating of 0 indicates that your partner met the class expectations of them. Rating your partner at -1 means that they refused to contribute to the project, failed to put in a reasonable effort or actively blocked you from participating. If a student receives three ratings of -1 they must attend a mandatory meeting with the instructor to discuss the situation, and receiving additional -1 ratings beyond that, the student risks losing a letter grade, or even failing the course.

6. Contribution

Every assignment you are required to describe your contribution to coding of the solution and writing of the report. You must include the percentage of your contribution. This should be submitted in the comment section of the BBLearn submission, and not in the report document.

7 Grading Rubric

This lab assignment will be graded according to this rubric:

Criteria / Component	Points
Report	10 pts
Correct computation of p & q	5 pts
Correct computation of n	5 pts
Correct computation of tot	5 pts
Correct computation of e	5 pts
Correct computation of d	5 pts
Correct implementation of the method Encrypt	2.5 pts
Correct implementation of the method Decrypt	2.5 pts

Lab assignment penalties:

Item	Points
Missing names	-4
Missing partner rating	-2
Missing screenshots	-2
Insufficient / No commenting	-2 / -4
Too late to pair (if attended)	-4
Absent	-12
Non-compiling program	-40
Improper format (e.g. NetBeans ZIP archive instead of a Java file)	-5
Missing or insufficient Javadoc comments	-5
Missing contribution description and/or percentage	-5

Note:

If your partner is not responding to your emails and/or not collaborating, don't hesitate to reach out to the lab TA aide and/or the primary instructor.

Colophon
This project was developed by Richard Lester and Dr. James Dean Palmer and improved by Dr. Mohamed M. Elwakil of Northern Arizona University. Except as otherwise noted, the content of this document is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.