Islamic Azad University

Journal of

Optoelectronical Nanostructures

Summer 2018 / Vol. 3, No. 3

Comparison of two Public Key Cryptosystems

Mahnaz Mohammadi[1], Alireza Zolghadr[*,2], Mohammad Ali Purmina[1]

[1] Department of Electrical and Electronic Eng., Science and Research Branch, Islamic Azad University, Tehran, Iran

[2] Faculty of Computer and Electrical Eng., Department of Communication and Electronics, Shiraz University, Shiraz, Iran

(Received 10 Jun. 2018; Revised 11 Jul. 2018; Accepted 26 Aug. 2018; Published 15 Sep. 2018)

Abstract: Since the time public-key cryptography was introduced by Diffie and Hellman in 1976, numerous public-key algorithms have been proposed. Some of these algorithms are insecure and the others that seem secure, many are impractical, either they have too large keys or the cipher text they produce is much longer than the plaintext. This paper focuses on efficient implementation and analysis of two most popular of these algorithms, RSA and ElGamal for key generation and the encryption scheme (encryption/decryption operation). RSA relies on the difficulty of prime factorization of a very large number, and the hardness of ElGamal algorithm is essentially equivalent to the hardness of finding discrete logarithm modulo a large prime. These two systems are compared to each other from different parameters points of view such as performance, security, speed and applications. To have a good comparison and also to have a good level of security correspond to users need the systems implemented are designed flexibly in terms of the key size.

Keywords: Cryptography, Public Key Cryptosystems, RSA, ElGamal.

1. INTRODUCTION

The security is a necessary for the network to do all the works related to ATM cards, computer passwords and electronic commerce. To achieve network security, many cryptography techniques have been presented [1-7]. Cryptography is the technique for creating secret codes. It is used to transform the message into a ciphertext such that an adversary who overhears the ciphertext cannot determine the message sent.

There are two types of cryptographic algorithms, first is the asymmetric key or public key cryptography and second is the symmetric key or private key cryptography [8]. The legitimate receiver possess a secret key that allows him/her to reverse the encryption transformation and retrieve the message. The

[*] Corresponding author. E-mail: **zolghadr@shirazu.ac.ir**

sender may have used the same key to encrypt the message (with the symmetric scheme) or used a different but related key (with the public-key scheme).

Modern cryptography uses a system of keys to solve the problems of conventional algorithms. This key might be one among several possible in a large key-space. Public-key cryptosystem uses different keys for encryption and decryption and all the security is in the key rather than the algorithm. This means that the algorithm can be safely published. Ideally, the decryption key cannot be calculated from the encryption key in any reasonable amount of time. The main reason they have been named "public-key" is that the encryption key can be made public. Anyone can use an encryption key to encrypt a message but only the person with the corresponding decryption key (private key) in the system can decrypt the message.

This paper focuses on efficient implementation and analysis of two public-key algorithms, RSA and ElGamal which will be described in detail in the following sections. The rest of the paper is structured as follows. In section 2, key generation and encryption scheme (encryption/decryption) of RSA is presented. Section 3 includes the same material as section 2 but for ElGamal cryptosystem. Test results are presented in section 4. The cryptosystems are compared in section 5. A brief conclusion is presented in section 6.

2. RSA PUBLIC-KEY CRYPTOSYSTEM

RSA was invented in 1978 by Ron Rivest, Adi Shamir and Leonard Adleman [9]. It is the most commonly used and currently most important public-key algorithm which can be used for both encryption and signing. RSA cryptosystem involves exponentiation modulo an integer number n that is the product of two large primes p and q. The security of the system is based on the difficulty of factoring large integers in terms of its key size and the length of the modulus n in bits which is said to be the key size [1, 10, 11].

2.1. RSA Key Generation Algorithm

In the RSA system each entity who wishes to receive encrypted message performs the following steps:

1. Selects two large primes p and q at random, each for example *100* digits long (it is up to the user to choose the key size).
2. Calculates the modulus of the system $n = p \cdot q$.
3. Selects at random a number e such that $gcd(e, \varphi(n)) = 1$ where $\varphi(n) = (p-1) \cdot (q-1)$ is the "*Euler's phi function of n"* [11] (e is usually small to produce large size d for better security).
4. Solves the equation $d.e = 1 \pmod{\varphi(n)}$ in the congruence for the unique d.
5. Publish the public key $E = (e, n)$.
6. Keeps secure the private key $D = d$.

2.2. RSA Encryption Scheme

(I) Encryption:
A sender can encrypt a message *M* (expressed as some large integer belongs to (0,n-1)) to form the ciphertext *C* using the public encryption key *(e, n)* of the receiver by calculating $C = M^e \pmod n$.

(II) Decryption:
The receiver can then decipher this using his/her own private decryption key by calculating $M = C^d \pmod n$.

3. ELGAMAL PUBLIC-KEY CRYPTOSYSTEM

ElGamal encryption system is an asymmetric key encryption algorithm for public-key cryptography which is based on the Diffie–Hellman key exchange. It was described by Taher Elgamal in 1984 [11]. ElGamal encryption can be defined over any cyclic group *G*. Its security depends upon the difficulty of a certain problem in *G* related to computing discrete logarithms. The ElGamal is a public key algorithm, which can be used for both digital signature as well as encryption. Its security is based on the difficulty of computing discrete logarithms in a finite field.

3.1. Elgamal Key Generation Algorithm

The following steps must be done to get the key pairs of each entity:
1. Generate a large random prime *p* and a generator (alpha) of the multiplicative group Z*p
2. Select a random integer a, 1<= a <=p-2 and compute $\text{alpha}^a \pmod p$.
3. Publish (p, alpha, alpha^a) as the public key.
4. Keep secure a as the private key.

In this algorithm, the problem of finding generators of group Z^*_p (where p is a very long prime) which is the basic step of key generation of ElGamal public-key system, could be very time-consuming. According to A.Menezes.et.al [10] in general no polynomial-time algorithm is known for finding generators or even for testing whether an element is a generator of a finite field F_q where q is a large prime. There are some methods to overcome this problem (finding a generator of a cyclic group of order *n* having the prime factorization of *n*): using one of the integer factorization algorithms to factor *n-1* which is not a polynomial time algorithm.

Another alternative approach is to generate a safe prime p (p = 2q+1 where q is a large prime) [12]. This method is more efficient (we will have the prime factors of n-1, which are 2 and q, without using factorization algorithm) and we usually have flexibility of selecting the prime p in cryptographic applications,

however it may not be optimal in terms of the speed since the number of safe primes of special size is much less than the number of primes themselves.

The second option has been selected for this research, because of providing a better security and also ease of implementation (algorithm (I)). Besides this way, each element of Z^*_p is a generator of the group with the probability of 1/2.

Algorithm (I), producing a safe prime:
Input: the bit length of the required prime
Output: a k-bit safe prime.

1. do the following:
   1.1 $q \leftarrow$ random prime of (k-1)-bit size (using prime generation algorithm).
   1.2 $p \leftarrow 2. q + 1$ and test whether p is prime (using the primality test algorithms)
   until p is prime.
2. return *(p)*.

3.2. Elgamal Encryption Scheme

(I) Encryption:
Suppose Bob needs to send the message M to Alice, so he has to perform the encryption operation algorithm as follows:

1. Get Alice's public key (p, α, α^a)
2. Represent the message M as an integer m ($m \varepsilon Z_p$)
3. Select a random integer k, $1 \leq k \leq p-2$
4. Compute $\gamma = \alpha^k \bmod p$ and $\delta = m * (\alpha^a)^k \bmod p$
5. Send the ciphertext $C = (\gamma, \delta)$ to Alice

(II) Decryption:
To recover the plaintext m from C Alice has to do the decryption operation using her own private key:

1. Compute $\gamma^{p-1-a} \bmod p$ (a is her private key)
2. Recover m by computing $(\gamma^{-a}) * \delta \bmod p$

This correctly recovered m because:
$(\gamma^{-a}) * \delta \equiv \alpha^{-ak} m \alpha^{ak} \equiv m \pmod p$.

4. TEST RESULTS

4.1. Key Generation Time

Figure 1 shows the experimental timing function for RSA key generation. It is considerably fast due to efficient implementation of prime generation in this research. Results averaged over 20 different (random) data.

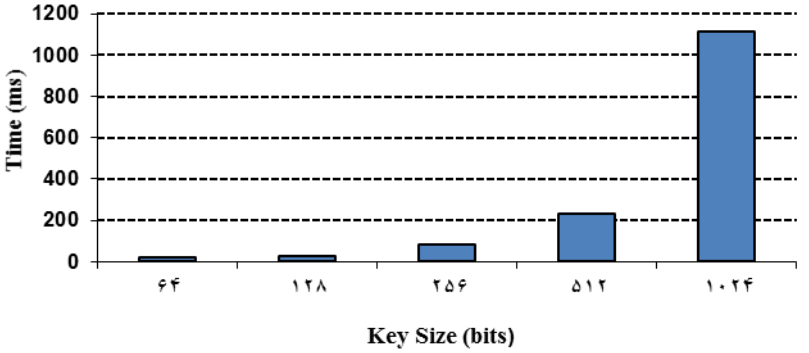

Fig. 1. Timing curve for key generation in RSA system

In RSA, the *generation of the prime numbers* is a crucial sub-process, which requires generating random numbers and testing for primality, a highly probabilistic procedure. Consequently, the time of execution for RSA key generation is not always the same even for the same key length; occasionally it can be very long.

Figure 2 shows the experimental timing function for ElGamal key generation. Results averaged over 20 different data for each different key size.

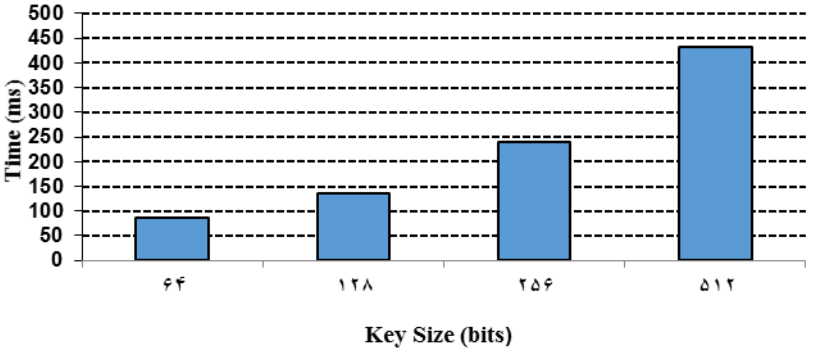

Fig. 2. Timing curve for key generation in ElGamal system

As it is seen from figures 1 and 2, key pair generation time grows linearly in ElGamal system with key sizes, while that of RSA grows exponentially.

4.2. Encryption/Decryption Results

Figures 3 and 4 demonstrate the encryption and decryption of a file of *2000* byte using RSA system. Encryption time is less than decryption because of the choice of encryption exponent which is much smaller than decryption exponent in the implementation to provide a better security against computation of decryption exponent easily.

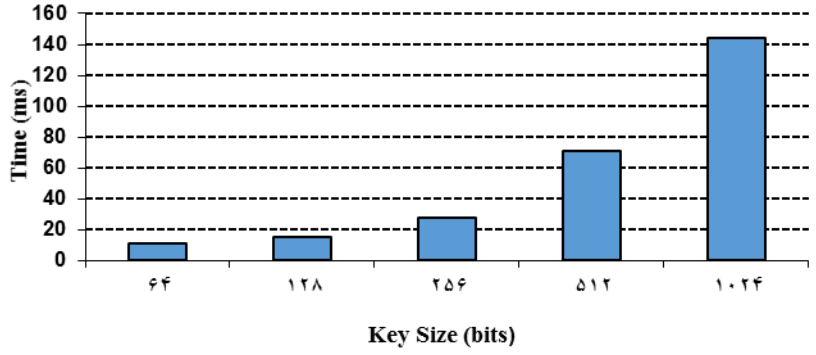

Fig. 3. Timing curve for RSA-encryption

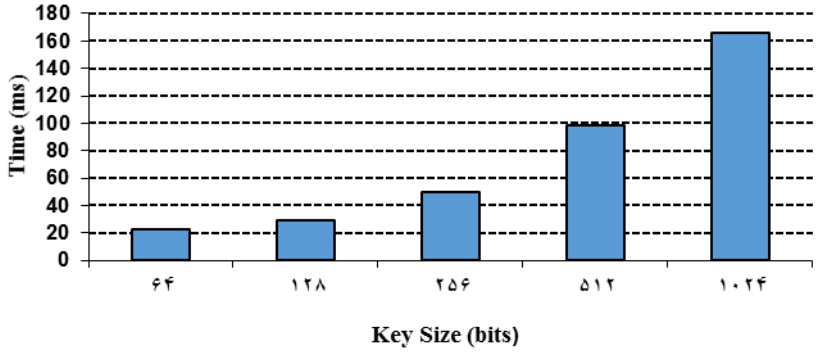

Fig. 4. Timing curve for RSA-decryption

Another important point about these results is the speed-up of the encryption and decryption operation using the efficient implementation of modular exponentiation (Montgomery modular exponentiation [13]) and optimization method for decryption operation using the Chinese remainder theorem [10].

Using these issues makes a great difference in the implementation of RSA encryption and decryption operations.
Figures 5 and 6 demonstrate the encryption and decryption of a file of 2000 byte using ElGamal system.

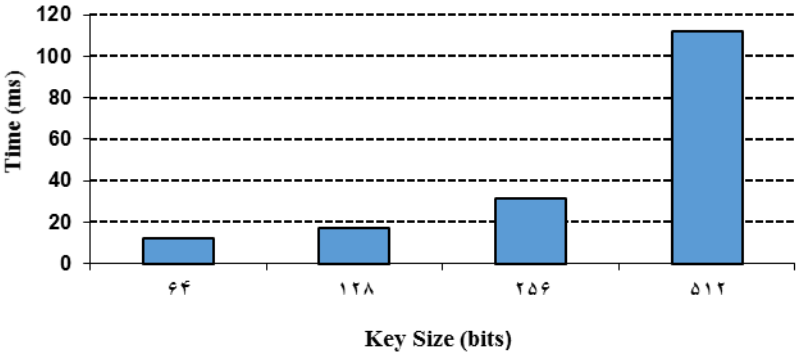

Fig. 5. Timing curve for ElGamal-encryption

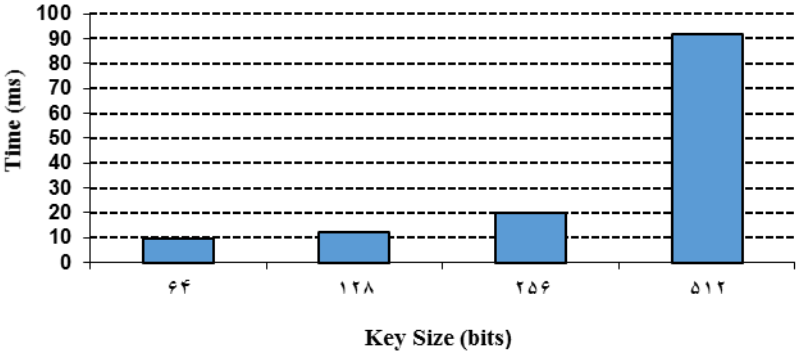

Fig. 6. Timing curve for ElGamal-decryption

5. COMPARISON OF THE TWO SYSTEMS

Analysis based on the best available algorithms for both factoring and discrete logarithm shows the asymptotic running time of the best discrete logarithm algorithm is approximately the same as the best general purpose factoring algorithm.

When choosing an asymmetric key size we are considered by two principles:

(I). Security: One of the key factors of the strength of cryptosystems is the size of the public / private key pairs.

(II). Speed: The longer the key the slower the public key operations and key generations.

For the majority of users, the main factor in the selection of public-key size is the security, based on best attacks known. Speed is very rarely a determining factor.

5.1. Security

In all public-key systems, a passive adversary can always mount a chosen ciphertext attack, where the adversary has a quantity of plaintext and corresponding ciphertexts, on a public-key encryption scheme. A stronger attack is a chosen-ciphertext attack, where an adversary selects the ciphertext of its choice and then obtains some information about corresponding plaintexts. RSA is very vulnerable to this type of attacks. [sch17], [men01] have great discussions about different types of attacks against RSA and other public-key cryptosystems.

In RSA system, the choice of p and q is very important, they should be selected so that factoring n = p.q is computationally infeasible. [sch17] shows this difficulty in terms of length of n. Primes p,q must have the same size (about half of *n*) because if for example, p is a small prime it will be very easy to find and then n will be easily factorized, for this reason in the implementation p and q are prime numbers derived from some random strings of the same size.

Theoretically, it is desirable to select strong primes *p* and *q* for RSA systems [12] but more recently fast factoring algorithms have been discovered that work very well against strong primes [14]. At the moment prime length is far more important than the structure especially for this research to collect different data and analyze the experimental results, *p* and *q* can be random primes of any length and due to efficient implementation of prime generation and modular arithmetic they can be produced very quickly as Figure 1 shows.

The security of ElGamal system which is equal to the security of Diffie-Hellman problem relies on the difficulty of computing discrete logarithm which is believed to be computationally equivalent to factoring large numbers [15]. It is sensitive to the choice of the strong prime p and the generator of the group Z*p, the size of the secret exponent is also important for its security.

Another important factor in this system is the need for a good random number generator, since it utilizes randomization in the encryption process and it is critical to have different independent random numbers for *k* in the encryption algorithm. Because for instance assume the same *k* for encryption of two messages m_1 and m_2 and the results of encryption are (γ_1, δ_1), (γ_2, δ_2). Then $\delta_1 / \delta_2 = m_1 . (\alpha^{a})^{k} / m2 . (\alpha^{a})^{k} = m_1 / m_2$, means knowing one of the messages the other can be computed easily.

Note that the basic ElGamal scheme just like RSA is completely insecure against adaptive chosen ciphertext attack [16].

5.2. Speed

Looking at other factors in making a choice of cryptosystems, RSA offers very fast key generation (and practically signature verification).

ElGamal system may take longer time for the key generation but offers more long-term security. The main disadvantage of ElGamal is the need for randomness and its slower speed for key generation. The expensive key component to generate is the public prime modulus p in order to be able to produce the generator α for the group Z^*_p and to have a reasonable level of security. But on the other hand, a group of keys (users) can share a common public modulus with no negative security implication other than the key presents a bigger target for pre-computation [17, 18].

TABLE I
Comparison of encryption operation for RSA and ElGamal systems

Key Size	RSA	ElGamal
64	11.21	12.23
128	15.35	17.37
256	27.75	31.24
512	71.05	112.29

TABLE II
Comparison of decryption operation for RSA and ElGamal systems

Key Size	RSA	ElGamal
64	22.75	9.82
128	29.05	12.31
256	50.05	19.74
512	98.47	91.86

Tables I and II from experimental data of this research (in ms) show that the decryption operation of ElGamal system is faster than RSA, so for the applications in which messages are deciphered many times since they are ciphered once, ElGamal is considered better in terms of performance.

5.3. Size of the Encrypted Data Files

The size of the encrypted data for RSA depends on the size of the key and the size of input data.

Results obtained in this research showed that this size depends on the modulus used and the size of input data file.

In ElGamal system, this size depends on the key size and the input data size. Also, the maximum size of data that can be encrypted in one step depends on the key size.

Another potential disadvantage of ElGamal system is the message expansion by a factor of two takes place during the encryption. However such message expansion is generally unimportant if the cryptosystem is used only for exchange of the secret files.

6. CONCLUSION

Public-key encryption can be used to eliminate problems involved with conventional encryption. However, it has not managed to be as widely accepted as conventional encryption because it introduces a lot of overheads. A hybrid of symmetric and asymmetric methods is used in most real applications.

In most real-world applications public-key systems mainly are used to encrypt the random session keys used with symmetric algorithms so performance hit is negligible.

In the comparison of the two public-key cryptosystems, RSA and ElGamal, it should be said that RSA is more efficient for encryption but, less efficient for decryption than ELGamal. Also, the ElGamal system is not so mature, as mathematicians believe that enough research has not yet been done in the mathematics behind it.

Research done about RSA is very vast and all details about it have been discussed. At the moment using RSA is so vast. However, the future of ElGamal looks bright.

RSA is more secure than ELGamal and is widely accepted and used but, ELGamal is new and not very popular in the market.

Even if the factorization problem is solved in future, in a polynomial time, by increasing the size of the primes used in the key generation of RSA system, it can be kept secure for years.

RSA is faster in key generation, but ElGamal system can provide the same level of security as RSA using shorter keys, thereby reducing processing overhead. Also, it should be considered that implementation of ElGamal is much more complicated than RSA.

In general, there is no single answer to the question of which public-key technology is best. The answer depends upon the setting (e.g. certificate based systems, secure email, online communications and wireless communications) in which public key methods are used.

REFERENCES

[1] S. William, *Cryptography and Network Security Principles and Practice*, 7th ed., Pearson Education, Prentice Hall, USA, 2017.

[2] JS. Coron, T. Holenstein, R. Künzler, et al., *How to Build an Ideal Cipher: The Indifferentiability of the Feistel Construction*, Journal of Cryptology, 29 (1) (2016) 61–114.

[3] I. Komargodski, G. Segev, E. Yogev, *Functional Encryption for Randomized Functionalities in the Private-Key Setting from Minimal Assumptions*, Journal of Cryptology, 31 (1) (2018, Jan.) 60–100.

[4] I. Mironov, O. Pandey, O. Reingold, Gil Segev, *Incremental Deterministic Public-Key Encryption*, Journal of Cryptology, 31 (1) (2018, Jan.) 134–161.

[5] P. K. Panda, S. Chattopadhyay, *A hybrid security algorithm for RSA cryptosystem,* in Proc. ICACCS, (2017) 1-6.

[6] S. B. Sadkhan, F. H. Abdulraheem, *A proposed ANFIS evaluator for RSA cryptosystem used in cloud networking,* in Proc. ICCIT, (2017) 48–51.

[7] F. Mo, Y. C. Hsu, H. H. Chang, S. C. Pan, J. J. Yan, T. L. Liao, *Design of an Improved RSA Cryptosystem Based on Synchronization of Discrete Chaotic Systems,* in Proc. ISAI, (2016) 9-13.

[8] Y. Jitarwal, P. K. Mangal, S. K. Suman, *Enhancement of ElGamal Digital Signature Based on RSA & Symmetric Key*, International Journal of Advanced Research in Computer Science and Software Engineering, 5 (5) (2015, May) 693-696.

[9] T. H. Cormen, C. E. Lieserson, R. L. Rivest, *Introduction to Algorithms*, MIT Press, 2009.

[10] A. J. Menezes, P.van Oorschot, S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press LLC, 2001.

[11] B. Schneier, *Applied cryptography: protocols, algorithms, and source code in C*, John Wiley & Sons, 2017.

[12] P. Ribenboim, *The Book of Prime Number Records,* 2nd ed., London, Springer-Verlag, 2001.

[13] T. Adiono, H. Ega, H. Kasan, S. Fuada, S. Harimurti, *Full custom design of adaptable montgomery modular multiplier for asymmetric RSA cryptosystem*, in Proc. ISPACS, (2017) 910–914.

[14] R. Patidar, R. Bhartiya, *Implementation of Modified RSA cryptosystem based on offline storage and prime number*, International Journal of Computing and Technology, 1 (2), (2014, March) 205-209.

[15] J. H. Seo, , *Short Signatures from Diffie–Hellman: Realizing Almost Compact Public Key*, Journal of Cryptology, 30 (3) (2017, July) 735–759.

[16] S. Hohenberger, B. Waters, *Realizing hash-and-sign signatures under standard assumptions*, in Proc. AICTACT, (2009) 333–350.

[17] A. Ara, M. Al-Rodhaan, Y. Tian, A. Al-Dhelaan. *A Secure Privacy-Preserving Data Aggregation Scheme Based on Bilinear ElGamal Cryptosystem for Remote Health Monitoring Systems.* IEEE Journals & Magazines, 5 (2017) 12601-12617.

[18] Y. Murakami, M. Kasahara, *Hybrid inter-organization cryptosystem using ElGamal cryptosystem,* in Proc. IEEE-ICCE, (2015) 378-379.