Analysis of a Public Key Cryptosystem Using Standard and Homomorphic Approaches

Nazek A. AL-Essa

Princess Noura University, Science Faculty, Math Department, Riyadh, Saudi Arabia,
nazekaa@yahoo.com

Abstract: Information security within an organization is important in the last decades. The security approach based on cryptography can transform information messages to make them secure and immune to attack. This works discusses and presents the important mathematics for the RSA algorithm as one of the most popular public key cryptography. The steps of the RSA algorithm are: key generation, encryption, and decryption. The algorithm involves a public key for encrypting messages and a private key for decryption. The necessary mathematics based on number theory are analyzed, discussed, and presented. The encryption and decryption of any messages depend on N; where N is the product of two prime numbers. Both the public key and the private key are dependent on these prime numbers. Because RSA can be broken by factoring N; the security based on integer factorization problem is discussed and handled. Three factorization methods will be applied and compared. Moreover, two homomorphic encryption algorithms are also analyzed and discussed. Such algorithms are considered scalar and probabilistic. The inspiration for homomorphic encryption came from the properties of RSA. The homomorphic encryption algorithms are promising for providing security to many applications. The performance of both the RSA algorithm and those based on homomorphism is evaluated and compared.
[Nazek A. AL-Essa. **Analysis of a Public Key Cryptosystem Using Standard and Homomorphic Approaches.** *J Am Sci* 2013; 9(5):350-360]. (ISSN: 1545-1003). http://www.jofamericanscience.org. 44

Key words: Public key Cryptosystem, RSA Algorithm, Prime Numbers, Factorization Problem, Modular Exponentiation, and Homomorphic Encryption.

1. Introduction

With the expansion of the internet, cryptography became a very important area for several fields specially for those working in the computer environment. There is a lot of communication over insecure channels and such pieces of information should be protected against any unwanted access. In this concern, the data transferred from one system to another over a network can be protected by an encryption method. Encryption can be briefly defined as a process in which the sender encrypts the message in such a way that only the recipient will be able to decrypt it.

To clarify both the cryptography and secrecy suppose that a sender wishes to send a message to a receiver in such a way that anyone else receiving the message will not be able to understand it. In this concern, there are three important involved items mainly: the plaintext, the ciphertext, and the key. The plaintext is that unencrypted text which is what the sender wants to tell to the receiver. The ciphertext (or the encrypted text) is the message the sender actually sent to the receiver. The key is that important theme which tells how to convert the plaintext to ciphertext and vice versa Heribert Vallmer, and Rainer Parchmann, 2009.]. Since the key is known to both the sender and receiver, it is sometimes called the shared key. The conversion of plaintext to ciphertext and vice versa can be thought of as functions. If M is the set of all possible plaintext messages and C is the set of all possible ciphertext messages, then the key determines a function f_k: $M \rightarrow C$. That function is used by the sender to encipher the message. The receiver uses the inverse function f_k^{-1} to decipher the message. This means f_k must be an injection and f_k^{-1} must exist to decipher the encrypted message [Heribert Vallmer, and Rainer Parchmann, 2009]..

Moreover, there are two types of keys: secret (or symmetric) key and public (or asymmetric) key. In a system of n users, the number of secret keys for point-to-point communication is $n(n\text{-}1)/2 = O(n^2)$. With the public key encryption system there are two keys needed per user: one public key and one private key. So, the total number of needed keys is $2n = O(n)$ [Juan Monterde and Jose Vallejo, 2012], [Kenneth Jacobs, 2011].

The RSA cryptosystem is one of the most widely-used public key cryptography algorithm. The RSA algorithm can be used for both public key encryption and digital signatures. Its security is based on the difficulty of factoring large integers [Heribert Vollmer, and Rainer Parchmann, 2009]. Several research works were presented in the public key cryptography. Examples of such efforts are briefly mentioned as follows:-

[Juan Monterde and Jose Vallejo, 2012] mentioned that a basic problem in cryptography is the reduction of the number of operations needed to deal with big numbers. This is usually done through the

use of congruences. An example is the RSA algorithm, in which calculations are done modulo N, where N is the product of two big primes p and q. When using RSA encryption one has to compute expressions of the form $a^b \ mod \ c$, with large a, b. For these, the Chinese remainder theorem is not so practical, and other algorithms are preferred. From the mathematical point of view, essential ingredients of RSA cryptosystem are Fermat's Little Theorem and its generalization, and Euler's theorem on congruences.

[Kenneth Jacobs, 2011] presented that an encryption algorithm is a map e: $A \times K \rightarrow A$, where A is a set called an alphabet, and K is a set called the key space. A decryption algorithm is a map d: $A \times K \rightarrow A$, where the alphabet and key space are the same as for e. For each key $k_1 \in K$, there must exist $k_2 \in K$ satisfying $d(e(x, k_1), k_2) = x$. The collection (A, K, e, d) forms a cryptosystem.

[Dima Grigoriev, *et. al*., 2009] presented a cryptosystem which is complete for the class of probabilistic public-key cryptosystems with bounded error. To formalize the notion of "the hardest-to-break" cryptographic primitive, the authors used reductions. A cryptographic primitive S_1 is reducible to another primitive S_2 iff there exists a probabilistic procedure R (called a reduction) that breaks S_1 and S_2. As an illustration, every one-way function can be inverted by a polynomial-time algorithm. It makes sense to use complete cryptographic primitives as they are less likely to be breakable by polynomial-time adversaries.

The organization of this work is as follows: Sections 2 and 3 present the characterization of the public key cryptosystem and the arithmetic operations of RSA respectively. Section 4 discusses a homomorphic encryption algorithm while Section 5 presents some methods for integer factorization. The applicability of cryptosystems using RSA and Paillier homomorphic encryption is handled in Section 6. Finally, the discussion of results and conclusion are presented in Sections 7 and 8 respectively.

2. Characterization of Public Key Cryptography

As mentioned before, cryptography is the study of ways in which a sender and a receiver can securely communicate information. In this concern, there are two modes: secret key cryptography and public key cryptography. Secret key cryptography (sometimes known as symmetric key) employs identical private keys for users, while they also hold unique public keys. Symmetric keys refer to the identical private keys shared by users. On the other hand public key cryptography is known as asymmetric cryptosystem. The key of a public key cryptography is composed of a public part and a private part. The public key is accessible to any body, where the private key is kept undisclosed by its owner. In public key cryptography, the decryption function can't be easily derived from the encryption function and vice versa [Heribert Vallmer, and Rainer Parchmann, 2009]. Regardless the modes figure1 shows the main elements of cryptography and cryptanalysis [Mike Knee, 2008].

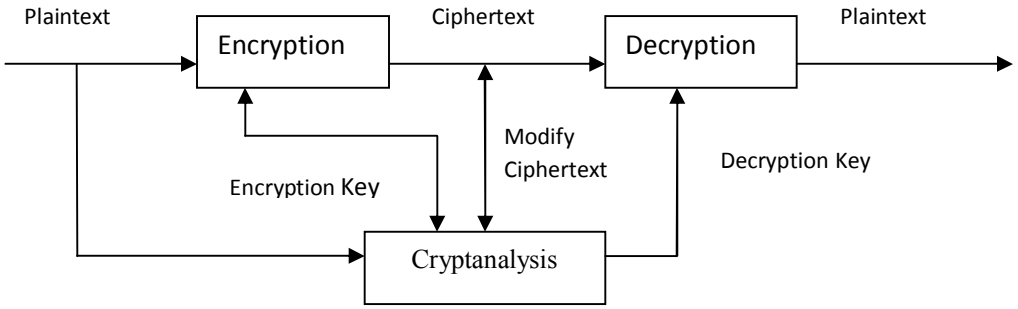

Figure 1: Cryptography and Cryptanalysis

The focus of this work is directed to the public key cryptography. A mathematical construct is important to specify the formal definition of the public key cryptosystem.

Definition 2.1: A public key cryptosystem (PKS) can be easily characterized by a set of components which are mentioned as follows:

PKS = (M, C, K, E, D), where

M is the message space; sometimes called the set of plaintexts or unencrypted text

C is the cipher text space; sometimes called the set of cipher texts or encrypted texts

K is the key space; sometimes called the set of keys. A key $k \in K$ consists of a public key e_k and a private key d_k

E is the encryption function which is written as

$$E_{e_k}: M \rightarrow C$$

D is the decryption function which is written as

$$D_{d_k}: C \rightarrow M$$

E and D should satisfy the following

$$D_{d_k} \circ E_{e_k} = 1_M$$

3. Arithmetic Operations of RSA Cryptosystem

RSA cryptosystem is one of the most popular asymmetric algorithms. It is well-known as a public key cryptography algorithm. It uses two keys; e and d; which work in pairs for encryption and decryption respectively. One of such keys is called a public key while the other is called a private key.

3.1 The Main Elements of the RSA Algorithm

The RSA algorithm has three important elements mainly: key generation, encryption, and decryption. The following sections briefly describe the arithmetic operations of such elements.

3.1.1 Key Generation

As mentioned before, the RSA cryptosystem handles two keys: a public key and a private one. The public key is used for encrypting data or messages and is known to everyone. The data or messages encrypted with the public key can only be decrypted using the private key.

Moreover, generation of keys is based on choosing two distinct prime numbers p and q. Such numbers should be of similar bit-length and randomly chosen [Satyendra Mandal, et. al., 2009]. After choosing the two large distinct prime numbers p and q by the key-owner he/she can compute the public modulus $N = pq$. N is used as the modulus for both the public and private keys. Then, a public exponent e is chosen to be coprime to $\phi(N) = (p-1)(q-1)$ such that $2 < e < \phi(N)$, e and $\phi(N)$ are coprime. e is released as the public key exponent. Choosing e is considered an addition chain results in more efficient encryption. The pair (e, N) forms the public key and is published while the private key d is determined using modular arithmetic. It is important to satisfy the congruence relation $ed \equiv 1 \ mod \ \phi(N)$ where d is considered as the private key exponent. The public key consists of the modulus N and the encryption exponent e while the private key consists of the modulus N and the decryption exponent d which must be kept secret [Satyendra Mandal, *et. al.*, 2009].

3.1.2 Encryption

Regarding the encryption steps, the message M is represented by a finite sequence of digits. If M consists of letters, it can be represented as: $A = 01$, $B = 02$, $C = 03$, …. , $Z = 26$. The message M is then broken into blocks $M = M_1 M_2$ …….. M_k, where each block satisfies $0 \leq M_i < N$. Each block is encrypted individually which means the plaintext is encrypted into cipher text.

$$C_i \equiv (M_i)^e \ mod \ N$$

3.1.3 Decryption

Decryption aims at recovering the original message M. This is done by reversing the concept as decryption works in an analogous way to encryption.

$$M_i \equiv (C_i)^d \ mod \ N$$

This means that the plaintext or message M is recovered [Heribert Vollmer, and Rainer Parchmann, 2009]. Because of symmetry in modular arithmetic, encryption and decryption are modular inverses and commutative. Moreover, the correctness of the RSA cryptosystem is guaranteed by the following steps in section 3.2.

3.2 Steps of the RSA Algorithm

From the above, the steps of RSA algorithm can be written. In this concern, two pairs of functions (f_{pub}, f_{pri}) are important for each user where *pub* and *pri* stand for public and private respectively. The steps can be briefly mentioned as follows [Junan Monterde and Jose Vallejo, 2012]:

1. The user chooses two large prime numbers p and q.
2. The product N can be computed as $N = pq$
3. The totient function on N and $\phi(N)$ can be evaluated
4. A number e can be chosen where
   $e \in \{1, 2, 3, ….. \phi(N)\}$, coprime with $\phi(N)$
   It is important to mention that e is an invertible element of the ring $\mathbb{Z}/\ \phi(N)\ \mathbb{Z}$ as the greatest common divisor $gcd(e, \phi(N)) = 1$
5. Let d is an integer where its equivalence class is the inverse of that e such that
   $$ed \equiv 1 \ (mod \ \phi(N))$$
6. The encryption function $f_{pub} : \mathbb{Z} / N\mathbb{Z} \rightarrow N / N\mathbb{Z}$ is given by
   $$f_{pub}(m) = m^e \ (mod \ N)$$
7. The decryption function f_{pri} is the inverse of f_{pub} such that
   $$f_{pri}\ (m) = m^d \ (mod \ N)$$

Definition 3.2.1: For any number x, let $\pi(x)$ is the number of primes p satisfying $2 \leq p \leq x$ i.e $\pi(8) = 4$ as the primes between 2 and 8 are 2, 3, 5, and 7.

Theorem 3.2.2:(Euler's theorem). Let $a \in \mathbb{Z}^*_m$ then $a^{\phi(m)} \equiv 1 \ mod \ m$, $gcd(a, m) = 1$.

Theorem 3.2.3:(Fermat's little theorem). Let p be prime and $a \in \mathbb{Z}^*_p$ then $a^{p-1} \equiv 1 \ mod \ p$.

Theorem 3.2.4: Let $a \geq 2$ be an integer, then a can be factored as a product of prime numbers

$$a = p_1^{e_1}.\ p_2^{e_2}.\ p_3^{e_3}\ . \ …….p_r^{e_r}$$

Definition 3.2.5: If p is prime, then the set $\mathbb{Z}/p\mathbb{Z}$ of integers modulo p with its addition, subtraction,

multiplication, and division rules is an example of a field.

The filed $\mathbb{Z}/p\mathbb{Z}$ of integers modulo p has only finitely many elements. It is a finitite field and is often denoted by $\mathbb{Z}_p$. Thus $\mathbb{Z}_p$ and $\mathbb{Z}/p\mathbb{Z}$ are just two different notations for the same object. Finitite fields are important throughout cryptography [Jeffrey Hoffstein, *et. al*., 2008].

Proposition 3.2.6: (Correctness of RSA)

$$M^{ed} \equiv M \bmod N$$

Proof As $ed \equiv 1 \bmod \phi(N)$, there must be $k \in \mathbb{N}$ such that

$$ed = k \,.\, \phi(N) + 1$$

$$M^{ed} = M^{(k \,.\, \phi(N)+1)} = M\,(M^{k \,.\, \phi(N)}) = M\,(M^{\phi(N)})^k$$

Euler's theorem states that $M^{\phi(N)} \equiv 1 \bmod N$ □

Proposition 3.2.7: If N is the product of two distinct prime numbers, then factoring N and computing $\phi(N)$ are equivalent.

Proof As mentioned above $N = pq$ and $\phi(N) = (p-1)(q-1)$. This means that $pq - p - q + 1 - \phi(N) = 0$ Substituting $q = N/p$ This gives the following

$$N - p - N/p + 1 - \phi(N) = 0 \text{ i.e.}$$

$$p^2 - p(N - \phi(N) + 1) + N = 0$$

$$p^2 - pA + N = 0 \text{ where } A = N - \phi(N) + 1$$

Solving this quadratic equation, it is easy to say that:

$$p, q = A/2 \pm (A^2/4 - N)^{1/2}$$

Also, if the prime factors p and q are known then $\phi(N)$ equals $(p-1)(q-1)$ □

Theorem 3.2.8: Let $N = pq$, $p < q < 2p$, $d < 1/3 \sqrt[4]{N}$, then d can be computed.

Theorem 3.2.9: For every $m \in Z/NZ$

$$f_{pri}\,(f_{pub}\,(m)) \equiv f_{pri}\,(m^e) \equiv m^{ed} \equiv m (mod\ N)$$

Proposition 3.2.10: If p is prime and r is a positive integer, then $\phi(p^r) = p^r - p^{r-1}$.

Examples for RSA Algorithm 3.2.11

To clarify the functions of RSA algorithm, let us present the following two examples [Artan Luma, and Nderim Zeqiri, 2012], [Natarajan Meghanathan, 2012]:

Let p=17 and q=23 are two prime numbers. It is required to find both the encryption and decryption keys to cipher a plaintext say 127.

Solution

As $N = pq$, so $N = 17*23 = 391$

As $\phi(N) = (p-1)(q-1)$, so $(p-1)(q-1) = 352$

The encryption exponent e is chosen which is relatively prime with $(p-1)(q-1)$, so e is assumed to be 13. This means that the encryption key is (13, 391).

To cipher a plaintext or message $m = 127$, so

$$c \equiv m^e \bmod (N)$$

$c \equiv 127^{13} \bmod (391)$

c can be easily found as shown in the following steps:

$(127)^1 \bmod 391 = 127$

$(127)^2 \bmod 391 \equiv (127 * 127) \bmod 391 = 16129 \bmod 391 = 98$

$(127)^4 \bmod 391 \equiv ((127)^2 * (127)^2) \bmod 391 \equiv (98 * 98) \bmod 391 \equiv 9604 \bmod 391 = 220$

$(127)^8 \bmod 391 \equiv ((127)^4 * (127)^4) \bmod 391 \equiv (220 * 220) \bmod 391 \equiv 48400 \bmod 391 = 307$

$(127)^{13} \bmod 391 \equiv ((127)^8 * (127)^4 * (127)^1) \bmod 391 \equiv (307 * 220 * 127) \bmod 391 = 213$, so the cipher text is 213.

The decryption of cipher text $c = 213$ can be obtained through

$$d \,.\, e \equiv 1 \,(mod\,(p-1)(q-1))$$

As the decryption key d is the multiplicative inverse of 13 modulo 352, so d is found to be 325. The decryption of the cipher text $c = 213$ can be found through

$$m \equiv c^d \bmod N$$

$m \equiv (213)^{325} \bmod 391$. Using the same computing procedure mentioned above, so $(213)^{325} \bmod 391 \equiv ((213)^{256} * (213)^{64} * (213)^4 * (213)^1) \bmod 391 \equiv (239*154*169*213) \bmod 391 = 127$. So, the plaintext is 127.

3.3 Modified RSA Algorithm

The RSA cryptosystem takes long computation cost. This cost is due to several themes specially for performing the modular exponentiation. The decryption operation takes long computation cost specially when using large public keys. This part tries to enhance the hardness of factoring N. It is a good thing to reduce the computation cost specially that decryption time than that consumed in the traditional RSA algorithm. This can be done by the following [Ren-Junn Hwang, *et. al*., 2005]:

1. The secret key is better to know the prime factors p-1, p+1, q-1, q+1.
2. $(p-1)$ and $(q-1)$ should contain a large prime factor such that $r_1 | (p-1)$ and $t_1 | (q-1)$, where r_1 and t_1 are two large primes.
3. $(p+1)$ and $(q+1)$ should contain a large prime factor such that $s_1 | (p+1)$ and $u_1 | (q+1)$, where s_1 and u_1 are two large primes.
4. The moduli p-1, p+1, q-1, and q+1 can be represented as follows:-

$$p-1 = \Pi^h_{i=1}\, r_i^{\,\alpha_i}$$

$$p+1 = \Pi^j_{i=1}\, s_i^{\,\beta_i}$$

$$q-1 = \Pi^k_{i=1}\, t_i^{\,x_i}$$

$$q+1 = \Pi^l_{i=1}\, u_i^{\,\delta_i}$$

5. The modular exponentiations can be computed as follows:-

$$y_1 \equiv c^d \bmod (p-1)$$

$$y_2 \equiv c^d \bmod (p+1)$$

$$y_3 \equiv c^d \bmod (q-1)$$

$$y_4 \equiv c^d \bmod (q+1)$$

6. x_1 and x_2 are computed such that
   $x_1 = 2^{-1} (y_1 + y_2 - z) \ mod \ p$, where $z = 0$ if $y_1 \geq y_2$; otherwise $z = 1$
   $x_2 = 2^{-1} (y_3 + y_4 - z) \ mod \ q$, where $z = 0$ if $y_3 \geq y_4$; otherwise $z = 1$
7. Generate the plaintext $m \equiv c^d \ mod \ N$ based on x_1 and x_2.

Theorem 3.4.1: Let p and q are distinct primes and $g = gcd\ (p-1, q-1)$
Then $a^{(p-1)\ (q-1)/g} \equiv 1 \ mod \ (pq)$ for all a satisfying $gcd\ (a, pq) = 1$

Proposition 3.4.2: Let p and q are distinct primes and $e \geq 1$ such that $gcd(e, (p-1)(q-1)) = 1$ and $de \equiv 1 \ (mod (p-1)(q-1))$
Then the congruence $x^e \equiv c \ (mod \ pq)$ has the unique solution $x \equiv c^d \ (mod \ pq)$

4. Homomorphic Encryption

Homomorphic encryption describes a property of an encryption scheme. Such property can perform computations on the ciphertext without decrypting it first. An encryption algorithm/scheme is homomorphic if it is possible to perform implicit operation on the plaintext by processing the ciphertext only. The scheme is called fully homomorphic when it is performed a sequence of operations both addition and multiplication. The scheme is somewhat homomorphic if it supports a limited number of operations [Nitin Jain, *et al.,* 2012], [Guilhem Castagnos and Fabien Laguillaumie, 2012]. It is important to mention that homomorphic cryptography can be used for several applications. Examples of such applications are: auction, voting, multi-party computation, digital signatures, and message authentication codes [Salil Vadhan, and Alon Rosen, 2006], [Ron Rivest, 2002].

Definition 4.1:

A homomorphic encryption can be defined by three polynomial-time algorithms (G, E, D), with at least one pair of polynomial-time computable binary operations (addition or multiplication) on the plaintext space P and the ciphertext space C respectively.
To illustrate such homomorphic encryption scheme, it is important to mention that:

1. The key generation algorithm G is a randomized algorithm which takes a security parameter 1^n as input and returns a pair (pk, sk) where pk and sk are the primary key and secret key respectively. It can be written as $(pk, sk) \xleftarrow{R} G(1^n)$.
2. The encryption algorithm E takes the public key pk and a plaintext m and outputs a ciphertext $c = E_{pk}(m)$. It can be written as $c = E_{pk}(m, r)$ where r is a random help value. The encryption algorithm E is homomorphic between the plaintext space P and the ciphertext space C if $c_1 = E(m_1)$ and $c_2 = E(m_2)$, then $c_1 \odot c_2 \in E(m_1 \odot m_2)$.
3. The decryption algorithm D is a deterministic algorithm in the sense that it takes a ciphertext $c = E(m, r)$ and either the secret key sk or the value r and returns a plaintext m. It can be written as $m = D_{sk}(c)$ or $m = D_r(c)$

4.1 Homomorphic Property of RSA

If the RSA public-key is modulus m and exponent e, then the encryption of a message x is given by $E(x) = x^e \ \text{mod} \ m$. The homomorphic property is then

$$E(x_1) \ . \ E(x_2) = x_1^e \ x_2^e \ mod \ m = (x_1 x_2)^e \ mod \ m = E(x_1 x_2)$$

The following example is presented to illustrate that concept.

Example

Assume that $p = 47$ and $q = 71$ are two prime numbers so $N = p\ q = 3337$, then $\phi = (p-1)(q-1) = 3220$
The public exponent e is chosen to be relatively prime to 3220, so $e = 79$. The public key is (N =3337, e =79). The inverse of $e \ mod \ N$ is $d = 79^{-1} \ mod \ 3220 = 1019$
Consider two messages that are 75 and 39 respectively.
The calculation for 75 is $E((3337, 79), 75) = 75^{79} \ mod \ 3337 = 2213$
The calculation for 39 is $E((3337, 79), 39) = 39^{79} \ mod \ 3337 = 2739$
The calculation for (75)(39) = 2925 is $E((3337, 79), 2925) = 2925^{79} \ mod \ 3337 = 1415$
It is noted that: (2213)(2739) = 6061407 i.e. 6061407 mod 3337 = 1415
This ensures the property from the multiplicative properties of modular arithmetic

$$(x_1 x_2)^e \equiv x_1^e \ x_2^e \equiv c_1 c_2 \ (mod \ N)$$

4.2 Paillier Homomorphic Encryption

Paillier homomorphic scheme is an efficient probabilistic and additive encryption scheme. Such scheme is an example of scalar homomorphic cryptosystems. The Paillier homomorphic encryption scheme is briefly described as follows [Nitin Jain, et. al., 2012], [Nitin Jain, et. al., 2012], [Salil Vadhan and Alon Rosen, 2006]

Paillier's trapdoor function is an isomorphism $f: \mathbb{Z}_N \times \mathbb{Z}_N^* \rightarrow \mathbb{Z}_{N^2}^*$ given by $f(a,b) = (1+N)^a \ . \ b^N \ mod \ N^2$,

where $N = pq$ for distinct odd primes p, q of equal length.

Assume that a is the message m, and b the random help value r, the Paillier homomorphic scheme is defined by the three polynomial-time algorithms (G, E, D) as follows:

1. The Key Generation Algorithm

1.1 Let A be a polynomial-time algorithm that, an input 1^n, outputs (N, p, q) where $N = pq$

1.2 Input 1^n run $A(1^n)$ to obtain (N, p, q). The public key is N and the private key is $(N, \phi(N))$, where $\phi(N)$ is the Euler's Totient function.

1.3 Set $pk = N = pq$, $sk = \phi(N) = (p-1)(q-1)$ where p and q are two n-bit primes.

2. The Encryption Algorithm

2.1 Let $m \in \mathbb{Z}_N$ is the message to encrypt and r is the random help value where $r \xleftarrow{R} \mathbb{Z}^*_N$

2.2 Set $c = E_N(m, r) = (1 + N)^m . r^N \equiv (1 + mN) r^N \bmod N^2$

3. The Decryption Algorithm

3.1 To decrypt the ciphertext c into plaintext m compute $\hat{c}$ using $\phi(N)$ as follows:

$$\hat{c} = c^{\phi(N)} \equiv (1 + N)^{m\,\phi(N)} \equiv (1 + m\,\phi(N)N) \bmod N^2$$

3.2 Compute m' as $m' = ((\hat{c} - 1) / \phi(N)) \bmod N^2$ where $m = m' / N$

3.3 Compute $\hat{c}$ to decrypt c using r where $\hat{c} = c\, r^{-N} \bmod N^2$, $m = (\hat{c} - 1) / N$

3.4 If the secret key sk is known, r can be recovered from c as follows:

$$r = c^{N-1 \bmod \phi(N)} \bmod N$$

4.3 Damgard-Jurik Homomorphic Encryption

This approach is considered a probabilistic homomorphic cryptosystem [Nitin Jain, *et. al.*, 2012]. Such homomorphic cryptosystem has three main phases mainly: key generation, encryption, and decryption. The details of such phases are as follows:

1. Key Generation

1.1 An RSA modulus is chosen as $N = pq = (2p' + 1)(2q' + 1)$ with primes p, p', q, q'

1.2 An element g is selected as $g \in Q_N$, where Q_N refers to the group of all squares in $(\mathbb{Z}/N\mathbb{Z})^*$, α is chosen as $\alpha \in \mathbb{Z}/\tau\mathbb{Z}$, where $\tau = pq = \{ Q_N \}$

1.3 h is computed as $h = g^{\alpha} \bmod N$

1.4 The public key a $\coloneqq (N, g, h)$ while the secret key is α.

2. Encryption

2.1 An integer s is chosen as $s > 0$ such that $m \in \mathbb{Z}/\, N^s\, \mathbb{Z}$

2.2 A random r is chosen as $r \in \mathbb{Z}/n\mathbb{Z}$, where $n = 4^{\log_2 N}$

2.3 $E_a(m, r)$ is the ciphertext where

$$E_a(m, r) = (g^r \bmod N, (h^r \bmod N)^{N^s} (N+1)^m \bmod N^{s+1}) \coloneqq (G, H)$$

2.4 A function L_s is set where

$$L_s((N+1)^m \bmod N^{s+1})\ m \bmod N^s$$

3. Decryption

3.1 The message m is found as:

$$m = L_s(h(g^a \bmod N)^{-N^s}) = L_s(g^{\alpha r} \bmod N)^{N^s} (N+1)^m (g^{\alpha r} \bmod N)^{-N^s}) = L_s((N+1)^m \bmod N^{s+1})$$

3.2 From $E_a(m_1, r_1)$ and $E_a(m_2, r_2)$ and the additive homomorphic approach $E_a(m_1 + m_2, r_1 + r_2)$ can be computed as:

$$E_a(m_1 + m_2, r_1 + r_2) = (g^{r_1 + r_2} \bmod N, (h^{r_1 + r_2} \bmod N)^{N^s} (N+1)^{m_1 + m_2} \bmod N^{s+1}) = E_a(m_1, r_1) \times E_a(m_2, r_2)$$

5. Integer Factorization for RSA Algorithm

The choice of secret primes is very important in the RSA cryptosystem. This means that the factorization is important for the key generation, encryption, and decryption [Jeffrey Hoffstein, *et. al.*, 2008].

As $N = pq$, the factorization process aims at determining the prime factors p and q. To find the primes p and q, it is better to consider an integer L where p-1 divides L and q-1 does not divide L.

$L = i(p-1)$ and $L = j(q-1) + k$, where i, j and k are integers and $k \neq$ zero.

From Fermat's Little Theorem, it is easy to say that:

$$a^L = a^{i(p-1)} = a^{(p-1)i} \equiv 1^i \equiv 1 (mod\ p),$$

$$a^L = a^{j(q-1)+k} = a^k\, (a^{q-1})^j \equiv a^k\, .\, 1^j \equiv a^k\ (mod\ q)$$

For the most choices of a, it is easy to say that p divides $a^L - 1$ and q doesn't divide $a^L - 1$. Moreover, p can be recovered with the simple *gcd*, so

$$p = gcd\ (a^L - 1, N)$$

If p-1 is a product of some small primes, then it will divide $n!$ for some values of n. For each number $n = 2, 3, 4, \ldots\ldots$ the value of a is chosen and computed

$$gcd\ (a^{n!} - 1, N)$$

If a number between 1 and N is obtained, then a nontrival factor of N is done [Jeffrey ,*et. al.,* 2008].

In fact, there are several methods of factorization; three of such methods will be briefly presented in the following sections.

5.1 Trial Division Factorization Method

This method is one of the oldest and important methods for factorizing an integer N. The trial division method can factorize an integer N by dividing that integer by any integer greater than one

and less than N. That method checks all possible prime factors of N. It starts from 2 and works up to $\sqrt{N}$. i.e such method requires $\pi(N) \approx \frac{2\sqrt{N}}{ln(N)}$ trial divisions, where $\pi(N)$ denotes the number of primes less than N. The factorization can be done by dividing the integer N by small prime numbers starting with 2, 3, 5, ... and so on. If the remainder of the division is zero, N is divisible by that prime number. Such process is repeated to keep all the prime numbers less than or equal to $\sqrt{N}$ than can divide N. This factorization method is simple and easy to understand. It performs well for small prime numbers up to a certain number of decimal digits. However, it will take long time to try all possible large prime factors. For example to factorize 95 it is advised to divide 95 by 2, 3, 5, 7. The remainder when 95 is divided by 5 is 0. So, 5 is one of the prime factors of 95. The other factor is 95/5=19.

Such type of factorization method is based on the fact that composite numbers N have at least one prime factor $\leq \sqrt{N}$. This method is an important approach for clearing integers N from small prime factors. Using such method, it needs approximately $\frac{\sqrt{N}}{log\sqrt{N}}$ divisions with remainders [Wikipedia, 2012]. For large number with potentially large prime factor, it may take a very long time to try all the possible prime factors.

5.2 Fermat Factorization Method

Fermat's factorization method is based on the representation of an odd integer as the difference of two squares: $N = a^2 - b^2$ Fermat's method tries values of a to check that $a^2 - N = b^2$ is a square [Wikipedia, 2012]. The algorithm of this method is briefly mentioned as follows:

Algorithm: Fermat Factorization (N) //N should be odd//
$a \leftarrow$ ceil ($\sqrt{N}$)
$b^2 \leftarrow a^2 - N$
While b^2 isn't a square
$a \leftarrow a + 1$
$b^2 \leftarrow a^2 - N$
Endwhile
Return $a - b$ //or $a + b$

If N is prime, one needs $O(N)$ steps. If N has a factor close to its square root, the method works quickly.

This factorization method rewrites the composite number N as the difference of squares as $N = a^2 - b^2$ [B.R. Ambedkor and S.S. Bedi, 2011]. This means $N = (a+b)(a-b)$. This means that P and Q are nontrivial odd factors of N such that $N=PQ$ where $P \leq Q$ and $P = (a-b)$ and $Q = (a + b)$. To factorize the number 95 (for example), the following steps are followed:

1. Let N =95 {the product of two prime numbers}
2. Decimal digits: 2 bits
3. Let factors are P and Q
4. Let a is $\sqrt{N}$ and take the ceiling function
5. Compute b as $b^2 = a^2 - N$
6. Check if b is integer then it is a factor otherwise increment a and compute again b.
7. This operation is repeated till b is an integer number; then y is a factor of N.

Moreover, Fermat's method is very good at finding the factorization of $N = PQ$ if P and Q are very close to each other. Fermat's method factors N quickly if N can be factored as $N = ab$ with $a, b \approx \sqrt{N}$. This algorithm is more efficient than trial division and it is used by some factorization packages to find factors between 5 and 11 digits [Franz Lemmermeyer, 2006].

5.3 Factorization Based on Pollard'Rho Method

This factorization is based on Pollard'Rho method and $\sqrt{N}$ [B.R. Ambedkar, and S.S. Bedig, 2011]. The main steps of this method are as follows:

1. Let the integer number to be factorized is N
2. LOOP
3. S = ceiling of ($\sqrt{N}$)
4. Let $x = S^2$ -1
5. Compute $gcd(x, N)$
6. IF the gcd >1
   THEN it is a factor of N and the other factor is N/gcd
   ELSE decrement S and go to LOOP till $x > 1$

To illustrate that method the following example is presented.

Example
Let N = 95
S =10
$P = gcd$ (99, 95) =1
S = 9
$P = gcd(80, 95)=5$
$Q = N/P = 95/5 = 19$
P and Q are factors of N

6. Applicability of the RSA and Homomorphic Cryptosystem

The public-key system is mainly characterized by the use of the cryptographic algorithm mentioned before. Such algorithm adopts two keys: one as a private while the other is a public key. Using the RSA cryptosystem, the sender encrypts

a message with the recipient's public key. Using the RSA mathematics mentioned before, the encryption and decryption operations of RSA were applied. This was done on different data sizes as well as different key sizes. Figures 2, 3 present the encryption and decryption time for some adopted different data sizes and key sizes. Figures 4 to 11 show the encryption and decryption time (ms) for the RSA, modified RSA, and homomorphic Paillier and Damgard-Jurik encryption. This involves also the key sizes and data sizes respectively. Figure 12 shows the number of iterations consumed to factorize some integer numbers using the three adopted factorization methods.

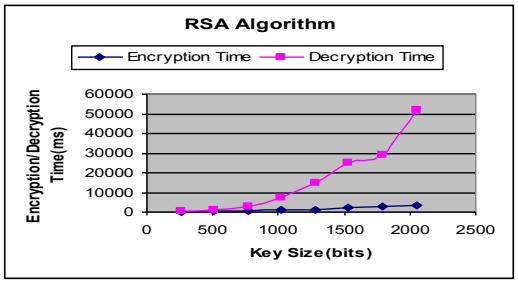

Figure 2: Time (ms) Vs. Key Sizes (bits)

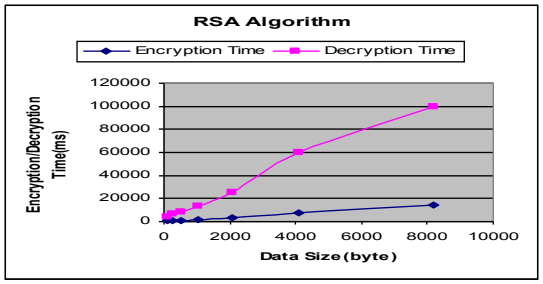

Figure 3: Time (ms) Vs. Data Sizes (Bytes)

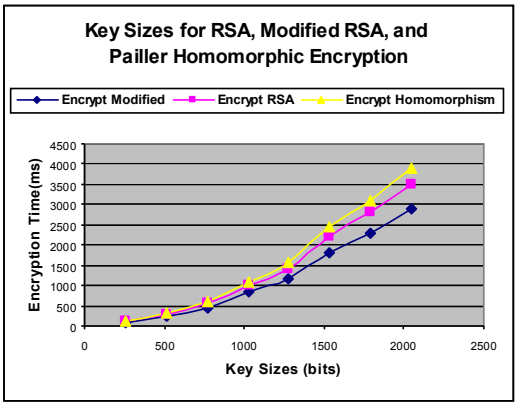

Figure 4: Encryption Time (ms) vs. Key Sizes (bits)

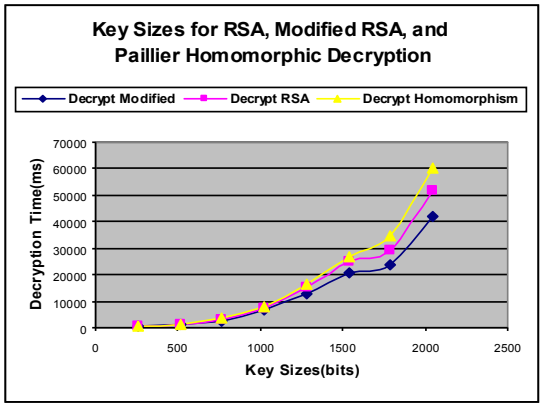

Figure 5: Decryption Time (ms) vs. Key Sizes (bits)

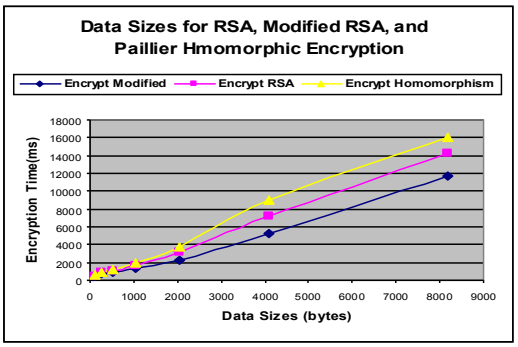

Figure 6: Encryption Time (ms) vs. Data Sizes (bytes)

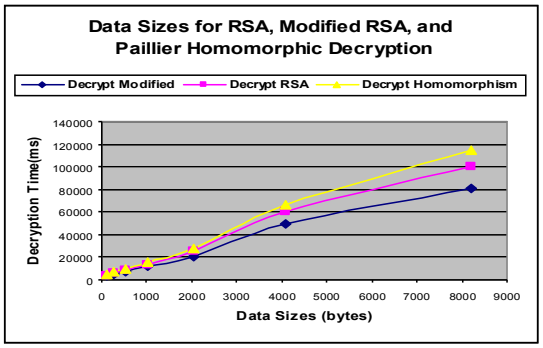

Figure 7: Decryption Time (ms) vs. Data Sizes (bytes)

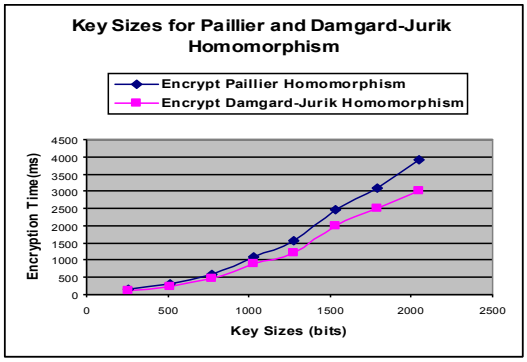

Figure 8: Encryption Time (ms) vs. Key Sizes (bits)

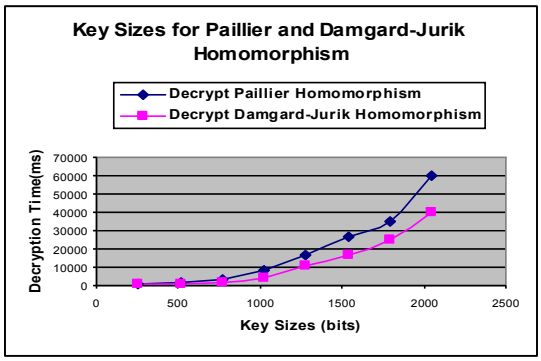

Figure 9: Decryption Time (ms) vs. Key Sizes (bits)

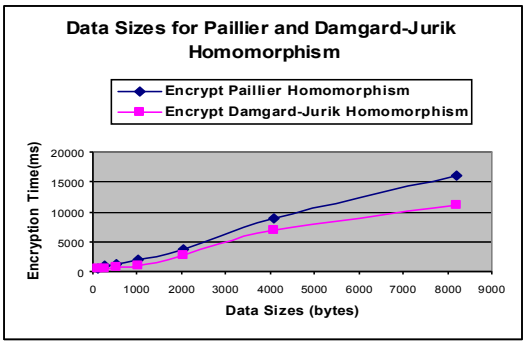

Figure 10: Encryption Time (ms) vs. Data Sizes (bytes)

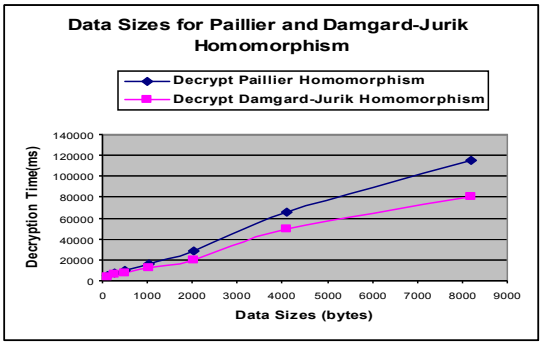

Figure 11: Decryption Time (ms) vs. Data Sizes (bytes)

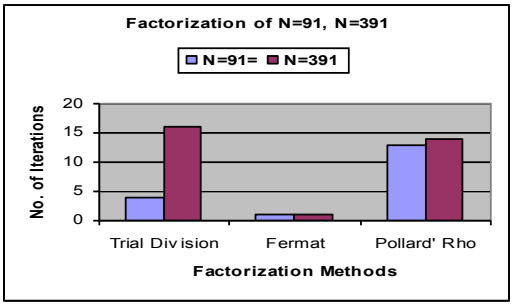

Figure 12a: Cost for Integer Factorization

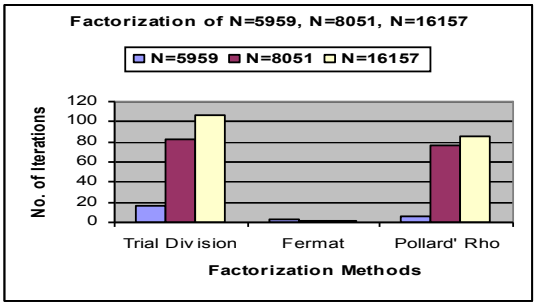

Figure 12b: Cost for Integer Factorization

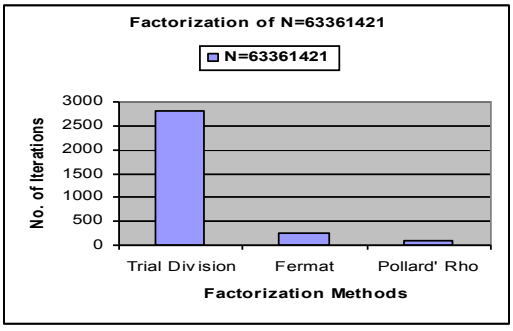

Figure 12c: Cost for Integer Factorization

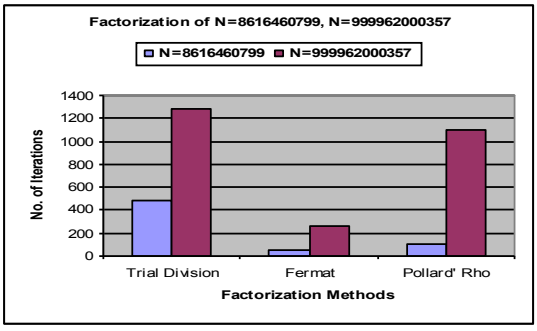

Figure 12d: Cost for Integer Factorization

7. Discussion of Results

The RSA cryptosystem is one of the most widely used asymmetric encryption system. The private key is kept secret while the public key is revealed to everybody in RSA. The necessary mathematics was presented to manipulate such public key cryptosystem. It is important for a sender to compute the cipher text by using the public key of the receiver for any message. The receiver should be able to decrypt the cipher easily to plaintext using his

private key. The security of RSA depends on the difficulty in factoring N into p and q if N is sufficiently large. It is advised to choose the size of N such that the cost for performing the factorization exceeds the value of the encrypted information.

Both the encryption and decryption times for the modified RSA algorithm are less than their corresponding values of the RSA algorithm. This has been occurred for the adopted key and data sizes. From the results it is shown that bigger keys result in longer encryption and decryption time. As the key size increases, the difference between encryption and decryption times also increases. Similarly, increasing the data sizes cause longer encryption and decryption time. The decryption time is always longer than the encryption time for different data sizes or key sizes. Moreover, the number of iterations to detect a prime number increases as the length of that number (in digits) increases. This means more time is consumed to detect large prime numbers. It is important to mention that homomorphic encryption enables mathematical operations to be performed on encrypted data without revealing the contents of the original plaintext. The cryptosystem is additive or multiplicative homomorphic depending upon the operation which can be addition or multiplication. Paillier cryptography algorithm has benefits. It gives the homomorphic property needed for some applications such as voting. Paillier cryptosystem is semantically secure assuming that it is hard to distinguish N^{th} residues from non-N^{th} residues *mod* N^2 [Ron Rivest, 2002]. The performance of Damgard-Jurik homomorphic algorithm is better than that of Paillier homomorphic algorithm. It is considered the best one compared to the other adopted algorithms in this work. This is true for the adopted key sizes as well as data sizes.

Moreover, the performance of the three adopted methods for integer factorization is different. Not all the factorization methods take the same factorization time. The trial division is easy and performs well for identifying small prime factors. It takes long time to find large prime factor in a large integer number. The Pollard'Rho method is effective and better than the trial division method for finding large prime factors in large integer numbers. That method is effective at splitting composite number with small factors. The majority of experiments show that the Fermat factorization method was the best compared to the other two methods. Fermat's method is very good at finding the factorization of $N = pq$ if p and q are very close to each other. Also, if N has a factor close to its square root that method works quickly.

8. Conclusion

The RSA algorithm has various security issues based on mathematical calculations. Although RSA is a good approach for security purpose, its key length is large which consumes much time to decrypt any message or plaintext. Due to the advance development in the factorization field of large primes, the key length for securing RSA is being increased. Increasing the key length causes a promising increase in the security of RSA cryptography. On the other hand, this increases the computation cost. i.e. large key sizes have regular increase in computing power and continuing refinement of factoring power.

Moreover, one of the main promising properties of the Paillier encryption is that it is additively homomorphic over plaintexts and also allows multiplication of plaintexts by a constant. This is also valid for the Damgard-Jurik homomorphic encryption in addition to its applicability for a wider application scope. The fully homomorphic encryption scheme allows to compute arbitrary functions over encrypted data without the decryption key. Encryption schemes based on homomorphism are critical for the design of highly functional cryptosystems.

References

1. Heribert Vollmer, and Rainer Parchmunn, “Cryptographic Applications of Algorithemic Number Theory”, Presented to Goufried Wilhlm Universitat Hannover, Fakultat Fur Elektrotechnik Und Informatik, November, 2009.
2. Juan Monterde and Jose Vallejo, “Implementing the RSA Cryptosystem with Maxima CAS”, The Electronic Journal of Mathematics and Technology, Vol. 6, No. 1, PP. 34-53, 2012.
3. Kenneth Jacobs, “Survey of Modern Mathematical Cryptography”, A Thesis Project Presented to the University of Tennesssee Honors Program, PP. 1-12, April 2011.
4. Dima Grigoriev, Edward Hirsch, and Konstantin Pervyshev, “A Complete Public-Key Cryptosystem”, Groups-Complexity-Cryptography, Heldermann Verlag, Vol. 1, No.1, PP. 1-12, 2009.
5. Mike Knee, “Good Old Mathematics: The Basis of Cryptography”, Snell & Wilcox, 2008.
6. Satyendra Mandal, Kumarjit Banerjee, Biswajit Maiti, and J. Palchaudhury, “Modified Trail Division for Implementation of RSA Algorithm with Large Integers”, The International Journal of Advanced Networking and Applications, Vol. 1, No. 4, PP. 210-216, 2009.

7. Karl Petersen, “Notes on Number Theory and Cryptography”, PP. 1-23, Downloaded From the Internet in 2012 From the Website http://math.unc.edu/. Faculty/petersen/Coding/cr2.pdf
8. Jeffrey Hoffstein, Jill Pipher, and Joseph Silverman, "An Introduction to Mathematical Cryptography", Springer, 2008.
9. Natarajan Meghanathan, "Number Theory and RSA Public-Key Encryption", Downloaded in 2012 From the Website http://williamstallings.com/Extras/ Security-Notes/lectures/publickey.html.
10. Artan Luma, and Nderim Zeqiri, "Data Encryption using an Algorithm Implemented in RSA Algorithm", Downloaded in 2012 From the Website http://artanluma.com/pdf/Data_encryption_using_AN_algorithms_implemented_in_RSA_algorithm.pdf.
11. Ren-Junn Hwang, Feng-Fu Su, Yi-Shuing Yeh, and Chia-Yao Chen, "An Efficient Decryption Method for RSA Cryptosystem", The Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA'05), PP. 1-6, 2005.
12. B.R.Ambedkar and S.S.Bedi, "A New Factorization Method to Factorize RSA Public Key Encryption", The International Journal of Computer Science Issues, Vol. 8, Issue 6, No. 1, PP. 242-247, November 2011.
13. Joseph Rabaiotti, "Implementing on RSA Cryptography System for Windows", A Report Presented to the Computer Science Department, Cardiff University, April, 2003.
14. Punita Meelu, and Rajni Meelu, "Implementation of Public Key Cryptographic System: RSA", The International Journal of Information Technology and Knowledge Management, Vol. 5, No. 2, PP.239-242, July-December 2012.
15. Gerard P. Michon, "Prime Factorization", Downloaded From the Internet in 2012 From the Website http://www.numericana.com/answer/factoring.htm.
16. Kostas Bimpikis, and Ragesh Jaiswal, "Modern Factoring Algorithms", A technical Report Presented to the University of California, San Diego, PP. 1-15, 2005.
17. Franz Lemmermeyer, "Introduction to Cryptography", December 15, 2006, Downloaded From the Internet From the Website http://www.fen.bilkent.edu.tr/~franz/crypto/cryp06.pdf.
18. Ljupco Kocarev, Marjan Sterjev, Attila Fekete, and Gabor Vattay, "Public-Key Encryption With Chaos", The American Institute of Physics, Vol. 14, No. 4, PP. 1078-1082, December 2004.
19. L. Sreenivasulu Reddy, "Efficient on Board J2-RSA Key Generation With Smart Cards", The IJSCE Journal, Vol. 2, No. 2, PP. 75-79, May 2012.
20. Wikipedia, "Factorization Methods", Downloaded From the Internet in 2012 From the Website http://www.wikipedia.com.
21. Nitin Jain, Saibal Pal, Dhananjay Upadhyay, "Implementation and Analysis of Homomorphic Encryption Schemes", The International Journal on cryptography and Information Security, Vol. 2, No. 2, PP. 27-44, 2012.
22. Guilhem Castagnos and Fabien Laguillaumie, "Homomorphic Encryption for Multiplication Pairing Evaluation" Springer-Verlag Berlin Heidelberg, LNCS 7485, PP. 374-392, 2012.
23. Salil Vadhan and Alon Rosen, "Public-Key Encryption in Practice, Downloaded From the Internet From the Website http://www. , PP. 1-7, 2006.
24. Ron Rivest, "Voting and Homomorphic Encryption", Downloaded From the Internet From the Website http://www.cnn.com/2002/allpoltics/10/29/elec02. bush.changes.ap/index.html.

3/27/2013