Modified Elgamal Cryptosystem Algorithm (MECA)

Prashant Sharma
M.Tech. Student
SBCET, Jaipur, Rajasthan, India
E-mail:-
prashant_harmony@rediffmail.com

Sonal Sharma
CSE & IT Department
MIT, Kota, Rajasthan, India
E-mail:-
sonal.gold@gmail.com

Ravi Shankar Dhakar
CSE Department
GIET, Kota, Rajasthan, India
E-mail:-
ravi.dhakar83@gmail.com

Abstract- **In 1985 a powerful and public-key scheme was produced by ElGamal. ElGamal encryption/decryption algorithm is based on the difficulty of discrete logarithm problem where it is straight forward to raise numbers to large powers but it is much harder to do the inverse computation of the discrete logarithm. Now, there are so many algorithms available which can solve the discrete logarithm problem of small size numbers in a reasonable time. So to improve security, we proposed a Modified ElGamal Cryptosystem Algorithm (MECA) to enhance the security for encrypting long messages and also secure against mathematical and brute-force attack as well as Low-Modulus and Known-Plaintext attack on Elgamal. The security of this algorithm is based on the difficulty of solving the discrete logarithm problem and Integer factorization problem. This paper also presents comparison between MECA and ElGamal cryptosystem in respect of security and complexity.**

Keywords- Encryption, Public key, Private key, Security, ElGamal parameters, Rabin.

I. INTRODUCTION

The importance of cryptography system and techniques is becoming a fundamental issue for large sector of society. The use of encryption is so important for both storing and transmitting the data. In order to secure this digital data, strong cryptography techniques are required.

The importance of encryption is increasing rapidly due to the growing traffic on the public network like Internet. Private persons, companies, government agencies and other organizations use encryption techniques in order to safely communicate with partners and costumers [1].

Cryptography is the study of mathematical techniques related to aspects of information security which aims to provide some or all of the services known as *Confidentiality*, *Data Integrity*, *Authentication* and *Non-Repudiation*, which we now briefly introduce[2]:-

- Confidentiality: preventing the unauthorized reception of the message.
- Authentication: verifying the message's origin.
- Data Integrity: establishing that a received message has not been altered.
- Non-Repudiation: This is a service which prevents an *entity* from denying previous commitments or actions.

The encryption are mainly divided into two types; 1) block cipher which split the plaintext into fixed length fragments (blocks) and then operate on each fragment separately to produce a corresponding ciphertext fragments. 2) Stream cipher which operates on the plaintext bit by bit (or character by character) rather than on plaintext block.

The cryptosystem can be one of two kinds; the symmetric encryption and asymmetric encryption. In both cases, it requires the use of some keys to perform the encryption and decryption process. Symmetric key encryption uses the same key for both encrypting and decrypting the data, this kind is also known as secrete key encryption. On the other hand, asymmetric encryption uses two different keys: the first one is the public key used to encrypt the data and the other one is the private key used to decrypt the data, asymmetric encryption is referred to public key encryption [3].

The conceptual difference between symmetric key cryptography/encryption and asymmetric cryptography/encryption are based on how these systems keep a secret. In symmetric key cryptography, the secret must be shared between two persons. In asymmetric cryptography, the secret is personal (unshared); each person creates and keeps his or her own secret [4].

A. The Integer Factorization Problem (IFP):

The integer factorization problem is the following: given a composite number *n* that is the product of two large prime numbers *p* and *q,* find *p* and *q*.

While finding large prime numbers is a relatively easy task, the problem of factoring the product of two such numbers is considered computationally intractable if the primes are carefully selected. Based on the difficulty of this problem, Rabin and Williams developed Rabin cryptosystem [5].

B. The discrete logarithm problem (DLP):

The discrete logarithm problem is the following: given a prime p, a generator α of Z_p, and a non-zero element $\beta \in Z_p$, find the unique integer r, $0 \le r \le p-2$, such that $\beta \equiv \alpha^r \pmod p$. The integer r is called the discrete logarithm of β to the base α. Elgamal scheme is based on discrete logarithm problem [5].

C. Rabin Cryptosystem:

The Rabin cryptosystem is example of an asymmetric cryptosystem. It was created by Michael O. Rabin in 1979. Rabin's work has theoretical importance because it provided the first provable security for public-key cryptosystems. It can be proven that recovering the entire plaintext from the ciphertext has same complexity as factoring large numbers [2].

Rabin's algorithm is stated as follows:

- Each user chooses two primes p and q each equal to 3 modulo 4, and forms the product $n = p \times q$.
- Public key: the number n.
- Private key: the numbers p and q.
- Encryption: to encrypt a message m, form the ciphertext $c = m^2 \pmod{n}$.
- Decryption: given ciphertext c, use the formulas below to calculate the four square roots modulo n of c: m_1, m_2, m_3, and m_4. One of them is the original message m.

II. ELGAMAL CRYPTOSYSTEM

The ElGamal cryptosystem was originally developed in 1985 and is based on difficulty of discrete logarithm problem for finite fields. The ElGamal Cryptosystem consists of three steps [2]:

A. Generating the key:

- Choose a large prime number p, such that $(p - 1) / 2$ is prime, too. N is the number of bits of p.
- Choose the base alpha $< p$.
- Choose the private key $a < p$.
- Compute beta = alpha$^a \pmod{p}$
- Publish p, alpha and beta as public key.

B. Encryption of a message:

- Split the plaintext into blocks of N - 1 bits (fill them up if necessary).
- Choose a secret random number k with $\gcd(k, p - 1) = 1$
- Compute for every block x the ciphertext $e(x, k) = (y_1, y_2)$, where y_1 = alpha$^k \pmod{p}$ and y_2 = beta$^k x \pmod{p}$. y_1 and y_2 are blocks of the length N of ciphertext.

C. Decryption of a message:

- Split the ciphertext in blocks of N bits
- For two successive blocks y_1 and y_2 solve
  $y_1^a x = y_2 \pmod{p}$ for x. Thus
  $d(y_1, y_2) = x = y_2 (y_1^a)^{-1} \pmod{p}$
  is the wanted block of plaintext.

III. OUR SCHEME (MECA)

In this section, we propose an efficient algorithm for enciphering a long plaintext. Our proposed algorithm (Modified ElGamal Cryptosystem Algorithm (MECA)) is based on both ElGamal cryptosystem [2] and Rabin cryptosystem [2].

Our algorithm consists of three steps:

A. Generating the key:

- Choose a large prime number p, such that $(p - 1) / 2$ is prime, too. N is the number of bits of p.
- Choose the base alpha $< p$.
- Choose the private key $a < p$.
- Compute beta = alpha$^a \pmod{p}$
- Generate two large random (and distinct) primes r and s, each roughly the same size and both primes when divided by 4 give remainder 3
- Compute n = r*s.
- Use the extended Euclidean algorithm to find integers A and B satisfying A*r + B*s = 1. Note that A and B can be computed once and for all during the key generation stage.
- Publish p, n, alpha and beta as public key and kept secret A, B, a, r, s as private key.

B. Encryption of a message:

- Split the plaintext into blocks of N - 1 bits (fill them up if necessary).
- Choose a secret random number k with $\gcd(k, p - 1) = 1$. There should be same k for every block of message.
- Compute for every block x the ciphertext $e(x, k) = (y_1, y_2)$, where y_1=alpha$^k \pmod{p}$ and y_2=beta$^k x \pmod{p}$. y_1 and y_2 are blocks of the length N of ciphertext.
- Append ten consecutive five ("5") digits at the end of y_1 and at the end of y_2 then add y_1 with y_2 as m=y_1 + (10 consecutive "5" digits) + y_2 + (10 consecutive "5" digits)
- Compute $c = m^2 \pmod{n}$.
- c is the cipher text message.

C. Decryption of a message:

To recover cipher text m from c, one should do the following:

- Compute $t_1 = c^{(r+1)/4} \bmod r$.
- Compute $t_2 = c^{(s+1)4} \bmod s$.
- Compute $x = (A*r*t_2 + B*s*t_1) \bmod n$.
- Compute $y = (A*r*t_2 - B*s*t_1) \bmod n$.
- Using x and y, we get four messages m_1, m_2, m_3 and m_4 as follows:
- $m_1 = x$, $m_2 = -x$, $m_3 = y$, and $m_4 = -y$.
- One of the four is the original message m and we can identify the original message by using the redundant consecutive bits which is appended at the last of original message m.
- Now by using the message m, we find the plain text as follows

- We divide the message m into two sub blocks y1 and y2 by using the redundant consecutive bits added at the end of y1.
- For two successive blocks y1 and y2 solve Compute $x = y_2 (y_1^a)^{-1} (\text{mod } p)$.
- x is the wanted block of plaintext.

IV. COMPARISION BETWEEN ELGAMAL CRYPTOSYSTEM AND MECA

A. Simulation Results:

For the simulation purpose, MECA cryptosystem is implemented as a user-friendly GUI. This GUI application is implemented using Java BigInteger library functions [6]. In this application, one can either enter the prime numbers or can specify the bit length of the prime numbers to generate automatically. BigInteger library provides operations for modular arithmetic, GCD calculation, primarily testing, prime generation, bit manipulation, and a few other miscellaneous operations. The simulation of the algorithm, implemented in JAVA [6], running on a 2.8 GHz P-IV Processor and 512 MB RAM, using a 1000 characters long message for encryption/decryption. In this section, we investigate the issues of complexity, efficiency and reliability by running the program with different sets of data. Moreover, comparison will be done between these different algorithms given the same data as input.

The comparison between both algorithms, are analyzed by changing only one parameter at a time while keeping the other parameter unchanged. The algorithm depends mainly on the *prime number* whose value depends on the number of bits used to generate this prime. It also depends on the *private key*, the *input message* and the chunk size (*cut length*) of the block message [1]. The key generation, encryption, decryption time is depends on the speed of the processor and the RAM. We can show the results through the graph on the basis of the reading in the Table 1 for the Module Size (2048 bit). Actually the module size is varying from 256 bit to 2048 bit, but we are showing here results only for 2048 bit.

a) Changing the length of the prime number (p) length:

The value of the prime number (p), plays an important role in forming the values of the public key (p, alpha, beta) and hence the encrypted message pair (y1, y2). This prime number is a function of the number of bits used to generate it. Changing this bit number is directly reflected on the length of the other parameter.

The length of the prime number (p) is directly proportional to its generating number of bits since the maximum value of (p).The length of encrypted message (y1, y2) also increase as a function of the prime number (p). As a result, increasing the n-bit length provides more security. On other hand by increasing the n-bit length increases the values of key generation time, encryption time and decryption time as illustrated by Figure 1 and 2.

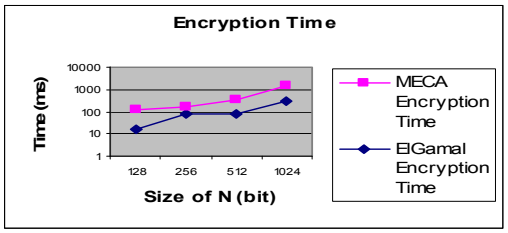

Figure1: Modulus (N) Size (bit) v/s Encryption time (ms), taking size of Private Key (a) 32 bit and Chunk Size 64 bit.

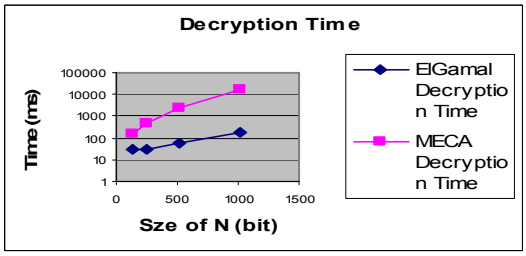

Figure2: Modulus (N) Size (bit) v/s Decryption time (ms), taking size of Private Key (a) 32 bit and Chunk Size 64 bit

b) Changing the private key length:

To study the performance of private key, the value of (bit), which is used to generate the prime number (p) and chunk size is fixed then the value of private key is changed. The Figure 3 and 4 illustrate the speed of encryption and decryption with variable private key.

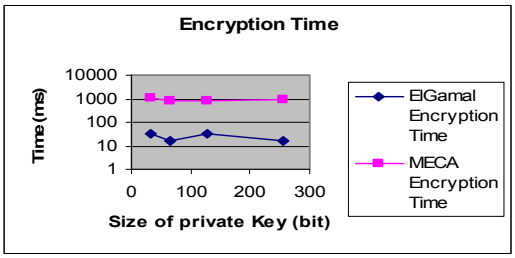

Figure3: Private Key (a) size v/s Encryption time, taking Modulus (N) size 2048 bit and chunk size 64 bit.

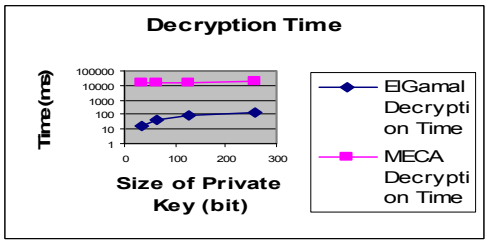

Figure4: Private Key (a) size v/s Decryption time, taking Modulus (N) size 2048 bit and chunk size 64 bit. q1

Module (N) size (bit)	p size (bit)	Private Key "a" size (bit)	Chunk Size (bit)	ELGAMAL CRYPTOSYSTEM				MECA			
				key generation time (ms)	Encryuption time (ms)	Decryption time (ms)	Total Execution time(ms)	Key Generation time (ms)	Encryption time (ms)	Decrytpion time (ms)	Total Execution time(ms)
2048	1024	32	64	15	313	187	500	17531	1141	15609	16750
2048	1024	32	128	15	63	94	157	17531	141	6031	6172
2048	1024	32	256	15	46	47	93	17531	47	2766	2813
2048	1024	32	512	15	31	16	47	17531	16	1344	1360
2048	1024	64	64	16	250	359	609	10078	797	16062	16859
2048	1024	64	128	16	63	172	235	10078	94	6140	6234
2048	1024	64	256	16	32	93	125	10078	94	2813	2907
2048	1024	64	512	16	16	47	63	10078	15	1407	1422
2048	1024	128	64	16	281	687	968	15625	797	16219	17016
2048	1024	128	128	16	63	375	438	15625	141	6297	6438
2048	1024	128	256	16	31	157	188	15625	47	2860	2907
2048	1024	128	512	16	31	78	109	15625	15	1391	1406
2048	1024	256	64	16	250	1265	1515	108859	875	17406	18281
2048	1024	256	128	16	63	672	735	108859	125	6735	6860
2048	1024	256	256	16	31	328	359	108859	47	3031	3078
2048	1024	256	512	16	16	156	172	108859	16	1531	1547

Table 1 : Effect of changing the Chunk Size and Private Key with the constant module size(N) 2048 bit

c) Changing the cut length (or chunk size):

To study the performance of chunk size the value of bit and private key kept unchanged and change the value of chunk size. The Figure 5 and 6 illustrate the speed of encryption and decryption with variable chunk size.

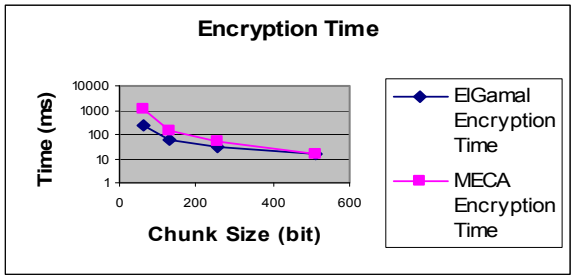

Figure5: Chunk size v/s Encryption time, taking size of Modulus (N) 2048 bit and size of Private key 32 bit.

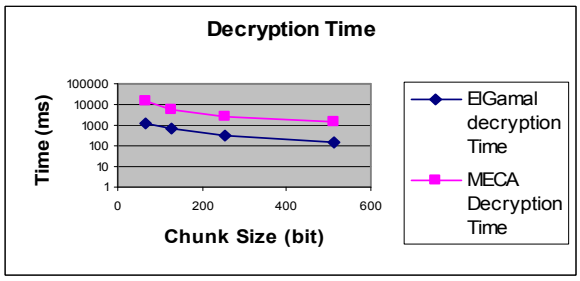

Figure6: Chunk size v/s Decryption time, taking size of Modulus (N) 2048 bit and size of Private key 32 bit.

B. Complexity analysis of MECA cryptosystem:-

Let n be the module used in the Elgamal cryptosystem. Elgamal encryption, decryption, each take 2n time scalar point multiplication and 2n times multiplication operation. Rabin encryption takes one modular multiplication and one modular squaring. Rabin decryption is slower than encryption; it means the complexity of Rabin system is as factoring a large number n into its two prime factors p and q. So in this way, computational complexity of MECA cryptosystem depends on 2n times scalar point multiplication and factoring of two large prime numbers p and q. The simulation results of both the algorithms shows that the execution time of MECA is 2 times more than ElGamal.

V. SECURITY ANALYSIS

Two possible approaches to attack the MECA / ElGamal algorithms are as follows [6, 7]:

- *Brute force:* This involves trying all possible private keys.
- *Mathematical attacks:* These are based on factoring the product of large primes such as factor n into its prime factors p and q respectively.

Estimated resources needed to factor a number within one year are as follows [8].

Size of number (Bits)	PCs	Memory
430	1	128 MB
760	215,000	4 GB

1020	342×10^6	170 GB
1620	1.6×10^{15}	120 TB

Table 2: Recourses needed to factor a number within one year

The security of our scheme is based on both Integer Factorization Problem (IFP) [5] and Discrete Logarithm Problem (DLP) [5], it is very difficult for an illegal user to compute the secret key 'a' from the equation beta = $alpha^a (\text{mod } p)$. It is also difficult for an intruder to obtain the system-generated random numbers r and s directly from equation A*r + B*s = 1 of MECA Algorithm. The difficulty relies on the complexity of computing discrete logarithm over finte fields. In *integer factorization problem (IFP)* finding large prime numbers is a relatively easy task, the problem of factoring the product of two such numbers is considered computationally intractable if the primes are carefully selected. For long-term security, 1024-bit or larger module *p* should be used in *discrete logarithm problem (DLP)* and *integer factorization problem (IFP)*.

The discrete logarithm problem can be solved by *naïve*[2], *baby-step/giant-step*[2] and *Pollard's rho* algorithm[2]. Because of these algorithms, the security of Elgamal cryptosystem is broken by the *Low-Modulus*[4] and *Known-Plaintext attack*[4]. As the MECA cryptosystem uses dual modulus, so for breaking this one have to factor both the modulus. That's why our cryptosystem provides the far better security against the *naïve, baby-step/giant-step* and *Pollard's rho* algorithm and attacks on Elgamal cryptosystem.

VI. CONCLUSIONS

The ElGamal cryptosystem is secured against passive attack but not against adaptive attack. Efficient algorithms for the discrete logarithm problem would render the ElGamal Cryptosystem insecure. So to improve the security, this paper presents a **M**odified ***E****lGamal* ***C****ryptosystem* ***A****lgorithm* (MECA). MECA algorithm is based on the combination of factorization of large number and the discrete logarithm problem. So if one want to break the MECA, one have to factor large numbers into its prime factors and also have to solve discrete logarithms problems. This paper also shows comparisons among MECA and ElGamal cryptosystems in terms of security & performance. The disadvantage of new cryptosystem is that, unlike Elgamal it can not be used for authentication as it is based on the one way function. Another disadvantage is the slow down of execution process as compare to Elgamal. But it is clear from the simulation results that it is more secure than the Elgamal algorithm and our cryptosystem provides the far better security against the *naïve, baby-step/giant-step* and *Pollard's rho* algorithm and attacks on Elgamal cryptosystem.

.

REFERENCES

[1] Allam Mousa,"Security and Performance of ElGamal Encryption Parameter," Journal of Applied Sciences 5, Asian Network foe Scientific Information, 883-886, 2005.

[2] Alfred J Menezes, Paul C van Oorschot, Scot A Vanstone, *Handbook of AppliedCryptography*, CRC Press, 1997.

[3] Wenbo Mano,"Modern Cryptography Theory and Practice," Prentice Hall.

[4] Behrouz A. Forouzan, Debdeep Mukhopadhyay, "Cryptography and Network Security" 2nd Edition, TMH.

[5] http://www.certicom.com, "The Elliptic Curve Cryptosystem," *September 1997,* (dated 02-04-2010)

[6] Neal R. Wagner, "The Laws of Cryptography with Java Code", *Technical Report, 2003*, pages 78-112

[7] William Stallings, "Cryptography and Network Security", *ISBN 81-7758-011-6*, Pearson Education, Third Edition, pages 42-62,121-144,253-297.

[8] CHUK, Chinese university (2009), "RSA Algorithm security and Complexity", Retrieved from *http://www.cse.cuhk.edu.hk/~phwl/mt/public/archives/old/ceg50 10/rsa.pdf* (dated 04-04-2011)

[9] J H Moore, *Protocol failures in Cryptosystems*, Contemporary Cryptology, The science of Information Integrity, Ed. G J Simmons, 541-558, IEEE Press 1992.

[10] H E Rose, "*A Course in Number Theory, Second Addition"*, Oxford Science Publications, 1994.