A Comparitive Study of RSA based Cryptographic Algorithms

Ramzi A. Haraty*, A. N. El-Kassar**, Hadi Otrok*
* Lebanese American University P.O.Box 13-5053
Chouran, Beirut, Lebanon 1102 2801
** Beirut Arab University, Mathmatics Department,
Beirut, Lebanon

April 30, 2004

Abstract

In 1978 a powerful and practical public-key scheme Hadi Otrokwas produced by RSA; there work was applied using 2 large random odd primes p and q, each roughly of the same size. El-Kassar and Awad modified the RSA public-key encryption scheme from the domain of natural integers, Z, to two principal ideal domains, namely the domain of Gaussian integers, Z[i], and the domain of the rings of polynomials over finite fields, F[x], by extending the arithmetic needed for the modifications to these domains. In this work we implement the classical and modified RSA cryptosystem to compare and to test their functionality, realiability and security. To test the security of the algorithms we implement an attack algorithm to solve the integer factorization problem. After factorization is found, the RSA problem could be solved by computing the order $\phi(n)$, and then finding the private key using the extended Euclidean algorithm for integers.

1 Introduction

Cryptography is the art or science of keeping messages secret. People mean different things when they talk about cryptography. Children play with toy ciphers and secret languages.Strong encryption is the kind of encryption can be used to protect information of real value against organized criminals, multinational corporations, and major governments. Strong encryption used to be only in the military domain; however, in the information society it has become one of the central tools for maintaining privacy and confidentiality.

Perhaps the most striking development in the history of cryptography came in 1976 when Diffie and Hellman published New Directions in Cryptography [3]. Their work introduced the concept of public-key cryptography and provided a new method for key exchange. Although the authors had no practical realization of a public-key encryption scheme at the time, the idea was clear and it generated extensive interests and activities in the world of cryptography. One of the powerful and practical public-key schemes was produced by Rivest-Shamir-Adleman (RSA) in 1978 [8].

El-Kassar and Awad [1] modified the RSA public-key encryption schemes from the domain of natural integers, Z, to two principal ideal domains, namely the domain of Gaussian integers, Z[i], and the domain of the rings of polynomials over finite fields, F[x], by extending the arithmetic needed for the modifications to these domains.

In this paper, we compare and evaluate the classical and modified RSA algorithms. We investigate the issues of complexity, efficiency and reliability by running the programs with different sets of data. The attack algorithm consists of subroutines used to crack encrypted messages. This is done by applying certain mathematical concepts to find the private key of the encrypted message. After finding the key, it will be easy to decrypt the message. A study will be done using the results of running the attack algorithm to compare the security of the different classical and modified cryptographic algorithms.

The rest of the paper is organized as follows: section 2 describes the classical technique of RSA cryptosystem. Then, we present the modifications done on RSA encryption scheme. In section 3, we deal with the attack algorithm. In section 4, a testing procedure is used to evaluate the classical and modified algorithms. Also, attack programs is run to test the complexity, efficiency and reliability of the different modified algorithms and compare them to the classical one. A conclusion is drawn in section 5.

2 Classical and Modified RSA Public-Key Cryptosystems

2.1 Classical RSA Encryption Scheme

In RSA public-key encryption scheme, entity A generates the public-key by first generating two large random odd primes p and q, each roughly of the same size, and computing the modulus $n = pq$ and the order $\phi(n) = (p - 1)(q - 1)$ [6]. Then, entity A selects the encryption exponent e to be any random integer in the interval $(1, \phi(n))$ relatively prime to $\phi(n)$. Using the extended Euclidean algorithm for integers, entity A finds the decryption exponent d which is the unique integer $(1; \phi(n))$ relatively prime to $\phi(n)$ such that $ed = 1$ in Z_n, i.e., d is the unique inverse of e in Z_n. Hence, the public-key is the pair $(n; e)$; and A's private-key is the triplet

$$(p; q; d):$$

To encrypt the message m in the complete residue system modulo n, Z_n, entity B first obtains A's public-key (n, e). Then, it computes the ciphertext $c \in Z_n$ such that

$$c \equiv m^e (\text{mod}\, n)$$

and sends it to entity A.

Now, to encrypt c, i.e. to recover the plaintext m from the sent ciphertext c, entity A uses the private-key d to compute

$$m = c^d (\text{mod}\, n)$$

which is the original message.

In order to be sure of the strengthness of the above scheme, we should mathematically prove that the real message m should be recovered by decrypting c using the decryption algorithm.

Algorithm 1 RSA Public Key Cryptography:

1. Find two large primes p and q and compute their product $n = pq$.
2. Find an integer d that is co-prime to $(p - 1)(q - 1)$.
3. Compute e from $ed = 1 \text{mod} (p - 1)(q - 1)$.
4. Broadcast the public key, that is, the pair of numbers $(e; n)$.
5. Represent the message to be transmitted, a, say as a sequence of integers $\{a\}$ each in the range 1 to n.
6. Encrypt each message a using the public key by applying the rule $c \equiv m^e (\text{mod}\, n)$.
7. The receiver decrypts the message using the rule $m = c^d (\text{mod}\, n)$.

2.2 RSA Cryptosystem in the Domain of Gaussian Integers, Z[i]

In RSA public-key scheme, entity A generates the public-key by first generating two large random Gaussian primes β and γ, each roughly the same size [5]. Sencondly, entity A computes $\eta = \beta\gamma$ and $\phi(\eta) = \phi(\beta)\phi(\gamma) = (\beta^2 - 1)(\gamma^2 - 1)$. It selects a random integer e such that $1 < e < \phi(\eta)$ and e is relatively prime to $\phi(\eta)$. Then, entity A finds the unique integer d such that $1 < d < \phi(\eta)$ and d is relatively prime to $\phi(\eta)$. A's public-key is

$$(\eta; e)$$

and A's private-key is

$$(¯; °; d).$$

To encrypt the message m chosen from the complete residue system modulo ´, G´, entity B first obtains A's public-key (´; e) then computes the ciphertext c ´ m^e(mod ´) and send it to entity A. To decrypt the ciphertext c sent by B, A uses the private-key d to recover the plaintext m ´ c^d(mod ´).

The following provides three algorithms for the description of the RSA public-key encryption scheme over the domain of Gaussian integers.

Algorithm 2 (Key Generation for the RSA Gaussian Public-Key Encryption)

The following algorithm shows how entity A creates an RSA Gaussian public-key and a corresponding private-key. Entity A should do the following:

1. Generate two distinct large random Gaussian primes ¯ and °, each roughly the same size.

2. Compute ´ = ¯° and Á(´).

3. Select a random integer e such that 1 < e < Á(´) and (e; Á(´)) = 1.

4. Use the extended Euclidean algorithm over the domain of Gaussian integers to compute the unique inverse d of e such that ed ´ 1(mod Á(´)).

5. A^0s public-key is (´, e).

6. A^0s private-key is (¯, °, d).

Algorithm 3 (RSA Gaussian Public-Key Encryption)

The following algorithm shows how entity B encrypts the message m in the complete residue system modulo a Gaussian integer ´. Entity B should do the following:

1. Obtain A's authentic public-key (´, e).

2. Represent the message as an integer m in the complete residue system modulo the Gaussian integer ´, G´.

3. Compute c = m^e(mod ´).

4. Send the ciphertext c to A.

Algorithm 4 (RSA Gaussian Public-Key Decryption)

The following algorithm shows how entity A recovers the real message m from the ciphertext c. Entity A should do the following:

1. Receive the ciphertext c.

2. Use the private-key d to recover m ´ c^d(mod ´).

2.3 RSA Cryptosystem over Quotient Rings of Polynomials over Finite Fields

Given a prime number p and a polynomial f(x) of degree n in the finite field $Z_p[x]$ as a product of two distinct irreducible polynomials in $Z_p[x]$, that is f(x) = h(x)g(x), where h(x) is of degree s and g(x) is of degree r. The ring $Z_p[x]$=hf(x)i is finite of order p^n and from chapter two we have that,

$$Z_p[x]=hf(x)i \cong \frac{Z_p[x]}{hf(x)i} © \frac{Z_p[x]}{hg(x)i}.$$

Also, we obtain that,

$$U(Z_p[x]=hf(x)i \cong U \; μ \frac{Z_p[x]}{hf(x)i} ¶ \; £ \; μ \frac{Z_p[x]}{hg(x)i} ¶ \; .$$

Since each of h(x) and g(x) is irreducible, the quotient rings $\frac{Z_p[x]}{hh(x)i}$ and $\frac{Z_p[x]}{hg(x)i}$ is a finite field of order p^s and p^r respectively. Also, the groups of units $\frac{Z^¤_p[x]}{hh(x)i}$ and $\frac{Z^¤_p[x]}{hg(x)i}$ are cyclic and of order Á(h(x)) = p^s ¡ 1 and Á(g(x)) = p^r ¡ 1 respectively [4].

Now, given a positive integer e such that (e; Á(f(x))) = 1 and a polynomial m(x), find a polynomial c(x) such that c(x) ´ $m(x)^e$(mod f(x)) in $Z_p[x]$. The polynomials h(x) and g(x) should be selected so that factoring f(x) = h(x)g(x) is computationally infeasible.

The following provides three algorithms for the description of the RSA public-key encryption scheme over polynomials.

Algorithm 5 (Key Generation for the RSA Public-Key Encryption over Polynomials)

The following algorithm shows how entity A creates an RSA public-key and a corresponding private-key. Entity A should do the following:

1. Generate a random odd prime integer p.
2. Generate two irreducible polynomial h(x) and g(x) in $Z_p[x]$.
3. Reduce the polynomial $f(x) = h(x)g(x)$ in $Z_p[x]$. Then compute the order of $Z_p^*[x]/\langle f(x)\rangle$ which is equal to $\phi(f(x)) = (p^s - 1)(p^r - 1)$.
4. Select a random integer e such that $1 < e < \phi(f(x))$ and $(e; \phi(f(x))) = 1$.
5. Use the Euclidean algorithm for integers to find the unique inverse d of e with respect to $\phi(f(x))$ such that $1 < d < \phi(f(x))$ and $e.d \equiv 1 (\bmod \phi(f(x)))$ in $Z_p[x]$.
6. A's public-key is (p, f(x), e), A's private-key is (p, d, g(x), h(x)).

Note that e and d should be chosen to be integers since they will be used as powers as we will see next.

Algorithm 6 (RSA public-key Encryption over Polynomials)

The following algorithm shows how entity B encrypts a message m(x) for A. Entity B should do the following:

1. Receive A's authentic public-key (p, f(x), e).
2. Represent the message as a polynomial m(x) in the complete residue system modulo f(x) in $Z_p[x]$.
3. Compute the polynomial $c(x) \equiv m(x)^e (\bmod f(x))$ in $Z_p[x]$.
4. Send the ciphertext c(x) to A.

Algorithm 7 (RSA Public-Key Decryption over Polynomials)

The following algorithm shows how entity A decrypts the sent ciphertext c(x) and recover the real message m(x). Entity A should do the following:

1. Receive the ciphertext c(x) from B.
2. Use the private-key d to recover m(x) by reducing $c(x)^d (\bmod f(x))$ in $Z_p[x]$.

3 RSA Public-Key Scheme Attack

In order to attack any protocol that uses the RSA public key encryption scheme, we should first solve the factorization problem in order to find the private key. Recall that the intractability of both the integer factorization problem and the RSA problem forms the basis for the security of the RSA public-key encryption scheme.

Hence, to attack any protocol that uses the RSA public-key encryption scheme, we should first solve the integer factorization problem as described in RSA public-key encryption attack. After factorization of integer n, the RSA problem could be solved by computing the order $\phi(n)$, and then finding the integer d using the extended Euclidean algorithm for integers. Once d is found, the adversary can decrypt any ciphertext intended for A. On the other hand, if an adversary could somehow compute d, then he/she could subsequently factor n efficiently as follows: First note that since $ed \equiv 1 (\bmod \phi(n))$, there is an integer k such that $ed - 1 = k\phi(n)$. Hence, by the fact that $a^{ed-1} \equiv 1 (\bmod n)$ for all $a \in Z_n^*$ (Eulers theorem). Let $ed - 1 = 2^s t$, where t is an odd integer. Then it can be shown that $a^{2^{s-1}t}$ is not congruent to either ± 1 modulo n for at least half of all integers $a \in Z_n^*$. If a is such an integer, then $\gcd(a^{2^{s-1}t} - 1; n)$ is a non-trivial factor of n.

The above discussion shows that the RSA problem and the integer factorization problem are computationally equivalent.

3.0.1 Example:

How to find the private key in the case of RSA?

Solution: Assume that the public key is:

(n= 8038438974502939,e= 180977512554819). We have to find the private key. Since n is the multiplication of p and q (where p and q are both primes), so we have to find p and q using Mathmatica.

Divisors[8038438974502939]

{1, 89657297, 89657387, 8038438974502939}

p=89657297 and q=89657387

Now, we have to find Á(n)= (p-1)(q-1)=(89657297-1)(89657387-1)=8038438795188256.

After, finding Á(n) we have to find d which is the inverse of e using Mathmatica.

PowerMod[180977512554819, -1, 8038438795188256] = 2500998620710731=d

Since d is found the private key will be (89657297,89657387,2500998620710731).

4 Testing and Evaluation

In this section, we compare and evaluate the different classical and modified cryptosystems by showing the implementation of the cryptosystems' algorithms with their running results. Also, we test the security of the algorithms by implementing different attack algorithms to crack the encrypted messages. All this is done using Mathematica 4.0 as a programming language and a PIV Dell computer with 2.4 GHZ CPU, 40 GByte hard-disk, and 512 MB DDRAM.

4.1 RSA based Algorithms

Using Mathmatica 4.0 functions and an additional abstract algebra library, we have written programs for the following algorithms:

1. Classical RSA.
2. RSA with Gaussian numbers.
3. RSA with irreducible polynomials.

After running the programs, it was clear that these programs have applied the RSA cryptosystem in the correct way. All the programs have generated a public and private key with different mathematical concepts. Then a message is encrypted using the encryption scheme and is sent encrypted to a decryption procedure which returned the original message.

Comparing these algorithms with each other, we conclude the following:

1. All programs are reliable; they can encrypt and decrypt any message.

2. The complexity for the three programs is the complexity of finding the inverse of a number with respect to ©(n). Thus, using the classical and Gaussian algorithms, it is easy to find the inverse of a number and it does not take much time. However, in the case of irreducible polynomial algorithm, we implemented a special sub routine to find the inverse of a polynomial. Although this subroutine helped us to apply the irreducible polynomial algorithm, but it is inefficient because it needs a lot of time specially when p is very large and when the polynomial order increases.

3. We faced a problem during the execution of the irreducible polynomial algorithms. This is due to the difficulty of generating a random irreducible polynomial according to a prime number p.

4.2 Attack Algorithm

In order to attack any protocol that uses the RSA public key encryption scheme, we should first solve the factorization problem in order to find the private key. To test the security of the algorithms, we implemented attack schemes and applied them on the classical and modified cryptosystem algorithms. After running these attack algorithms, we observed the following:

1. All the attack programs are reliable so that they can hack an encrypted message by finding the private key.

2. Attacking the RSA algorithm using the extended Euclidean algorithm for integers is easy if we are dealing with small prime numbers. However, when it comes to 100-digit prime numbers or higher, it needs about many computers working in parallel processing to compute the prime factorization of the multiplication of two 100-digit prime numbers.

3. Attacking the RSA polynomial algorithm is much more secure than the classical one since ©(n) will be in the form (p^t-1) (p^r-1). Where t and r are the power of the two irreducible polynomials.

5 Conclusion

In this work, we presented the classic RSA cryptosystem and two modifications to it, namely, the RSA cryptosystem in Zn, in the domain of Gaussian integers, Z[i] and over quotient rings of polynomials over finite fields. We implemented these algorithms and tested their efficiency, reliability, and security. The results obtained showed that all the algorithms applied the RSA cryptosystem correctly and generated public and private key using different mathematical concepts. Messages were then encrypted using the encryption scheme and were sent in encrypted form to a decryption procedure which returned the original messages.

We also built attack scenarios directly aimed at solving the factorization problem. We modified the RSA attack algorithm to handle the modified algorithms. We observed that the polynomial domain algorithm was the most challenging to attack due to mathematical complexity.

As for future work, we plan to compare and evaluate the efficiency of the modified algorithms using very large numbers by using parallel computing techniques. We plan to run the programs in parallel on many computers and split the complex mathematical calculations between these computers. We plan to write a function that is capable of finding any random irreducible equation with respect to a specific prime number p. We also plan to apply the modified algorithms in many fields such as database, communications and network security.

6 REFERENCES

[1] Awad, Y. A. Applications of Number Theory to Cryptography, M.S. Thesis, Beirut Arab University, 2002.

[2] Cross, J. T. The Euler's f ¡ function in the Gaussian Integers, American Mathematics Monthly 90, pp. 518-528, 1983.

[3] Diffie, W. and Hellman, M. E. New Directions in Cryptography, IEEE Transaction on Information Theory, IT-22, pp. 472-492, 1978.

[4] El-Kassar, A. N., Chihadi H., and Zentout D. Quotient rings of polynomials over finite fields with cyclic group of units, Proceedings of the International Conference on Research Trends in Science and Technology, pp. 257-266, 2002.

[5] Kenneth, A. R. Elementary Number Theory and its Applications, AT&T Bell Laboratories in Murray Hill, New Jersey, 1988.

[6] Menezes, A. J., Van Oorshot, and Vanstone, P. C. S. A. Handbook of Applied Cryptography, CRC press, 1997.

[7] Otrok, H. Security Testing and Evaluation of Cryptographic Algorithms, M.S. Thesis, Lebanese American University, June 2003.

[8] R. Rivest, A. Shamir and L. Aldeman, A Method for Obtaining Digital Signatures and PublicKey Cryptosystems, Communications of the ACM 21, pp. 120-126, 1978.

[9] Smith J. L. and Gallian, J. A. Factoring Finite Factor Rings, Mathematics Magazine 58: pp. 93-95, 1985.

This work is funded by Lebanese American University