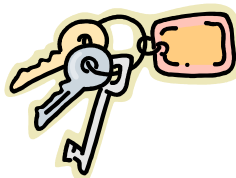


Educational Software For A Cryptographic Laboratory

(Specification – Version 1)



Luu Pham – lpham2@gmu.edu

GMU – Nov. 2003

Table of Contents

- 1. ABSTRACT**
- 2. INTRODUCTION**
- 3. DESIGN SPECIFICATIONS**
 - i. Languages, compiler, and platform to the run application**
 - ii. Range of variables for arithmetic operations**
 - iii. Main functions**
 - iv. Inputs/Outputs**
- 4. TESTING AND SIMULATION PLAN**
- 5. LITERATURE**
- 6. PROJECT SCHEDULE**

1. ABSTRACT

The purpose of this project is to develop an educational program to demonstrate RSA public key encryption/decryption and RSA signature with small key lengths for a Cryptographic laboratory. The application is designed and implemented in such a way so that it is a user friendly interface; it offers the user flexibilities in data input/output and capabilities to visualize main calculations in the crypto-algorithm. In addition, main calculations needed for the RSA algorithm are also implemented; this allows user to compute calculations when performing public key encryption/decryption or RSA signature.

2. INTRODUCTION

Public key cryptography, especially RSA algorithm, has been widely used in industry since 90s. In last fifteen years, many studies and researches have been performed and published on RSA algorithm; however, there is very few software that efficiently shows the main calculation process of the algorithm. When studying public key cryptography, students have very few chances to fully explore and verify intermediate values a long with its calculations. In real life, most of RSA cryptographic products are tightly integrated with other applications, and users are unable to see any cryptographic transformations. In attempt to bridge this gap, this application is designed to be implemented to offer students a user friendly interface software, which allows users to manipulate input values to visualize all process of cryptography and trace through intermediate values a long with its calculations. In addition, main calculation utilities needed for RSA encryption/decryption and RSA signature are also implemented to offer the user useful tools to compute necessary values when performing public key encrypting or decrypting.

3. DESIGN SPECIFICATIONS

i. Languages, compiler, and flat form to run the application

a. Languages:

Delphi 6 and Assembly are used to implement the application.

b. Compiler:

Delphi 6 - Borland.

c. Platform:

Since our purpose is to design an application with good performance, reliability, and user friendly user interface; therefore, Windows is the main platform for our application.

ii. Range of variables for arithmetic operations

The range of variables for arithmetic operations in the program is 64-bit integer, whose range is -2^{63} and 2^{63} .

16-bit key length is initially selected to implement the RSA algorithm.

iii. Main functions

a. Main functions needed for Database management:

- + **Add:** Add new user to the database
- + **Delete:** Delete a user currently in the database
- + **Edit:** Edit to revise user information in database
- + **Change key:** Change key values for a user in the database
- + **Search:** Search to display all user information currently in the database.

b. Main functions for calculation utilities

+ **Check prime number:**

Input: An integer number.

Output: Confirmation whether input number is an prime number or not.

+ **Multi-Inverse:** Find x so that $a \cdot x = 1 \pmod N$

Input: a, N : integer.

Output: x so that $a \cdot x = 1 \pmod N$.

+ **GCD:** Find $\text{GCD}(a,b)$

Input: a, b : integer.

Output: $\text{GCD}(a,b)$

+ **Congruence:** Find x so that $a.x=b \pmod N$

Input: a, b, N : integer.

Output: x so that $a.x=b \pmod N$.

+ **Left-To-Right Binary Exponential:** Find $Y=M^E \pmod N$

Input: M, E, N : integer.

Output: Compute $Y=(M^E \pmod N)$ by using Left-To-Right Binary Exponential

+ **Right-To-Left Binary Exponential:** Find $Y=M^E \pmod N$

Input: M, E, N : integer.

Output: Compute $Y=(M^E \pmod N)$ by using Right-To-Left Binary Exponential

c. GetPublicKey: $[e,n]=\text{GetPublicKey}(IP, \text{PKIDirectoryFile});$

Input: Either IP Address or Email Address and PKI Directory file.

Output: Public key e and n for the desired IP/ email address.

d. RSA_Key_Generator: $[e,n,d]=\text{fk}(p,q)$

Input: p, q : large prime numbers.

Output: Public key $[e,n]=\text{fpub}(p,q)$

Private key $[d,n]=\text{fprv}(p,q)$

e. MD5 Hash function: $h=H(m)$

Input: Message m : arbitrary length

Output: h : 128-bit fixed length $h=H(m)$

f. Encryption: $C=\text{fe}(M,n,e)$

Input: Message arbitrary length, public key (e,n)

Output: Cipher text $C=\text{fe}(M,n,e)$

g. Decryption: $M=\text{fd}(C,n,d)$

Input: Cipher text arbitrary length, private key (d,n)

Output: Message $M=\text{fd}(C,n,d)$

h. Verify Digital Signature: Boolean function $b=\text{fv}(e,n,Ds)$

Input: Public key (e,n)

Message Digital Signature Ds

Output: True or False;

iv. Inputs/Outputs

a. Inputs:

There are two input options that user can select when running the program; input by typing directly from the key board or reading entire messages from a text file.

b. Outputs:

All information during the process of encryption and decryption will not only be logged into a log file, but also can be seen on the screen if users would like to do so.

4. TESTING AND SIMULATION PLAN

Functional test for each function will be carried out during the implementation process.

Available test vectors, sample input/output from reference sources will be utilized to test main functions to verify the accuracy of the application.

The system clock will be used to measure the timing for each part of the encryption and decryption process. That will help the user easily evaluate the performance of each function in the program, and also for the entire program.

5. LITERATURES

- [1] Dr. Kris Gaj, ECE646 Course lecture notes - Fall 2003.
- [2] Laboratory instruction for ECE590
- [3] DES: <http://www.itl.nist.gov/fipspubs/fip46-2.htm>
- [4] RSA: http://www.rsasecurity.com/rsalabs/rsa_algorithm/index.html
- [5] MD5: <http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1321.html>
- [6] AES: <http://csrc.nist.gov/CryptoToolkit/aes/>
- [7] IEEE: <http://standards.ieee.org/regauth/posix/>
- [8] <http://www.rsasecurity.com/rsalabs/cryptobytes/>
- [9] Ron Rivest, "SDSI - A Simple Distributed Security Infrastructure" alternative to the creation of a global certificate hierarchies envisioned by the X-509 group.
- [10] Bruce Schneier, Applied Cryptography - Protocols, Algorithms, and Source Code in C, 2nd ed., John Wiley & Sons, Inc., New York, 1995.
- [11] M. Welschenbach, Cryptography in C and C++, Apress, 2001
- [12] Douglas R. Stinson, Cryptography - Theory and Practice, CRC Press, Inc., Boca Raton, 1995

- [13] William Stallings, *Cryptography and Network Security: Principles and Practice*, 3rd ed., Prentice Hall, Upper Saddle River, 2003.
- [14] K. Jallad, J. Katz, and B. Schneier, *Implementation of Chosen-Ciphertext Attacks against PGP and GnuPG*, Information Security Conference 2002, Springer-Verlag, 2002.
- [15] Canetti, Ran Catherine Meadows, and Paul Syverson, "Environmental Requirements for Authentication Protocols", *Proceedings of the International Symposium on Software Security*. Springer-Verlag, November 2002
- [16] David M. Burton, *Elementary Number Theory*, International Series in Pure and Applied Mathematics, 3rd. ed., The McGraw-Hill Companies, Inc., 1997
- [17] Meadows, Catherine, "What Makes a Cryptographic Protocol Secure? The Evolution of Requirements Specification in Formal Cryptographic Protocol Analysis." *Proceedings of ESOP 03*, Springer-Verlag, April 2003.
- [18] Charlie Kaufman, Radia Perlman, and Mike Speciner, *Network Security: Private Communication in a Public World*, PTR Prentice Hall, Englewood Cliffs, 2002
- [19] Ueli Maurer, *Some Number-theoretic Conjectures and Their Relation to the Generation of Cryptographic Primes*, *Cryptography and Coding II*, Oxford University Press, 1992.
- [20] Ueli Maurer, *Fast Generation of Prime Numbers and Secure Public-Key Cryptographic Parameters*, *Journal of Cryptology*, vol. 8, no.3, 1995
- [21] Song Y. Yan, *Number Theory for Computing*, 2nd ed., Springer-Verlag, Berlin, 2002
- [22] David M. Burton, *Elementary Number Theory*, International Series in Pure and Applied Mathematics, 3rd. ed., The McGraw-Hill Companies, Inc., 1997
- [23] N. Ferguson and B. Schneier, *A Cryptographic Evaluation of IPsec*. J. Kelsey, B. Schneier, and D. Wagner, *Mod n Cryptanalysis, with Applications Against RC5P and M6*, *Fast Software Encryption*, Sixth International Workshop Proceedings (March 1999), Springer-Verlag, 1999.

6. PROJECT SCHEDULE

Task	Week beginning											
	09/29	10/06	10/13	10/20	10/27	11/03	11/10	11/17	11/24	12/01	12/08	12/15
Initial Project specification	■	■										
Explore and compile referenced sources	■	■										
Analysis sources, select appropriate info.	■	■	■									
First progress report		■	■									
Program structure, algorithms design		■	■	■	■							
Coding, Debugging, and partly testing			■	■	■	■	■	■	■	■		
Testing and improving				■	■			■	■	■		
Second progress report					■							
Third progress report							■					
Final progress report and draft of presentation										■		
Final Report and presentation											■	