IE 170 Laboratory 10: Public Key Cryptography

Dr. T.K. Ralphs

Due April 24, 2006

1 Laboratory Description and Procedures

1.1 Learning Objectives

You should be able to do the following after completing this laboratory.

1. Understand the basic principles of public key cryptography.
2. Understand how the RSA encryption algorithms work.
3. Undertand how to produce a digital signature.
4. Understand the limitations of the built-in data types for implementing encryption algorithms.
5. Understand the number theoretic algorithms underlying cryptographic methods.

1.2 Key Words

You should be able to define the following key words after completing this laboratory.

1. Public/private key
2. RSA encryption
3. Digital signature
4. Modular arithmetic
5. Euclidean algorithm

1.3 Scenario

In this lab, you are to implement a protocol for communicating securely over the Internet. The ability to conduct business over the Internet in a secure fashion is crucial to the continued growth of e-commerce and other applications in which private data must be transmitted over public parts of the network. The security of these transactions is provided through the use of encryption. As use of the Internet for conducting business has grown, these techniques have come into wide use and have become increasingly sophisticated.

Although transmitting data securely over an electronic network is obviously a relatively new concept, cryptography has been used for secure transmission of data for at least two thousand years. Julius Caesar is credited as one of the first users of this technology. Early coding schemes, however, required a prearrangement between the sender and the receiver since encryption/decryption

required a shared key that had to be known a priori to both parties. This is obviously not desriable for conducting business transactions since the prior arrangement would have to be done off the network by US mail or by phone to ensure security.

To solve this problem, the concept of public key cryptography was invented n 1976 by Diffie and Hellman. In public key cryptography, the encryption and decryption keys are different and one cannot be easily derived from the other. Thus, an individual can share his or her encryption key (the *public key*) without revealing the decryption key (the *private key*). This allows anyone with the public key to send a secure encrypted message to the owner of that public key that cannot be easily decrypted without the private key, which is not revealed. This enables secure communication without prior arrangement. The most common used public key encryption algorithm is the RSA algorithm described by Rivest, Shamir, and Adleman in 1978.

Public key cryptography enables any two people to establish secure communication without prior arrangement over a network, but this system is still not completely secure. Although it is not possible for anyone else to read the encrypted message, it is also not possible to verify the true origin of the message since anyone with knowledge of the receiver's public key can send a valid encrypted message to the receiver. Hence, it is possible for the message to be intercepted and replaced with another different encrypted message enroute without detection.

To solve this problem, Rivest, Shamir, and Adleman invented the concept of a *digital signature*. First, note a key (public or private) can be used for encryption or decryption. If the public key is used for encryption, the private key decodes the message and if the private key is used for encryption, the public key decodes the message. To digitally sign a message, the sender encypts the message using his or her private key. This is the digital signature. The sender then transmits both the message and the digital signature. To verify that the message was not modified in transit, the signature is decoded with the public key and compared to the unencrypted version of the message. If they are identical, then the message is authentic. Otherwise, either the unencrypted message or the signature must have been corrupted in transit. To digitally sign a message *and* encrypt it, the sender first creates the digital signature using his or her own private key and then encrypts the message and the signature using the receiver's public key. Of course, you still have to be able to verify the identity of the person holding a given public key. This is usually done through the use of a mutual trusted authority that verifies the validity and ownership of keys.

1.4 Design and Analysis

This lab will be focused on the design and analysis of algorithms for public key encryption. Specifically, you will be implementing a basic version of the RSA encryption scheme. Implementiung the RSA algorithm involves two separate methods. First, one must be able to generate public and private key pairs that function as described above. The steps in key generation are as follows:

1. Generate two large prime numbers p and q with $p \neq q$ and set $n = pq$.
2. Choose an odd integer e relatively prime to $\phi(n) = (p-1)(q-1) = n + 1 - p - q$.
3. Compute d as the multiplicative inverse of e modulo $\phi(n)$.

The public key is then the pair (e, n) and the private key is the pair (d, n). We will see how to use these for encoding and decoding below. Generating large prime numbers is difficult, so the first step is usually accomplished by randomly generating large numbers and applying primality tests until one is found that passes all tests. Such a number is prime with "high probability" (good enough for our purposes). The probability that a random integer k is prime is approximately $1/\ln k$, so this should only require approximately $\lg n$ steps on average. The second step is also

accomplished by generating random numbers and checking for satisfaction of the given condition. This also should not take "too many" steps on average. The third step is accomplished using a version of the Euclidean algorithm. The running time of this step is $O(\lg^3 n)$,

Given a public key (e, n) and a private key (d, n), the encoding function is

$$f(P, e, n) = P^e \mod n \tag{1}$$

where P is an integer equivalent of the plaintext message unit to be encoded. The decoding function is

$$f(C, d, n) = C^d \mod n. \tag{2}$$

Note that the encoding and decoding functions are actually the same function applied with different exponents. Given a key and a plaintext message unit P, the encoding function f can be computed by repeated squaring in $O(\lg e)$ steps, but note that each step involves multiplying two numbers of size $O(n)$, so for very large n, this can be more time-consuming than multiplication with standard 32 bit integers.

1.5 Program Specifications

Your task is to provide a class implementing the RSA encryption algorithm specified above. Your class should be capable of generating key pairs, encoding, and decoding.

1.6 Program Function

You are to implement a class supporting the interface provided in the file `rsa.hpp`.

1.7 Algorithms

The algorithms involved in implementing RSA encryption are specified above.

1.8 Data Structures

No special data structures are required for this lab.

2 Laboratory Files

The files for this laboratory are in the zip archive `Lab10.zip` available on the course Web site. The archive will unpack into a directory called `Lab10` with `shell` and `data` subdirectories. The shell directory contains the files

1. `rsa.cpp`: the implementation file (with some functionality not implemented yet).
2. `rsa.hpp`: the interface for the RSA class. classes.
3. `main.cpp`: the client program used for testing.

The `data` subdirectory contains test files containing messages that can be read in and encoded/decoded.

3 Laboratory Assignment

3.1 Programming (50 points)

1. Implement a class supporting the given interface.
2. Verify that your class compiles and runs correctly using the test files provided for the lab.

3.2 Analysis and Follow-Up Questions (50 Points)

1. (10 points) Generate a key pair using your program. Using the resulting private key, encode a short message and send it to the TA, along with your public key.
2. (10 points) Generate a digitally signed AND encrypted version of the above message and send it to the TA (in a separate e-mail).
3. (10 points) There are two methods of implementing the `getPower_()` method that computes $P^e \mod n$ in the RSA class. The first is the "naive" method in which P is multipled by itself e times. The second is the method of repeated squaring described in CLRS. Implement both methods and compare the time required to encode the message in the file `small_message.txt` for each of the methods. What is your observation?
4. (10 points) CLRS 31-1
5. (10 points) CLRS 31.7-1