Implementation of RSA Algorithm

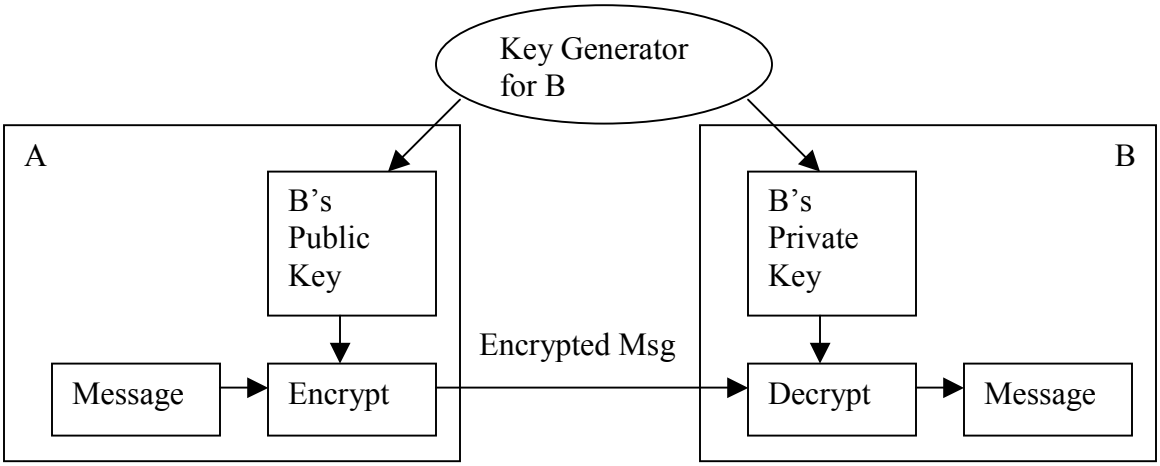
Name	Bhaskar Bora
Date	March 2, 2003
Place	Bangalore, India

1 Introduction

This assignment is to study and implement RSA algorithm in software. Implementation includes private and public key (32-bit) generation and support of text file encryption and decryption. This document contains the algorithms and information required to design and implement RSA algorithm in VC++. This software is only for text data encryption and decryption. Document also contains the software usage guidelines and screen-shots.

RSA algorithm is mainly a public key encryption technique used widely in network communication like in Virtual Private Networks (VPNs). In public key encryption technique, a key is split into two keys and they are called as public and private keys. Public key is advertised to the world and private key is kept secret. It is not possible to generate private key using the public key. So, someone who knows the public key cannot decrypt a message after it has been encrypted using the public key.

A diagrammatic representation of public key encryption is shown below



Let us take a case where A needs to send a message to B using a public key encryption algorithm. Key generator generates the keys for B and distributes the public key to each person who needs to send a message to B. The private key is kept secret and only B has it. In this case, key generator gives the public key to A so that A can send message to B.

In RSA algorithm decryption is possible only through private key. And there is no way by which private key is generated using public key. So the message transmitted from A to B using RSA encryption is secure even though others know B's public key. For two-way communication between A and B there will be another set of keys for A.

2 Description of The Algorithm

The paper by Diffie and Hellman introduced a new approach to cryptography. This, in effect, challenged cryptologists to come up with a cryptographic algorithm that met the requirements of public-key systems. Ron Rivest, Adi Shamir and Len Adleman developed the algorithm and gave the implementation details in the year 1978. Since then, Rivest-Shamir-Adleman (RSA) scheme it is considered as the only widely accepted and implemented general-purpose approach to public-key encryption.

RSA algorithm is a block cipher technique in which plain text and cipher text are integers between '0' and 'n-1' from some 'n'. In RSA algorithm encryption and decryption are of following form, for some plain text M and cipher text C:

C = M^e mod n
M = C^d mod n

Both sender and receiver must know the value of 'n'. The sender knows the value of 'e' and only receiver knows the value of 'd'. Thus, this is a public-key encryption algorithm with a public key of KU={e, n} and private key of KR={d, n}. For the algorithm to be satisfactory for public-key encryption, the following requirement must be met

1. It is possible to find values of e, d, n such that M^ed = M mod n for all M<n.
2. It is relatively easy to calculate M^e and C^d for all values of M<n.
3. It is infeasible to determine d given e and n.

The key generation process is as follows	
Select p and q	p and q are prime numbers
Calculate n = p x q	
Calculate Φ(n) = (p-1) x (q-1)	
Select integer e	gcd(Φ(n),e)=1; 1<e<Φ(n); e and Φ(n) are relative prime
Calculate d	d = e-1 mod Φ(n)
Public key	KU = {e,n}
Private key	KR = {d,n}

Encryption process	
Plaintext	M<N
Ciphertext	C = M^e (mod n)

Decryption process	
Plaintext	C
Ciphertext	M = C^d (mod n)

3 RSA Software

RSA software is a prototype software to demonstrate key generation using RSA cryptography algorithm. Software is capable of generating 32-bit long keys. It has the following features.

1. RSA software can generate two prime numbers randomly. Also, user can specify one or both prime numbers.
2. Software calculates and displays the value of 'n'.
3. It calculates and displays the value of 'e'.
4. It generates and displays the value of 'd' using 'e' and 'n'.
5. Displays both public and private keys for analysis.
6. User can encrypt a text file using the public key.
7. User can decrypt the encrypted file using the private key.
8. User can compare the original text file and decrypted text file.

An example is shown below to describe the key generation process

1. Selecting two prime numbers, p=13 and q=19
2. Calculate n = p x q = 13 x 19 = 247
3. Calculate $\Phi(n) = (p-1) \times (q-1) = 216$
4. Select 'e' such that e is relatively prime to $\Phi(n)=216$ and less than $\Phi(n)$; in this case, e = 5
5. Determine d such that
   $d \times e \equiv 1 \bmod 216$ and $d<216$.
   Where, $d \times e = k\Phi(n)+1$

   The correct value of d = 173
6. Thus, the resulting public key KU = {5, 247} and private key KR = {173, 247}

3.1 Software Usage Guidelines

Guidelines for using RSA software to generate public and private keys

1. Select two distinct prime numbers and enter in the edit box provided. Also, user can selectively ask the software to generate ether or both the prime numbers. Software generated primes numbers will always be distinct. Note that even if two prime numbers provided are same, software will go ahead and do the key generation. This provision is kept for study purpose, just to see what happens when primes numbers are same. Typically, RSA algorithm will not work if two prime numbers are same.

2. On pressing generate keys button, software will generate the keys and display in the edit boxes tagged KU (key-public) and KR (key-private). Software also gives full details for various other integers generated and used during key generation. Up till now only keys have been generated.

Guidelines for encrypting a text file

1. To encrypt a text file user needs to press encrypt button, which will open up a file dialog box. User then selects a text file. Software will use the last generated public key to encrypt the text file.

2. Upon selecting a text file, software will encrypt the entire file and save the encrypted data into a file with extension ".enc". Encrypted file contains data in binary form and stored as 8-byte floating-point data. This is very much implementation specific and not a standard.

Guidelines for decrypting an encrypted file

1. To decrypt a text file user needs to press decrypt button, which will open up a file dialog box. User then selects an already encrypted (.enc) file. Software will use the last generated private key to decrypt the text file.

2. Upon selecting the encrypted file, software will decrypt the entire file and save the decrypted data into a file with extension ".dec". Decrypted file contains data in text form, which can be opened in any text editor (like Notepad.exe).

To compare two files (original text file and decrypted file)

1. To compare the original text file and decrypted file compare files button should be pressed.

2. Software will compare last used text file for encryption and last generated .dec file. The result will be displayed in the window accordingly after the compare operation.

4 Algorithms used in RSA software

Different algorithms used in the software are listed below.

4.1 Main Key Generation Algorithm

Algorithm: MAIN_KEY_GENERATION

1. Generate prime number (p and q) if not specified by the user (refer algorithm PRIME_NO_GENERATE).
2. Calculate n ← p x q
3. Calculate Φ(n) ← (p-1) x (q-1)
4. Select the value of 'e' (refer algorithm SELECT_e).
5. Calculate the value of 'd' (refer algorithm CALCULATE_d).
6. Display public key KU={e,n} and private key KV={d,n}
7. End

4.2 Prime Number Generation Algorithm

Algorithm: PRIME_NO_GENERATE

1. Generate a random number (p) using pseudorandom number generator
2. If the random number is not odd then add one to the number to make it odd.
   If 'p' no odd then p←p+1;
3. Let I ← 2
4. Compute remainder ← p%I.
5. If remainder = 0 then goto step 2;//this odd number is not prime
   Else I ← I+1;
6. If I = p/2+1 then goto 7;
   Else Goto 4;
7. Return the odd number as prime number
8. End

4.3 Algorithm for Selecting the Value of 'e'

This algorithm is to determine the value of integer 'e' using Φ(n). As stated earlier Φ(n) and e should be relatively prime.

Relatively Prime: To explain this let us consider two numbers a and b. If a and b are called as relatively prime then

Mod(a,b) = 1

Algorithm: SELECT_e
Input: a, b (where X>Y)

1. X ← b; Y ← a (where X>Y)
2. If Y=0 return X=gcd(X,Y)
3. R = X mod Y
4. X ← Y
5. Y ← R
6. goto 2

4.4 Algorithm for Calculating the Value of 'd'

Following equation is used for calculating the value of 'd'

$ed = k\Phi(n) + 1$

Where, e and Φ(n) are known

Also, k is varied from 1 to 100000

Algorithm: Calculate_d

1. k ← 1
2. (Float calculation) d = (Φ(n) x k +1)/e
3. (Integer calculation) d_dash = (Φ(n) x k +1)/e
4. If d = d_dash then Return d as the calculated value
5. k ← k+1
6. If k=100001 goto 8
7. Goto 2
8. Couldn't find d

4.5 Algorithm for Encrypting Data

The algorithm presented in this section describes how data is encrypted in the software using the public key. Basically for any encryption the formula used is given below

Plaintext M, where M<N
Ciphertext C = M^e (mod n)

Algorithm: Encrypt_Data
Input: data_in
Output: data_out

1. Convert ‘e’ into binary and store in ‘bits’
2. Get length of the converted binary number in ‘length’
3. c ← 0, d ← 1, i ← length-1
4. c ← c*2
5. d ← mod(d*d, n)
6. If bits_I = 1 then
   {
   c=c+1;
   d=fmod(data_in*d,n);
   }
7. i ← i-1
8. If I != -1 goto 4
9. data_out = d
10. Return data_out

4.6 Algorithm for Decrypting Data

The algorithm presented in this section describes how data is decrypted in the software using the private key. Basically for any decryption the formula used is given below

Plaintext C
Ciphertext M = C^d (mod n)

Algorithm: Decrypt_Data
Input: data_in
Output: data_out

1. Convert ‘d’ into binary and store in ‘bits’
2. Get length of the converted binary number in ‘length’
3. c ← 0, data_out ← 1, i ← length-1
4. c ← c*2
5. data_out ← mod(data_out * data_out, n)
6. If bits_I = 1 then
   {
   c=c+1;
   data_out =fmod(data_in* data_out,n);
   }
7. i ← i-1
8. If i != -1 goto 4
9. Return data_out

5 Example of RSA encryption and decryption

This section presents an example of RSA encryption and decryption process with key generation.

P = 61 ← first prime number (destroy this after computing E and D)
Q = 53 ← second prime number (destroy this after computing E and D)
PQ = 3233 ← modulus (give this to others)
E = 7 ← public exponent (give this to others)
D = 1783 ← private exponent (keep this secret!)

Your public key is (E, PQ).
Your private key is (D, PQ).

Public Key	**Private Key**
e=7, n=3233 thus, KU = {7, 3233}	d=1783, n=3233 thus KR = {1783, 3233}

The encryption function is:

encrypt(T) = (T^E) mod PQ
= (T^7) mod 3233

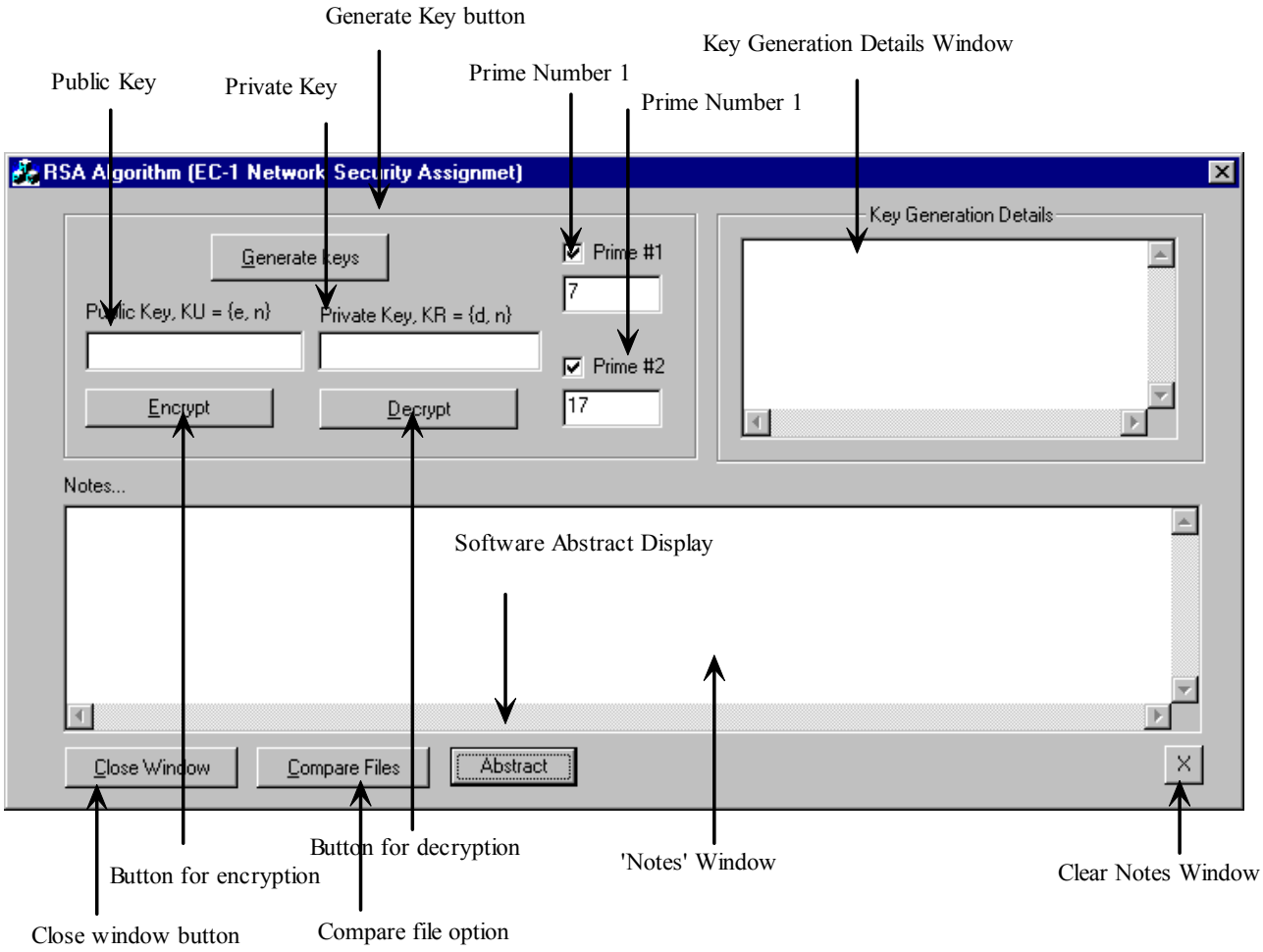
The decryption function is:

decrypt(C) = (C^D) mod PQ
= (C^1783) mod 3233

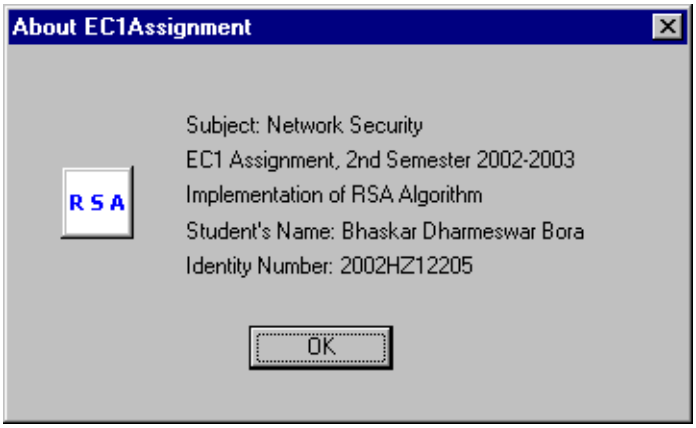
6 Appendix-A (RSA Software Screen Snap-Shots)

6.1 Main Screen

RSA Software screen shot with various control tags is shown below

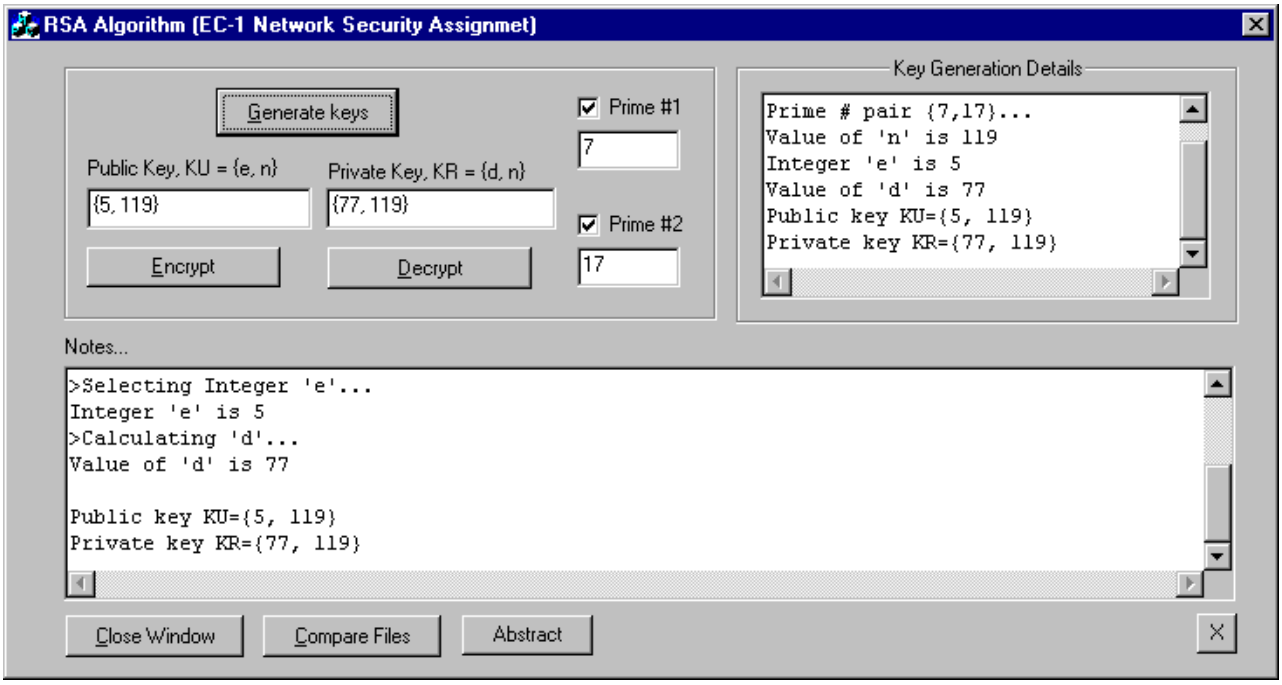


About Box



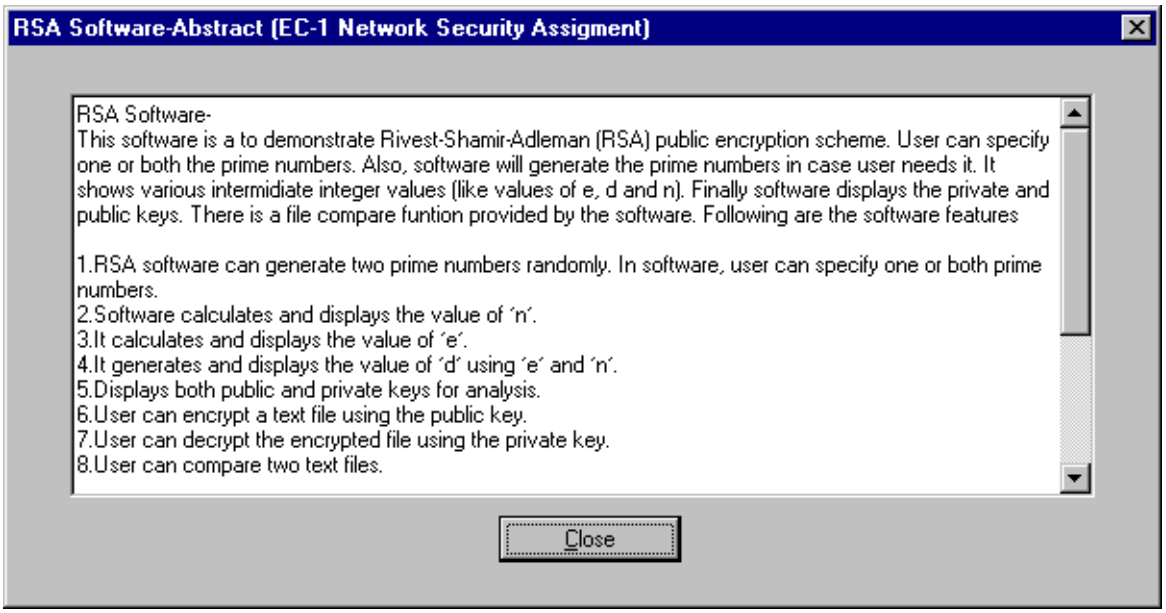
6.2 Screen Shot Showing Key Generation

In the example below prime numbers (7 and 17) are taken from the user.
The public key KU={5,119} and private key KR={77,119



6.3 Abstract Screen

This is the abstract screen snap-shot.



7 Appendix-B (Source Files)

This section contains the information of various source files used to develop the software. The software is written in VC++. Sources are categorized into source files (function definitions) and header files (function declarations) heads.

1. Under source files we have following files
    I. Abstract.cpp
    II. Decrypt.cpp
    III. EC1Assignment.cpp
    IV. EC1Assignment.rc
    V. EC1AssignmentDlg.cpp
    VI. Encrypt.cpp
    VII. GenPrimeNo.cpp
    VIII. RSA.cpp
    IX. Stdafx.cpp

2. Under header files we have the following files
    I. Abstract.h
    II. EC1Assignment.h
    III. EC1AssignmentDlg.h
    IV. Resource.h
    V. StdAfx.h