CS337: Project 2

TA : Lara Schmidt (lschmidt@cs.utexas.edu)

February 19, 2013

1 Introduction

Encryption is used when secrecy is desired, for example, in communication between two parties. The RSA encryption algorithm is arguably the most popular encryption algorithm in current use. It was invented in 1978 by Ron Rivest, Adi Shamir and Leonard Adleman. For this project, you will implement the RSA algorithm to encrypt and decrypt files. You are expected to be familiar with the algorithm as described in the class notes (Section 3.3.2).

2 Description

This project consists of three parts :

- Generating an RSA public key and private key pair.
- Writing a program to encrypt files.
- Writing a program to decrypt RSA-encrypted files.

Each of these parts will now be described in detail.

3 Generating the RSA key pair

In this part of the project, you are required to generate an RSA public key - private key pair. Short of using an existing RSA key generator program (say, from/on the Internet), you are free to use any method that you can think of (you can write code in any language, you can do it manually, etc).

The key is a triple of numbers n, e and d(in the same notation as the class notes). n is **required** to be between 2^{24} and 2^{30}. The purpose of this restriction is to allow you to do all the required computations for encryption and decryption using just the Java built-in *long* data type.

3.1 Submission

To submit this part of the project, create a text file named "key.txt" which has n on the first line, e on the second line, and d on the third line. An example file would look like

246229427
11
111907091

Create the file on a Linux machine so as to avoid DOS-style linebreaks. Note that it is absolutely necessary for you to follow these specifications because the project will be graded automatically - if any of your files do not meet the specifications, your submission will be considered erroneous.

To submit this file, use turnin (see Section 7).

4 File encryption

In this part of the project, you will write a Java program to encrypt an arbitrary file.

How do you go about doing this? Note that the RSA algorithm specifies how to encrypt a single number ($< n$). To encrypt a file, it is sufficient to break up the file into blocks so that each block can be treated as a number and encrypted by the RSA algorithm. For this project, you will use a block size of 3 bytes. To treat each block as a number, simply concatenate the bit representation of the 3 bytes to form a single number. Note that this number will be between 0 and 2^{24} and so, will be less than n (why?), allowing us to to use the RSA encryption algorithm for a single number.

After encrypting the block, the encrypted number has to be written out to the output file. This number is between 0 and n, and therefore, potentially between 0 and 2^{30}. To write out this number, simply break the number into 4 bytes based on the bit representation of the number and write out the 4 bytes in order from *left to right*.

Let us now walk through the entire process with an example. Suppose the input file consists of the following bytes (in order) : 75, 34, 107, 23 .... To generate the first encrypted number, pick up the first block consisting of three bytes : 75, 34, 107. Form the number from this block by concatenating the bit representations of these bytes - 01001011, 00100010, 01101011. This number is 00000000 01001011 00100010 01101011. Now, encrypt this number by the RSA algorithm to get the number (say) 01011000 00110110 00001000 10001100. To write out this encrypted number to the file, break it up into 4 bytes, reading the bits from left to right - 01011000, 00110110, 00001000 and 10001100. Write out these 4 bytes in this order to the output file.

4.1 Submission

The encryption algorithm should be runnable from a file named "Encrypt.java" (that is, this file should have the *main* method). The program will be called with the input plaintext file and the key file as arguments. The key file will be in the same format as described in Section 3.1.

The program will be invoked as :

```
$java Encrypt plaintext key.txt
```

where `plaintext` is the name of the file to be encrypted.

The encrypted file that the program generates **should** be named "encrypted".

5 File decryption

In this part of the project, you will write a program to decrypt the encrypted file that has been generated by the encryption procedure described earlier.

How do you go about decrypting the encrypted file? This procedure is the exact reverse of the encryption procedure. You will read from the encrypted file in blocks of 4 bytes, with each set of 4 bytes forming an encrypted number. The RSA decryption algorithm specifies how to decrypt an encrypted number. This will give you the number that was formed from the original plaintext file. Now, remember that this number was formed by concatenating 3 bytes from the plain text file. To get back those 3 bytes from this number, you will have to pick out the lower order 3 bytes from this number and write them out in order to the output file (the topmost byte of this number will always be 0, can you see why?).

Let us walk through this procedure with an example. Suppose the encrypted file consists of the following bytes : 88, 54, 8, 140, ... (these were the same bytes that were written out in the example for the encryption procedure). The encrypted number is formed by concatenating the bit representations of these numbers together - 01011000 00110110 00001000 10001100. Decrypting this number by the RSA algorithm gives the original number - 00000000 01001011 00100010 01101011. This number is then split into the original 3 bytes by taking the lower order 3 bytes in the same order - 01001011, 00100010, and 01101011.

5.1 Submission

The decryption algorithm should be runnable from a file named "Decrypt.java" (that is, this file should have the *main* method). The program will be called with the input encrypted file and the key file as arguments. The key file will be in the same format as described in Section 3.1.

The program will be invoked as :

```
$java Decrypt encrypted key.txt
```

where encrypted is the name of the file to be encrypted. Note that your program should work with any key.

The decrypted file that the program generates **should** be named "decrypted".

6 Points to take note of

Your submission for this project is complete if your submission for the first part is a valid RSA key pair and if your encryption and decryption programs work correctly as specified. Note that your program should stick to the specifications laid out in this description *in all respects*, from the order in which bytes are written to the order of the command line arguments of the program. **Your encryption routines should work correctly when used with other correct decryption routines and vice versa**. It is therefore necessary for you to follow every specification exactly.

The RSA encryption and decryption algorithms involve modular exponentiation. Straightforward exponentiation will require you to manage really huge numbers that will be outside the limits of the basic Java data type 'long'. **For this reason, it is necessary that you use the algorithm for fast modular exponentiation that is described in the class notes (Section 3.3.2)**. You **should not** use any other large number data types like the Java BigInteger class or any similar built-in crypto libraries. Any submissions that use such data types or libraries will face a significant penalty. The entire project has been designed so that all operations can be performed using Java's standard 'long' data type.

It is not necessary for your programs to handle erroneous input formats. In cases like these, or say, when an attempt is made to decrypt a file with a key file different from that used to encrypt

the file, it is okay for your program to crash. We will not test your code for such cases.

7 Turning in the project files

You will be turning in the key pair file - "key.txt" and the code for your encryption and decryption programs (which should include the files "Encrypt.java" and "Decrypt.java"). All these files should be turned in to 'project2', to the grader 'lschmidt'. A sample turnin command line is
`$turnin --submit sanketh project2 <file-name>`
or
`$turnin --submit lschmidt project2 Encrypt.java Decrypt.java key.txt readme.txt`

Do not turn in a compressed or tarred version of your files. However if you have other java files please sumbit them.
Do not submit directories structures, class files, or any text files except key.java. Only submit the files needed to compile and key.txt

The deadline for this project is midnight, **March 3 (Sunday)**. No changes are expected to the project description, but be sure to watch Piazza just in case there are any. You are responsible for noting any such updates.

All the best working on this project!!