A comparative study of RSA-based cryptographic algorithms

Ramzi A. Haraty
A. N. El-Kassar
Hadi Otrok

• RSA cryptosystem algorithms
• testing and evaluation

In 1978 the powerful and practical RSA public-key scheme was produced. It is the most widely used public-key cryptosystem. Its security is based on the intractability of the integer factorization problem and on solving the RSA problem of finding the e-th root of an integer c modulo n, where n is the product of large distinct primes. El-Kassar *at al.*, 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, $\mathrm{Z}[i]$, 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, reliability and security. To test the security of the algorithms we implement attack algorithms to solve the factorization problem in Z, $\mathrm{Z}[i]$, and $F[x]$. After factorization is found, the RSA problem could be solved by finding the private key using the extended Euclidean algorithm.

1 Introduction

Cryptography maybe defined as the art or science of keeping messages secret. Encryption can be used to protect information of real value. In the information society, encryption has become one of the central tools for maintaining privacy, confidentiality, trust, access control, electronic payments, corporate security, and other fields. In 1976, one of the most striking development in the history of cryptography came when Diffie and Hellman published *New Directions in Cryptography* [2]. The concept of public-key cryptography was introduced and generated extensive interests and activities in the world of cryptography. In 1978, Rivest-Shamir-Adleman (RSA) produced one of the powerful and practical public-key schemes, see [12]. The RSA public-key cryptosystem scheme is the most popular and widely used public-key cryptosystem. Its security is based on the intractability of both the integer factorization problem and the RSA problem. The RSA problem, see [11], is the problem of finding an integer m such that $m^e \equiv c \pmod{n}$, where n is a product of two distinct large odd primes p and q, e is a positive random integer such that $\gcd(e,(p-1)(q-1))=1$, and c is any integer. That is, the RSA problem is that of finding e-th roots of an integer c modulo a composite integer n.

The classical cryptosystems, such as RSA, ElGamal and Rabin cryptosystems, are described in the settings of the domain of integers Z. Many aspects of arithmetic over the domain of integers can be carried out to the domain of Gaussian integers $\mathrm{Z}[i]$, the set of all complex numbers of the form $a+bi$, where a and b are integers, and to the domain of polynomials over finite fields $\mathrm{F}[x]$. Recently, the classical cryptosystems were modified in many directions in these domains. El-Kassar *et al.* [7] modified the ElGamal public-key encryption schemes from its classical settings of the domain of natural integers to the domain of Gaussian integers by extending the arithmetic needed for the modifications to the domains. Similar extensions to the domain $\mathrm{F}[x]$ was given by El-Kassar and Haraty in [3] and [4]. Haraty *et al.* [8] and [9] gave a comparative study of the extended ElGamal cryptographic algorithms. A modification of Rabin cryptosystem to polynomial rings over finite fields was given in [6]. In [5], two extensions of the RSA cryptosystem in the domain of Gaussian and the domain of polynomials over finite fields were presented. It was pointed out that the extended algorithms require a little additional computational effort than the classical one and accomplish much greater security.

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. Moreover, these different algorithms will be compared. In addition, implementation of an attack algorithm will be presented. 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 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, the attack programs are conducted 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

The classical and modified RSA cryptosystems are described in this section. Algorithms and examples are given. These algorithms will be implemented to evaluate and compare the various methods in section 5.

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 Euler phi-function $\phi(n)=(p-1)(q-1)$, see [11]. Entity A then selects the encryption exponent e to be any random integer in the interval $(1, \phi(n))$ such that $\gcd(e, \phi(n))=1$. 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 $ed \equiv 1 \pmod{\phi(n)}$. 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 Z_n, the complete residue system modulo n, entity B uses A's public-key (n, e) to compute the ciphertext $c \in Z_n$ given by

$$c \equiv m^e \pmod{n}$$

and sends it to entity A.

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

$$m \equiv c^d \pmod{n}$$

which is the original message.

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 relatively prime to $\phi(n)=(p-1)(q-1)$.
3. Compute e from $ed \equiv 1 \pmod{\phi(n)}$.
4. Broadcast the public key (n, e).
5. Represent the message m to be transmitted by a sequence of integers in the range 1 to n.
6. Encrypt the message m using the public-key by applying the rule $c \equiv m^e \pmod{n}$.
7. The receiver decrypts the message using the rule $m \equiv c^d \pmod{n}$.

Example 1.
In order to generate the public-key, entity A selects the primes $p=89657387$ and $q=89657297$ and then computes the modulus $n=pq=8038438974502939$ and the Euler phi-function $\phi(n)=(p-1)(q-1)=8038438795188256$. Next, A selects the encryption exponent $e=180977512554819$ and uses the extended Euclidean algorithm for integers to find the de-

cryption exponent $d=2500998620710731$ so that $ed \equiv 1 \pmod{\phi(n)}$. Now, the public-key is the pair

$$(n=8038438974502939,\ e=180977512554819)$$

and A's private-key is the triplet

$$(p=89657387,\ q=89657297,\ d=2500998620710731)$$

To encrypt the message $m=1101100100111$ in the complete residue system modulo $n=8038438974502938$, entity B computes the ciphertext $c=1101100100111^{180977512554819}$ in $Z_{8038438974502938}$ to get $c=1419747618737142$ and sends it to A. To decrypt the sent ciphertext $c=1419747618737142$, A computes $m=1419747618737142^{2500998620710731}$ in $Z_{8038438974502938}$ to get $m=1101100100111$ which is the original message m.

Note that since $m \in \{0,1,2,...,8038438974502938\}$, there are exactly 8038438974502939 values in the message space for m to be represent by. Also, there are $\phi(n)=(p-1)(q-1)=8038438795188256$ values for the encryption exponent to be selected from or for the decryption exponent to be hidden in.

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, γ and computes $\eta=\beta\gamma$. The Gaussian primes of $Z[i]$ up to multiples of ± 1 and $\pm i$, see [10], are of the form:

i) $\alpha=1+i$;
ii) $\pi=a+bi$ and ???\overline$\pi=a-bi$, where $q=\pi\cdot$???\overline$\pi=a^2+b^2$ is an odd prime integer of the form $4k+1$;
iii) p, where p is an odd prime integer of the form $4k+3$.

If β and γ are selected to be of the form π_1 and π_2, then the modified scheme is equivalent to the classical one, see [5]. If $\beta$and γ are selected to be of the form π and p, then η can be easily factored. Hence, β and γ are selected to be odd integers of the form $4k+3$. Next, entity A computes $\phi(\eta)=\phi(\beta)\phi(\gamma)=(\beta^2-1)(\gamma^2-1)$, where $\phi(\eta)$ is Euler phi-function in $Z[i]$, see [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 $\phi(\eta)$ such that $ed \equiv 1 \pmod{\phi(\eta)}$. A's public-key is

$$(\eta,\ e)$$

and A's private-key is

$$(\beta,\ \gamma,\ d).$$

To encrypt the message m chosen from the complete residue system modulo η, $G_\eta=\{a+bi \mid 0 \le a \le \beta\gamma-1, 0 \le b \le \beta\gamma-1\}$, entity B first obtains A's public-key (η, e). Then B computes the ciphertext $c \equiv m^e \pmod{\eta}$ and sends it to entity A. To decrypt the ciphertext c sent by B, A uses the private-key d to recover the plaintext $m \equiv c^d \pmod{\eta}$. We note that the message space is enlarged so that its order is the square of that of the classical case. Also, the range for the ciphering exponent e is enlarged by more than the square of that of the classical case.

In the following we provide three algorithms describing 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 β and γ, each being an odd integer of the form $4k+3$.

2. Compute $\eta=\beta\gamma$ and $\phi(\eta)=(\beta^2-1)(\gamma^2-1)$.
3. Select a random integer e in $(1, \phi(\eta))$ and relatively prime to $\phi(\eta)$.
4. Use the extended Euclidean algorithm for integers to compute the unique inverse d of e in $(1, \phi(\eta))$ such that $ed\equiv 1 \pmod{\phi(\eta)}$.
5. A's public-key is (η, e).
6. A's private-key is (β, γ, 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 η. Entity B should do the following:

1. Obtain A's authentic public-key (η, e).
2. Represent the message as an integer m in G_η, the complete residue system modulo the Gaussian integer η.
3. Compute $c\equiv m^e \pmod{\eta}$.
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 $m\equiv c^d \pmod{\eta}$.

Example 2. **RSA Gaussian encryption with small parameters**

Public-Key Generation: Let $\beta=27743$ and $\gamma=23291$ be two Gaussian primes of the form $4k+3$. Compute the product $\eta=\beta\gamma=646162213$ and $\phi(\eta)=(27743^2-1)(23291^2-1)=417525604196912640$. Entity A chooses $e=151588853712295213$ such that $\gcd(e, \phi(\eta))=1$ and $1<e<\phi(\eta)$. Using the extended Euclidean algorithm for integers, A finds $d=24293572299784357$ such that $ed\equiv 1 \pmod{\phi(\eta)}$. Hence, A's public-key is the pair

$$(\eta=646162213,\ e=151588853712295213)$$

and A's private-key is the triplet

$$(\beta=27743,\ \gamma=23291,\ d=24293572299784357).$$

Public-Key Encryption: To encrypt the message $m=6461622+3i$, entity B computes the Gaussian integer

$$c=m^e=(6461622+3i)^{151588853712295213}\equiv 495038485+372009420i \pmod{646162213}$$

and sends it to entity A.

Public-Key Decryption: To decrypt the ciphertext $c=495038485+372009420i$, A computes the Gaussian integer

$$c^d=(495038485+372009420i)^{24293572299784357}\equiv 6461622+3i \pmod{646162213}$$

which is the original message m.

The following are some of the advantages of the RSA scheme in $Z[i]$. First, generating two primes p and q in the form $4k+3$ in both the classical and the modified methods requires the same amount of effort. Second, the modified method provides more security than the classical method since the number of elements that can be chosen from to represent the message m and the number of elements that the ciphering exponent can be selected are about the square of those used in the classical case. The computations involved in the modified method do not require computational procedures that are different from those of the classical method.

RSA cryptosystem over quotient rings of polynomials over finite fields

Let p be a prime number and let $h(x)$ and $g(x)$ be two distinct\ irreducible polynomials in $Z_p[x]$, the domain of polynomials over the finite field Z_p, where $h(x)$ is of degree s and $g(x)$ is of degree r. Let $f(x)=h(x)g(x)$. The polynomials $h(x)$ and $g(x)$ should be selected so that factoring $f(x)=h(x)g(x)$ is computationally infeasible. The quotient ring $Z_p[x]/\langle f(x)\rangle$ is finite of order p^n, where $n=r+s$ is the degree of $f(x)$. It is well known that the quotient ring $Z_p[x]/\langle f(x)\rangle$ is the direct sum of $Z_p[x]/\langle g(x)\rangle$ and $Z_p[x]/\langle h(x)\rangle$, that is

$$Z_p[x]/\langle f(x)\rangle \cong Z_p[x]/\langle g(x)\rangle \otimes Z_p[x]/\langle h(x)\rangle.$$

Its group of units $U(Z_p[x]/\langle f(x)\rangle)$ is the direct product of groups of units $U(Z_p[x]/\langle g(x)\rangle)$ and $U(Z_p[x]/\langle h(x)\rangle)$, that is

$$U(Z_p[x]/\langle f(x)\rangle) \cong U(Z_p[x]/\langle g(x)\rangle) \times U(Z_p[x]/\langle h(x)\rangle).$$

Since $h(x)$ and $g(x)$ are irreducible, the quotient rings $Z_p[x]/\langle h(x)\rangle$ and $Z_p[x]/\langle g(x)\rangle$ are finite fields of order p^s and p^r, respectively. Hence, the groups $U(Z_p[x]/\langle g(x)\rangle)$ and $U(Z_p[x]/\langle h(x)\rangle)$ are cyclic with orders $\phi(h(x))=p^s-1$ and $\phi(g(x))=p^r-1$, respectively, so that $\phi(f(x))=(p^s-1)(p^r-1)$. Now, given a message, represented by a polynomial $m(x)$ in $Z_p[x]$, and positive integer e such that $\gcd(e, \phi(f(x)))=1$ the message can be encrypted by a polynomial $c(x)$ such that $c(x)\equiv m(x)^e \pmod{f(x)}$.

In the following we provide three algorithms for describing the RSA public-key encryption scheme in a polynomial ring over a finite fields.

Algorithm 5. Key generation for the RSA public-key encryption using 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 $Z_p[x]$.
4. Compute $\phi(f(x))=(p^s-1)(p^r-1)$.
5. Randomly select e in $(1, \phi(f(x)))$ with $\gcd(e, \phi(f(x)))=1$.
6. Use the Euclidean algorithm for integers to find the unique inverse d of e in $(1, \phi(f(x)))$ with $ed\equiv 1 \pmod{\phi(f(x))}$.
7. A's public-key is $(p, f(x), e)$, A's private-key is $(d, g(x), h(x))$.

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

Algorithm 6. RSA public-key encryption using 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 $Z_p[x]$, the complete residue system modulo $f(x)$.
3. Compute the polynomial $c(x)\equiv m(x)^e \pmod{f(x)}$ in $Z_p[x]$.
4. Send the ciphertext $c(x)$ to A.

Algorithm 7. RSA public-key decryption using polynomials

The following algorithm shows how entity A decrypts the sent ciphertext $c(x)$ and recovers 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 $c(x)^d \pmod{f(x)}$ in $Z_p[x]$.

This is an example illustrating the RSA scheme using polynomials.

Example 3. RSA encryption over polynomials with small parameters

Public-Key Generation: Let $p=101$. Entity A chooses the two irreducible polynomials $h(x)=18x^2+71x+88$ and $g(x)=28x^3+83x^2+3x+95$ in $Z_{101}[x]$. Reducing the polynomial

$f(x)=h(x)g(x)$ in $Z_{101}[x]$ and computing $\phi(f(x))$, A gets $f(x)=100x^5+48x^4+28x^3+36x^2+40x+78$ and $\phi(f(x))=(101^3-1)(100^2-1)=10509060000$. Entity A then chooses the integer $e=2580882461$ such that $\gcd(e, \phi(f(x)))=1$ and $1<e<\phi(f(x))$. Using the extended Euclidean algorithm for integers, A finds $d=4894193141$ satisfying $ed\equiv 1 \pmod{\phi(f(x))}$. Hence, A's public-key is

$$(p=101, f(x)=100x^5+48x^4+28x^3+36x^2+40x+78, e=2580882461)$$

and A's private-key is

$$(d=4894193141, g(x)=28x^3+83x^2+3x+95, h(x)=18x^2+71x+88).$$

Public-Key Encryption: To encrypt the message $m(x)=1+x+3x^2$, entity B reduces the polynomial

$$c(x)=m(x)^e=(1+x+3x^2)^{2580882461}\equiv 8x^4+98x^3+39x^2+90x+40 \pmod{f(x)}$$

in $Z_{101}[x]$ and sends it to entity A.

Public-Key Decryption: To decrypt the ciphertext $c(x)=8x^4+98x^3+39x^2+90x+40$, A reduces

$$c(x)^d=(8x^4+98x^3+39x^2+90x+40)^{4894193141}\equiv(1+x+3x^2) \pmod{f(x)}$$

in $Z_{101}[x]$ to recover the original message $m(x)$.

3 RSA public-key scheme attack

The security of the RSA public-key encryption scheme is based on the intractability of both the integer factorization problem and the RSA problem. Various attack schemes have been studied in the literature as well as appropriate measures to counteract these threats. Given the public-key, to recover the plaintext m from the corresponding ciphertext c, a passive adversary must solve the RSA problem. There is no known efficient algorithm for this problem. One possible approach which an adversary could employ is to find the private key. In order to attack any protocol that uses the RSA public key encryption scheme by finding its private key, the factorization problem must be solved first. After factorization, the RSA problem could be solved by computing the value of Euler phi-function, and then finding the decryption exponent d using the extended Euclidean algorithm for integers. Once d is found, the adversary can recover the original message from any ciphertext intended for A.

On the other hand, if the classical method is used and an adversary could somehow compute d, then n can efficiently be factored as follows, see [11]. Since $ed\equiv 1 \pmod{\phi(n)}$, there is an integer k such that $ed-1=k\phi(n)$. Hence, by Euler theorem, $a^{ed-1}\equiv 1 \pmod{n}$ for all a such that $\gcd(a, n)=1$. Write $ed-1=2^s t$, where t is an odd integer. It can be shown that $a^{2\hat{}(s-1)t}$ is not congruent to either ± 1 modulo n for at least half of all integers a with $\gcd(a, n)=1$. If a is such an integer, then a non-trivial factor of n is $\gcd(a^{2\hat{}(s-1)t}-1, n)$. This shows that in the classical case, the RSA problem and the integer factorization problem are computationally equivalent. It is not known if this remains true for the modified schemes.

In the next section we evaluate the various RSA cryptosystems by recovering the private key using the software package Mathematica. We illustrate the attack schemes in the following example.

Table 1. Running time in seconds: Classical RSA

Size of primes	Classical RSA Public-Key	Encryption	Decryption
50-digit	0.0231	0.003	0.001
100-digit	0.1723	0.005	0.009
200-digit	0.7641	0.035	0.0431
250-digit	0.8092	0.052	0.0711
300-digit	2.0259	0.084	0.114

Table 2. Running time in seconds: Gaussian integers

Size of primes	Classical RSA Public-Key	Encryption	Decryption
50-digit	0.0231	0.026	0.026
100-digit	0.1723	0.091	0.097
200-digit	0.7641	0.4042	0.4529
250-digit	0.8092	0.666	0.7602
300-digit	2.0259	1.0275	1.1897

Example 4. Attacking the RSA cryptosystems

Assume that the public key is: $(n=8038438974502939,\ e=180977512554819)$. To find the private key, we use the built-in Mathematica functions FactorInteger and PowerMod. The prime factors p and q are obtained from the output of $FactorInteger[8038438974502939]$ which is $\{\{89657297, 1\}, \{89657387, 1\}\}$. Hence, $p=89657297$ and $q=89657387$. Next, we calculate $\phi(n)=(p-1)(q-1)=(89657297-1)(89657387-1)=8038438795188256$. The decryption exponent $d=2500998620710731$ is the output of $PowerMod[180977512554819, -1, 8038438795188256]$. The private key is $(p=89657297,\ q=89657387,\ d=2500998620710731)$.

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 5.0 as a programming language and an acer computer with Intel Pentium M715 processor, 1.5 GHZ CPU and 256 MB DDRAM.

RSA based Algorithms

Using Mathematica 5.0 functions and an additional abstract algebra library, we have written programs for the following algorithms:

- Classical RSA;
- RSA with Gaussian integers;
- RSA with polynomials over a finite field.

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.

The classical and Gaussian schemes were tested using the same public-key. The average running time of several runs using 50, 100, 200, 250 and 300-digit primes are given in Table 1 and Table 2. The public-key was generated by randomly selecting odd integers having a given number of digits and of the form $4k+3$ The odd integers were tested for primality using the built-in Mathematica function $PrimeQ$ until a prime is found.

To evaluate RSA algorithms using polynomials, we ran programs for various values of the prime p and degree of the irreducible polynomials. The average running time of several runs are listed in Table 3. The public-key was generated using the built-in Mathematica function $IrreduciblePolynomial[x, p, d]$.

Table 3. Running time in seconds: RSA using polynomials

RSA Using polynomials				
Prime p	Degree d	Public-Key	Encryption	Decryption
$p=2$	$2 \le d \le 10$	0.016	0.012	0.018
	$21 \le d \le 30$	0.4938	0.3306	0.5676
	$101 \le d \le 105$	25.3067	18.377	33.748
$p=101$	$2 \le d \le 10$	0.6349	0.1953	0.2814
	$11 \le d \le 20$	5.4139	1.3659	2.1471
	$21 \le d \le 30$	14.832	4.72356	7.21275

Table 4. Attack time in seconds: Classical RSA

Classical RSA					
Digits of p and q	20	22	24	26	30
Time	0.5969	1.6724	7.4156	11.8891	94.245

Table 5. Attack time in seconds: RSA algorithms using polynomials

RSA algorithms using polynomials				
Digits of p	5	5	10	22
Degree d	$10 \le d \le 11$	$12 \le d \le 13$	$5 \le d \le 6$	$2 \le d \le 3$
Time	3.565	17.836	4.186	7.53

Comparing these algorithms, we conclude the following:

1. All programs are reliable; they can encrypt and decrypt any message.
2. The running time for the encryption/decryption algorithms is negligible in the classical and Gaussian cases. In the polynomial case the time for the encryption/decryption

algorithms becomes significant for large primes and irreducible polynomials with large degree.

3. The complexity for the three programs depends on the complexity of generating the public-key. Thus, the classical and Gaussian algorithms are equivalent since their public-key generation algorithms are identical when restricting the choice of primes to those of the form $4k+3$. The Gaussian method is therefore recommended since the modified method provides an extension to the message space and the encryption exponent range.
4. The public-key generation algorithm using polynomials requires the search for irreducible polynomials. The Mathematica built-in algorithm for generating irreducible polynomials appears to be inefficient as p becomes very large and the degree of the polynomial increases.

Attack algorithm

In order to attack any protocol that uses the RSA public key encryption scheme by finding its private key, the factorization problem must be solved first. To test the security of the algorithms, we implemented attack schemes applied to the classical and modified cryptosystem algorithms. For the classical and Gaussian algorithms, we generated a public key using primes of various sizes. The attack was conducted using the Mathematica built-in function *FactorInteger* to recover the prime factors. The Euler phi-function was then computed. Finally, the decryption exponent was obtained. The average running time of several runs are listed in Table 4.

For the RSA algorithms using polynomials, we generated a public-key using a prime p of various sizes and irreducible polynomials $f(x)$ and $g(x)$ of different degrees d. The attack was conducted by factoring $f(x)$ using the built-in function $Factor[f, \text{modulus}->p]$ to recover the irreducible factors. The Euler phi-function was then computed. Finally, the decryption exponent was obtained. The average running time of several runs are listed in Table 5.

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 classical and Gaussian RSA algorithms 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 becomes more difficult as the size of p or the degree of the irreducible polynomials become larger.

5 Conclusion

In this work, we presented the classic RSA cryptosystem and two of its modifications, namely, the RSA cryptosystem 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 Gaussian method is preferred since it is as secure as the classical one but provides an extension to the message space and to the encryption exponent range.

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.

REFERENCES

1. Cross J.T. (1983), 'The Euler's ϕ-function in the Gaussian Integers', *American Mathematics Monthly*, vol. 90, pp. 518–528.
2. Diffie W. and Hellman M.E. (1978), 'New directions in cryptography', *IEEE Transaction on Information Theory*, IT–22, pp. 472–492.
3. El-Kassar A.N. and Haraty R. (2005), 'ElGamal public-key cryptosystem in multiplicative groups of quotient rings of polynomials over finite fields', *Journal of Computer Science and Information Systems (ConSIS)*, vol. 2, pp. 63–77.
4. El-Kassar A.N. and Haraty R.A. (2004), 'ElGamal public-key cryptosystem using reducible polynomials over a finite field', *Proceedings of the 13th International Conference on Intelligent & Adaptive Systems and Software Engineering (IASSE'2004)*, pp. 189–194.
5. El-Kassar A.N., Haraty R.A. and Awad Y. (2004), 'Modified RSA in the domains of gaussian integers and polynomials over finite fields', *Proceedings of the International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications (CSITeA'04)*, Cairo, Egypt.
6. El-Kassar A.N., Haraty R.A., and Awad Y. (2004), 'Rabin public-key cryptosystem in rings of polynomials over finite fields', *Proceedings of the International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications (CSITeA'04)*, Cairo, Egypt.
7. El-Kassar A.N., Rizk M., Mirza N.M. and Awad Y.A. (2001), 'ElGamal public-key cryptosystem in the domain of gaussian integers', *International Journal of Applied Mathematics*, vol. 7, no. 4, pp. 405–412.
8. Awad Y., Otrok H. and El-Kassar A.N. (2004), 'A comparative study of ElGamal-based cryptographic algorithms', *Proceedings of the Sixth International Conference on Enterprise Information Systems (ICEIS'2004)*, vol. 3 , pp. 79–84.
9. Haraty R., El-Kassar A.N. and Otrok H. (2005), 'Attacking ElGamal-based cryptographic algorithms using Pollard's Rho Algorithm', *Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'2005)*, Cairo, Egypt, January.
10. Kenneth A.R. (1988), *Elementary Number Theory and its Applications*, AT&T Bell Laboratories in Murray Hill, New Jersey.
11. Menezes A.J., Van Oorshot P.C. and Vanstone S.A. (1997), *Handbook of Applied Cryptography*, CRC press.
12. Rivest R., Shamir A. and Aldeman L. (1978), 'A method for obtaining digital signatures and publickey cryptosystems', *Communications of the ACM*, vol. 21, pp. 120–126.