Modification on the Algorithm of RSA Cryptography System

By:

Assad Ibraheem Khyoon

Ms.c Degree in Electronic and Communication Engineering

Assist Instructor in Electronic Department

College of Engineering

Diala University

Assadkyoon@yahoo.com

Abstract. The protection of information for long period time is very critical in many environments. One way of implementation this protection with high activity is RSA cryptosystem. In this paper the old algorithm of RSA system is showed. And for more complexity and to increase time of attack coding. A new coefficient is z has been added to generator function Φ(n) in additional to selected p and q witch are depended in old algorithm.

Furthermore: A flowchart of this moderation is presented with corresponding Matlab program ,and for more illustration one mathematical example is employed here.

1-Introduction

There are numerous ways of speeding up RSA encryption and decryption in software and hardware implementations. Some of these techniques include fast modular multiplication, fast modular exponentiation and the use of the Chinese remainder theorem for faster decryption. Even with these improvements, RSA encryption/decryption is substantially slower than the commonly used symmetric-key encryption algorithm such as DES. In practice, RSA encryption is most commonly used for the transport of symmetric-key encryption algorithm keys and for the encryption of small items. The RSA cryptosystem has been patented in the U.S. and Canada. Several standards organizations have written, or are in the process of writing, standards that address the use of the RSA cryptosystem for encryption, digital signatures, and key establishment [1].

For understanding the work of RSA system, it is interest to show and study the following items:

1-1 Introduction to public-key cryptography

In the classical model of cryptography that we have been studying up to now, Alice and Bob secretly choose the key K. K then gives rise to an encryption rule e_K and a decryption rule d_K. In the cryptosystems we have seen so far, d_K is either the same as e_K, or easily derived from it (for example, DES decryption is identical to encryption, but the key schedule is reversed). Cryptosystems of this type are known as *private key* systems, since exposure of e_K renders the system insecure[2].

One drawback of a private-key system is that it requires the prior communication of the key K between Alice and Bob, using a secure channel, before any ciphertext is transmitted. In practice, this may be very difficult to achieve. For example, suppose Alice and Bob live faraway from each other and they decide that they want to communicate electronically, using e-mail. In a

situation such as this, Alice and Bob may not have access to a reasonable secure channel [3].

The idea behind *public-key* system is that it might be possible to find a cryptosystem where it is computationally infeasible to determine d_K given e_K. If so, then the encryption rule e_K could be made public by publishing it in a directory (hence the term public-key system). The advantage of a public-key system is that Alice (or anyone else) can send an encrypted message to Bob (without the prior communication of a secret key) by using the public encryption rule e_K. Bob will be the only person that can decrypt the ciphertext, using his secret decryption rule d_K [1].

The idea of a public-key system was due to Diffie and Hellman in 1976. They suggested the following: Each of $n \in \mathrm{N} \setminus \{1\}$ partner is having a public and a private key, K_m^p and K_m^s for $1 \le m \le n$. K_m^p determines the encryption e_{K_m} and K_m^s the decryption d_{K_m}. All keys K_m^p are published in a directory. The following properties must be satisfied for each $m, 1 \le m \le n$.:

1. $d_{K_m} * e_{K_m} = id$,

2. d_{K_m} and e_{K_m} can be computed in an efficient way,

3. K_m^s and d_{K_m} are secret and cannot be efficiently determined from K_m^p and e_{K_m}, respectively.

Suppose, participant *A* (Alice) sends message *m* encrypted to *B* (Bob) using the public key e_{K_B} of *B*, i.e.,

c = e_{K_B} (*m*)

B receives c and decrypts it to *m* using the private key K_B^s as follows:

$$d_{K_B}(c) \overset{(4.1)}{=} d_{K_B}(e_{K_B}(m)) = (d_{K_B} \circ e_{K_B})(m) \overset{1.}{=} m \ .$$

We can present this method graphically.

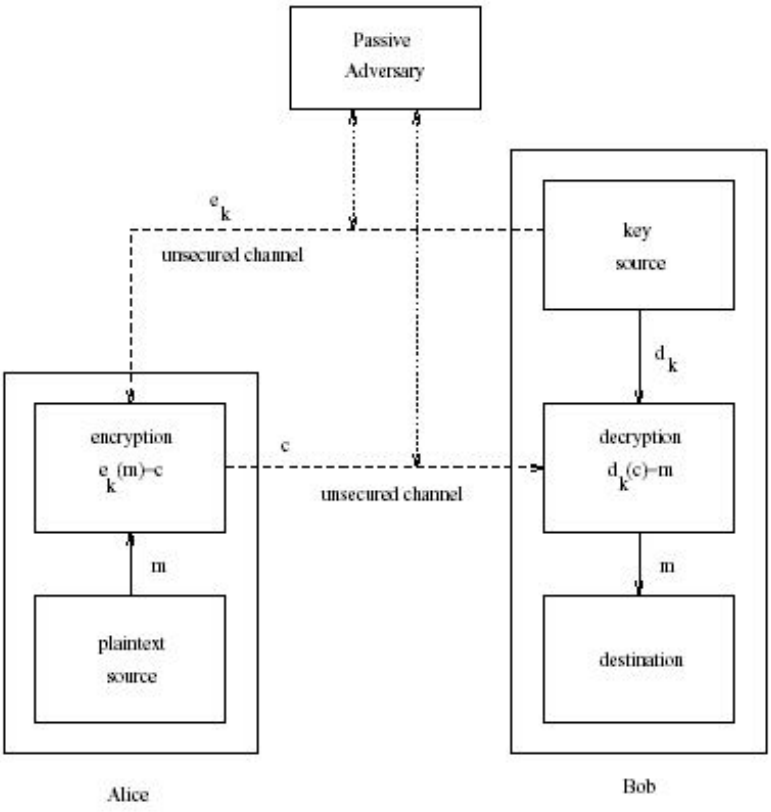


Fig.(1). Encryption using public-key techniques

Notice how Figure (1) differs from Figure (2) for a symmetric-key cipher. Here the encryption key is transmitted to Alice over an unsecured channel. This unsecured channel may be the same channel on which the ciphertext is being transmitted [4].

Since the encryption rule e_K need not be kept secret, it may be made public. Any entity can subsequently send encrypted messages to Bob which only Bob can decrypt. Figure (2) illustrates this idea, where A1;A2 and A3 are distinct entities. Note that if A1 destroys message *m1* after encrypting it to *c1*, then even A1 cannot recover *m1* from *c1*.

As a physical analogue, consider a metal box with the lid secured by a combination lock. The combination is known only to Bob. If the lock is left

open and made publicly available then anyone can place a message inside and lock the lid. Only Bob can retrieve the message. Even the entity which placed the message into the box is unable to retrieve it [5].

Public-key encryption, as decribed here, assumes that knowledge of the public key e_K does not allow computation of the private key d_K. In other words, this assumes the existence of trapdoor one-way functions which we will define now. Therefore, we give the definition of a one-way function at first [4].

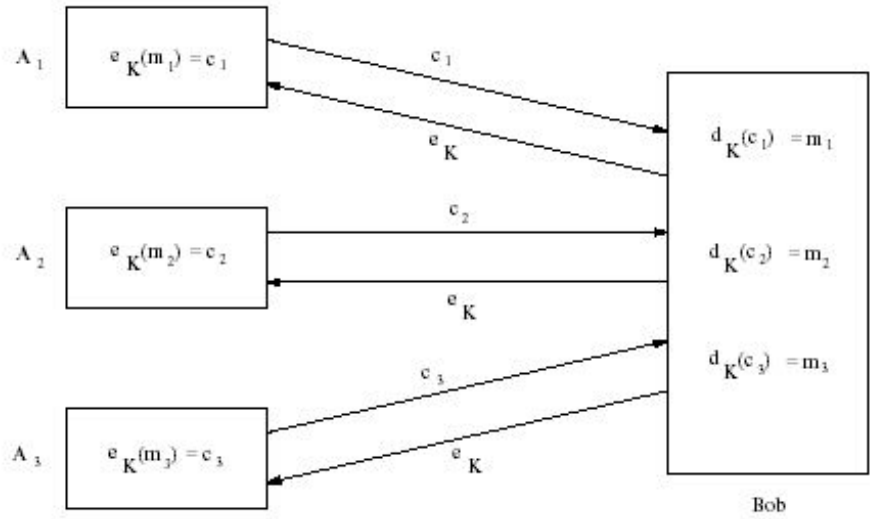


Fig.(2). Encryption using public-key techniques

<u>1-2 Mathematical background of RSA cryptosystem</u>.

We are now well prepared to introduce the RSA system and to understand why it works. So, here we go[1].

Definition (1). Let n = p - q, where p and q are distinct primes. Let P = C = Z_n, and

define

$K = \{(n, p,q, a,b) \mid n = pq, p, q \text{ prime}, ab \equiv 1 \pmod{j(n)}\}$:

۰

For $K = (n, p, q, a, b)$, define

$e_K(x) = x^b \bmod n$ and $d_K(y) = y^a \bmod n$

$(x,y \in Z_n)$. The values n and b are public, and the values p, q, a are secret.

This cryptosystem uses computations in Z_n, where n is the product of two distinct odd primes p and q. For such n, note that $j(n) = (p - 1)(q - 1)$. We will now verify that decryption works.

Since $ab \equiv 1 (\bmod\ j(n))$, there is an integer k such that $ab = 1 + k.\ j(n)$. Now, if $gcd(x, p) = 1$ then by Fermat's theorem.

$x^{p-1} \equiv 1 \pmod p$:

We compute

$$x^{ab} = x^{1+k.j(n)} = x^{1+k.(p-1).(q-1)} = x.(x^{p-1})^{k.(q-1)} \equiv x \pmod p$$

and therefore, $x^{ab} \equiv x \pmod p$. On the other hand, if $gcd(x, p) = p$, then this last congruence

is again valid since each side is congruent 0 modulo p. Hence, in all cases

$x^{ab} \equiv x \pmod p$.

By the same argument,

$x^{ab} \equiv x \pmod q$:

By definition, there must be two integers r and s such that $x^{ab}-x = p.r = q.s$. Since p and q are distinct and prime, p must be a factor of s due to the unique prime factorization (or q must be a factor of r). Again, there has to be an integer s' such that $x^{ab}-x = qs = qp\,s'$, i.e.

$n = p - q$ divides $x^{ab}- x$ or in other words

$x^{ab} \equiv x \pmod n$:

Therefore,

$d_K(e_K(x)) = d_K(x^b \bmod n) = (x^b \bmod n)^a \bmod n = x^{ab} \bmod n = x \bmod n$

which means that the RSA system works. Note, that also $e_K(d_K(x)) = x$ *in* Z_n, i.e., the RSA system can also be used to set up a signature scheme [6].

The security of RSA is based on the hope that the encryption function $e_K(x) = x^b \bmod n$ is one-way, so it will be computationally infeasible for an opponent to decrypt a ciphertext.

The trapdoor that allows Bob to decrypt is the knowledge of the factorization $n = p \cdot q$. Since

Bob knows this factorization, he can compute $j(n) = (p-1).(q-1)$ and then the decryption exponent a using Algorithm (1)[3].

Algorithm(1). Computing multiplicative inverses in Z_n

INPUT: $b \in Z_n$.

OUTPUT: $b^{-1} \bmod n$, provided it exists.

1. $n_0 \leftarrow n$, $b_0 \leftarrow b$, $t_0 \leftarrow 0$, $t \leftarrow 1$, $q \leftarrow [n_0/b_0]$, $r \leftarrow n_0 - q.b_0$

2. while $r > 0$ do

(a) $temp \leftarrow t_0 - q.t$

(b) If $temp \geq 0$ then $temp \leftarrow temp \bmod n$. Otherwise, $temp \leftarrow n - ((-temp) \bmod n)$.

(c) $t_0 \leftarrow t$, $t \leftarrow temp$, $n_0 \leftarrow b_0$, $b_0 \leftarrow r$, $q \leftarrow [n_0/b_0]$, $r \leftarrow n_0 - q.b_0$.

3. If $b_0 \neq 1$then b has no inverse modulo n. Otherwise, $b^{-1} \leftarrow t \bmod n$, return($b^{-1}$).

Algorithm (2). Key generation for RSA public-key encryption

SUMMARY: each entity creates an RSA public key and a corresponding private key.

Each entity A should do the following:

1. Generate two large random (and distinct) primes p and q, each roughly the same size.

2. Compute $n = p - q$ and $j(n) = (p-1) \cdot (q-1)$.

3. Select a random integer b, $1 < b < j(n)$ such that $gcd(b, j(n)) = 1$.

4. Use Algorithm (1) to compute the unique integer a, $1 < a < j(n)$, such that $ab \equiv 1 (mod\ j(n))$.

5. A's public key is *(n,b)*, A's private key is *a*.

The integers *a* and *b* are called the *encryption exponent* and the *decryption exponent*, respectively, while n is called the *modulus*.

Algorithm (3). RSA public-key encryption

SUMMARY: B encrypts a message m for A, which A decrypts.

1. *Encryption*. B should do the following:

(a) Obtain A's authentic public key *(n, b).*

(b) Represent the message as an integer *m* in the set {0; : : : ; n-1}.

(c) Compute $c = m^b \bmod n$.

(d) Send the ciphertext *c* to A.

2. *Decryption*. To recover plaintext *m* from *c*, A should do the following:

(a) Use the private key *a* to recover $m = c^a \bmod n$.

1-3 Attacks on RSA

It is important to mention some threats concerning with this tittle are [2]:

a. Relation to factoring
b. Small Encryption Exponent b
c. Small Decryption Exponent a
d. Forward Search Attack
e. Multiplicative Properties
f. Common modulus attack
g. Message Concealing
h. Cycling attacks

2-Presentation the modification of RSA algorithm

To ensure the proper complexity of RSA algorithm, a new coefficient is Z has been added to generator function φ(n) as follow:

φ(n)=(p-1)(q-1)(Z-1)

And then complete all steps of last section take Z in our consideration.

It is clear that the degree of eq. becomes grater by 1 about that similar in previous section, consequentially that effects on all equations of RSA algorithm and their attacks.

A flowchart below is obviously explain a new algorithm of RSA which is adopted here. .

And for make the process that dealt with the present algorithm is easy and benefit, A program by MATLAB language has been implemented here (see Appendix).

Example.

We do an RSA encryption with artificially small parameters.

Key generation. Entity A chooses the primes $p = 7, q = 11$ and $z=13$

and computes $n = p.q.z = 1001$ and $j(n) = (p-1)(q-1)(z-1)=720$. A chooses $b = 79$ and, using algorithm (1), find $a = 319$ such that $ab=1 \bmod j(n)$. A's public key pair is ($n = 1001, b = 79$), while A's private key is $a = 319$.

Encryption. To encrypt a message $m = 52$, B uses an algorithm for modular exponentiation 2 to compute

$c = m^b \bmod n = 52^{79} \bmod 1001 =$

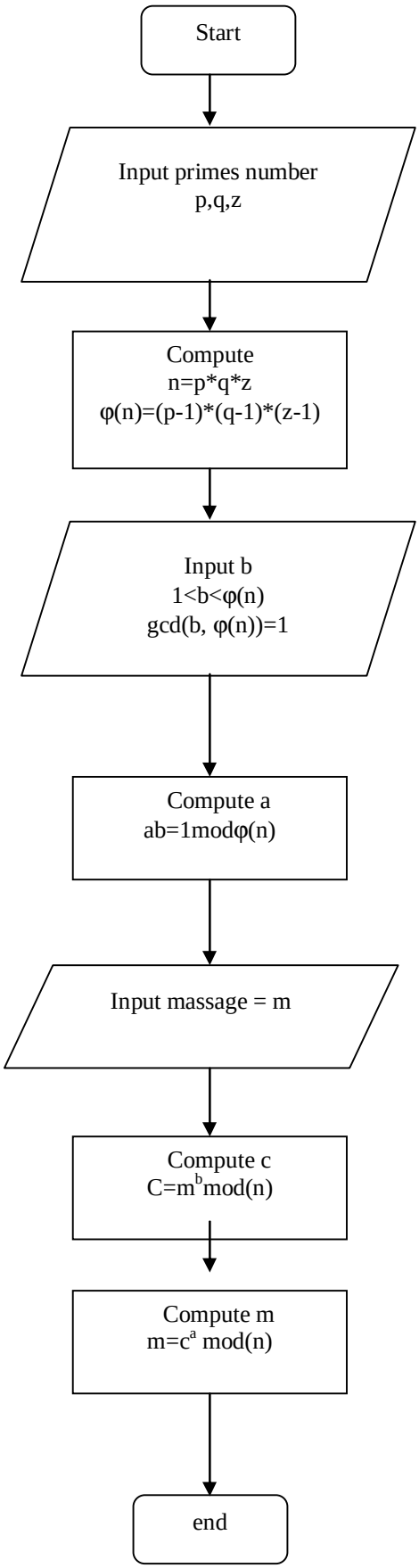
$=3.666605175738835391711333684436 1e+135 . \bmod 1001 = 869$

And sends this to A.

Decryption. To decrypt c, A computes

$m=c^a \bmod n = 689^{319} \bmod 1001$

$=2.4656520448335737886101385498388e+905 \bmod 1001=52.$


Start
Input primes number
p,q,z
Compute
n=p*q*z
φ(n)=(p-1)*(q-1)*(z-1)
Input b
1<b<φ(n)
gcd(b, φ(n))=1
Compute a
ab=1modφ(n)
Input massage = m
Compute c
C=m^b mod(n)
Compute m
m=c^a mod(n)
end

3-Discussion and evaluation the modification:

One concept which is absolutely fundamental to cryptography is that of a function in the mathematical sense .so it is interest to deal with generator function φ(n) and make it to be most difficult with three variables instead two in old RSA algorithm. The increasing degree of that function by one leads to change all concerning equations of cryptography and their attacks. it is clear that, the modification of algorithm presents a new strategy is different than the other of classical.

When we want to break a new algorithm during attacks process which related directly by factorial multiplication properties (see sec.1-3), we need to increase by one the number of probabilities than that of old RSA algorithm such that in old RSA system the attack takes factorial of two variables are p and q only, however at anew algorithm he takes factorial of three variables are p, q, and z in consideration. So we need to more time in sake of breaking the new algorithm.

Here, we also described what kind of security services the cryptographic algorithms provide for the RSA system. How the cryptographic algorithms are used in the system also determine what kind of attacks can be launched in practice to break the algorithms.

We don’t forget that complexity of the new algorithm is solved here through programming it by MATLAB language .and fortunately, results of program provide us with the efficient decryption algorithm we seek.

The modification certain the success of credit cards and all cards of phone and communication in commercial uses.

A more advanced, the modification resolve the problem of sharing secrete keys between entities (perhaps competing each-other) with in the manufacturing process. The issuer keeps under its control the screte data introduced in the cards, we avoid the usage of costly and unpractical.

Lastly, the modification does not give arise the military applications in consideration, perhaps it needs to some development.

4-Conclusion

The modification, which presented here provide us with high level of security, and it exhausts a far time to break the algorithm during attacks process.

So it may be consider a good academic method for understanding the work of cryptography system concerning with commercial applications.

The program of MATLAB language which is implemented here, solve the complexity of algorithm. That obviously proved by results.

Appendix

```
'%Generate two large random (and distinct) primes p,q and z,each roughly the same size%'
p=input('p=')
q=input('q=')
z=input('z=')
'%Compute n=p*q*z%'
n=p*q*z
'%Compute k=(p-1)*(q-1)*(z-1)%'
k=(p-1)*(q-1)*(z-1)
'%Selct a random integer b,1<b<k such that gcd(b,k)=1.%'
b=input('b=')
'%Comput the unique intger a,1<a<k,such that ab=1mod(k)%'
for a=1:k;
s=a*b;
d=mod(s,k);
if d==1
break
end
end
disp(a)
'%Encryption:To encrpt a message m%'
m=input('m=')
'%to compute encrypt amassege c%'
h=m.^b
c=mod(h,n)
'%DEcryption:To dencrpt c%'
f=c.^a
m=mod(f,n)
```

REFERENCES

[1] S.Babbage and L. Frisch. On MISTY1 Higher Order Di_erential Cryptanalysis. Proceedings
of ICISC 2000, Lecture Notes in Computer Science 2015, Springer-Verlag,
2000, 22{36.

[2] E. Barkan, E. Biham and N. Keller. Instant Ciphertext-Only Cryptanalysis of GSM
Encrypted Communication. Proceedings of Crypto 2003, Springer-Verlag, 2003.

[3] J. S. Kang, S. U. Shin, D. W. Hong, O. Y. Yi. Provable Security of KASUMI and
3GPP Encryption Mode f8. Advances in Cryptology - Asiacrypt 2001, Lecture Notes
in Computer Science 2248, Springer-Verlag, 2001.

[4] L.R. Knudsen and C.J. Mitchell. An analysis of the 3gpp-MAC scheme. In Daniel
Augot and Claude Carlet (Eds.) Workshop on Coding and Cryptography, WCC 2001,
Les Ecoles de Cotquidan, 2001, 319-328.

[5] L.R. Knudsen and D. Wagner. Integral Cryptanalysis. In J. Daemen and V. Rijmen
(Eds.) Fast Software Encryption - FSE 2002, Lecture Notes in Computer Science
2365, Springer-Verlag, 2002, 112{127.

[6] U. Kuhn. Cryptanalysis of Reduced-Round MISTY. In B. P_tzmann (Ed.) Advances
in Cryptology - Eurocrypt 2001, Lecture Notes in Computer Science 2045,
Springer-Verlag, 2001, 325{339.