Analysis of the RSA Encryption Algorithm

Betty Huang

Computer Systems Lab 2009-2010

Abstract

The RSA encryption algorithm is commonly used in public security due to the asymmetric nature of the cipher. The procedure is deceptively simple, though; given two random (large) prime numbers p and q, of which n = pq, and message m, the encrypted text is defined as c = me (mod n). E is some number that is coprime to the totient(n). The public key (n, e), however, makes it difficult for the user to find the private key (n, d), due to the fact that given only n, it is extremely difficult to find the prime factors p and q. The fastest methods currently have O(sqrt(n)) complexity, but require expensive resources and technology (Kaliski). The aim of this paper is to improve on the factorization process required by the RSA encryption algorithm.

Introduction

The RSA algorithm is heavily based on mathematical concepts, utilizing Euler's totient function and prime factorization, and the modulo function. In general, there is a public key function defined as (n, e) and a private key function defined as (n, d). The algorithm statement is given c = m^e % n, where c is the ciphertext and m is the integer representation of the original message, m = c^d % n, for some d. First, note that the totient function returns the number of integers that are coprime to the input, is used to calculate d. Note that with any prime number p, the totient function returns p-1:

p = 5
totient(p) = 4

Since we are looking for n=pq, where p and q are prime numbers, the mathematical multiplication property allows us to calculate totient(n).
p = 5, q = 11
n = pq = 5*11 =55
totient(n) = (p-1)(q-1) = (5-1)(11-1) = 4(10) = 40

Now, we are interested in any positive number e that is coprime to totient(n). In this case, assume e = 3. Now, we want to find d such that de = 1 (mod totient (n)). Since we are guaranteed that a modular multiplicative inverse exists by specifying e such that e is coprime to totient n, we can find d by calculating the modular multiplicative inverse of e modulo totient(n):
3d = 1 (mod 40)
1 (mod 40) = 41, 81, 81+ 40 ...for simplicity, we use 81, because it divides evenly into 3
3d = 81
D = 9

Now, in this example, our public key is (55, 3), and our private key is (55, 9). Now, assume message m = 32 (1 < m < n). Then c= 32^3 % 55 = 43

We could find m by using the decrypt function (m = c^d % n): 43^9 % 55.

```
int totient(int X) // calculates how many numbers
   between 1 and N - 1 which are relatively prime to
   N.
{
       int i;
        phi = 1;
        for (i = 2 ; i < X ; ++i)
            if (gcd(i, X) == 1)
                ++phi;
          return phi;
}
```

Discussion

During the previous quarter, I experimented with coding a break of the RC5. At first, I worked on trying to
identify weak sections of the algorithm by studying the effects of simplifying the round and
rotation number. The first program written simply shifted through all the possible bit combinations
of RC5. This is the algorithm that Yin et al outlined in their paper. This quarter, I expanded Rivest's research into the RSA, which has greater practical applications while retaining a simpler structure.

Results

RESULTS
(message modification generated from CrypTool version 1.4.21)
Original Text:

```
00000  44 65 61 72 20 4D 72 20 53 68 6F 70 61 68 6F 6C 69 63   Dear Mr Shopaholic
00012  2C 0D 0A 0D 0A 70 6C 65 61 73 65 20 6F 72 64 65 72 20   ,....please order
00024  61 20 50 6F 72 73 63 68 65 20 61 6E 64 20 61 20 70 72   a Porsche and a pr
00036  65 70 61 69 64 20 69 6E 73 75 72 61 6E 63 65 20 73 63   epaid insurance sc
00048  68 65 6D 65 20 66 6F 72 20 4D 72 2E 20 44 6F 64 67 79   heme for Mr. Dodgy
0005A  2E 0D 0A 0D 0A 52 65 67 61 72 64 73 0D 0A 48 6F 6E 65   .....Regards..Hone
0006C  73 74 20 4A 6F 68 6E 0D 0A                              st John..
```

Name: SHA-1
Length in bit: 160
Algorithm ID: 30 21 30 09 06 05 2B 0E 03 02 1A 05 00 04 14

SHA-1 HASH - 2C 6B 70 15 B2 59 7A A6 43 44 43 80 08 B2 9B A9 8F EE 24 88

RSA KEY
Bit length of N: 304
RSA modulus N:
6429507761112837689643763499274122434437202712643424378894205954992714908919292778096
711251
phi(N) = (p-1)(q-1):
6429507761112837689643763499274122434437202707570891834526367855246908954559138973355
674192
Public key: 65537
Private key:
2587911946387468840732911749810825574835878514778103295893480898630669870642774722342
330737

ENCRYPTED HASH VALUE

Padding string: 01 00
Algorithm ID: 30 21 30 09 06 05 2B 0E 03 02 1A 05 00 04 14
Hash value: 2C 6B 70 15 B2 59 7A A6 43 44 43 80 08 B2 9B A9 8F EE 24 88

ASN-1 hash value: 01 00 30 21 30 09 06 05 2B 0E 03 02 1A 05 00 04 14 2C 6B 70 15 B2 59 7A A6 43 44 43 80 08 B2 9B A9 8F EE 24 88
Length in bit: 296

Encrypted hash value: 15 45 EF 40 D3 49 DB 69 48 6D 1B 2E F5 A4 EC D6 51 EC AC 99 10 F3 78 E2 CF 45 0C E4 74 3C 03 FD 30 BA 07 5D B8 02
Length in bit: 304

Result from program:

Enter N:
6429507761112837689643763499274122434437202712643424378894205954992714908919292778096
711251

Enter Message (in hexadecimal -- post message modification):
2C6B7015B2597AA64344438008B29BA98FEE2488

Public Key (304,65537)
Private Key
(204,258791194638746884073291174981082557483587851477810329589348089863066987064277472
2342330737)

Ciphertext:
1545EF40D349DB69486D1B2EF5A4ECD651ECAC9910F378E2CF450CE4743C03FD30BA075DB802
Time: 1.324 seconds