A COMPARITIVE STUDY OF ELGAMAL BASED CRYPTOGRAPHIC ALGORITHMS

Ramzi A. Haraty, Hadi Otrok
Lebanese American University P.O.Box 13-5053 Chouran, Beirut, Lebanon 1102 2801
Email: rharaty@lau.edu.lb, hadiotrok@hotmail.com

A. N. El-Kassar
Mathmatics Department, Beirut Arab University, Beirut, Lebanon
Email: ak1@bau.edu.lb

Keywords: ElGamal cryptosystem, testing and evaluation

Abstract: In 1985 a powerful and practical public-key scheme was produced by ElGamal; his work was applied using large prime integers. El-Kassar et al. and El-Kassar and Haraty modified the ElGamal public-key encryption scheme from the domain of natural integers, Z, to two principal ideal domains, namely the domain of Gaussian integers, Z[i], and the domain of the rings of polynomials over finite fields, F[x], by extending the arithmetic needed for the modifications to these domains. In this work we implement the classical and modified ElGamal cryptosystem to compare and to test their functionality, reliability and security. To test the security of the algorithms we use a famous attack algorithm called Baby-Step-Giant algorithm which works in the domain of natural integers. We enhance the Baby-Step-Giant algorithm to work with the modified ElGamal cryptosystems.

1 INTRODUCTION

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

As we move further into an information society, the technological means for global surveillance of millions of individual people are becoming available to major governments. Cryptography has become one of the main tools for privacy, trust, access control, electronic payments, corporate security, and countless other fields.

Perhaps the most striking development in the history of cryptography came in 1976 when Diffie and Hellman published New Directions in Cryptography (Diffie, 1978) . Their work introduced the concept of public-key cryptography and provided a new method for key exchange. This method is based on the intractability of discrete logarithm problems. Although the authors had no practical realization of a public-key encryption scheme at the time, the idea was clear and it generated extensive interests and activities in the world of cryptography. One of the powerful and practical public-key schemes was produced by ElGamal in 1985 (ElGamal, 1985).

El-Kassar (El Kassar, 2001) and Haraty (Haraty 2003) modified the ElGamal public-key encryption schemes from the domain of natural integers, Z, to two principal ideal domains, namely the domain of Gaussian integers, Z[i], and the domain of the rings of polynomials over finite fields, F[x], by extending the arithmetic needed for the modifications to these domains.

In this paper, we compare and evaluate the classical and modified ElGamal algorithms by implementing and running them on a computer. We investigate the issues of complexity, efficiency and reliability by running the programs with different sets of data. Moreover, comparisons will be done between these different algorithms given the same data as input. 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 different classical and modified cryptographic algorithms.

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

2 CLASSICAL AND MODIFIED ELGAMAL PUBLIC-KEY CRYPTOSYSTEM

The ElGamal encryption scheme is typically described in the setting of the multiplicative group Z_p^*. But, it can be easily generalized to work in any finite cyclic group G. As with the classical ElGamal encryption, the security of the generalized ElGamal encryption scheme is based on the intractability of the discrete logarithm problem in the group G. The group G should be carefully chosen so that the group operations in G should be relatively easy to apply for efficiency and the discrete logarithm problem in G should be computationally infeasible for the security of the protocol that uses the ElGamal public-key cryptosystem.

Menezes (Menezes, 1997) showed that the groups that appear to meet the above criteria of which the first three have received the most attention are the multiplicative group Z_p^* of the integers modulo a prime p, the multiplicative group $F_{2^m}^*$ of the finite field F_2^m of characteristic two, the group of points on an elliptic curve over a finite field, the multiplicative group F_q^* of the finite field F_q, where $q=p^m$, p is a prime, the group of units Z_n^*, where n is a composite integer, the Jacobean of a hyper elliptic curve defined over a finite field, and the class of an imaginary number field.

For any of the above cases used to generalize ElGamal public-key scheme, the following procedures are followed: To generate the public-key, entity A should select an appropriate cyclic group G of order n, with generator α. Assuming that G is written multiplicatively, a random integer a, $1 \le a \le n-1$ is selected and the group element α^a is computed. A′s public-key is (α,α^a), together with a description of how to multiply elements in G. A′s private-key is a. To encrypt a message m in the cyclic group G, entity B should obtain A′s authentic public-key (α,α^a), then select a random integer k, $1 \le k \le n-1$ and compute $\gamma=\alpha^k$ and $\delta=m.(\alpha^a)^k$. Finally, B sends the ciphertext $c=(\gamma, \delta)$ to entity A. To recover the plaintext m from c, entity A should use the private-key a to compute γ^a and then compute γ^a, the recovered message m is obtained by computing $(\gamma^a.\delta)$. The algorithm for the case Z_p and Z_n can be for (Menezes, 1997).

The following algorithms show the functionality of the ElGamal cryptosystem:
Algorithm 1: (Key generation for ElGamal public-key encryption).

1- Generate a large random prime p and generator θ of Z_p^*.
2- Select a random integer a, $1 \le a \le p-2$, and compute $\theta^a \pmod p$.
3- A's public key is (p, θ, θ^a); A's private key is a.

The following algorithm shows how entity B encrypts a message m for A.
Algorithm 2: (ElGamal public-key Encryption). B should do the following:

1- Obtain A's authentic public key (p, θ, θ^a).
2- Represent the message as an integer m in the range {0, 1,....., p-1}.
3- Select a random integer k, $2 \le k \le p-2$.
4- Compute $\gamma=\theta^k \pmod p$ and
$\delta \equiv m.(\theta^a)^k \pmod p$
5- Send the ciphertext $c=(\gamma, \delta)$ to A.

The following algorithm shows how entity A decrypts a message c from B.
Algorithm 3: (ElGamal public-key decryption). A should do the following:

1- Use the private key a to compute γ^{p-1-a} (mod p)
(Note: $\gamma^{p-1-a}=\gamma^{-a}=\theta^{-ak}$).
2- Recover the message m by computing $\gamma^{-a}.\delta$ (mod p).

Example 1: In order to generate the public-key, entity A selects an odd prime p=359 and finds a generator α=124 of Z_{359}^{*}. Then, A chooses the private-key a=292 and computes

$$124^{292}\equiv 205=\alpha^{a} \pmod{359}.$$

Therefore, A's public-key is

$$(p=359,\ \alpha=124,\ \alpha^{a}=205)$$

and A's private-key is a=292.
To encrypt the message m=101chosen from Z_{359}, B selects a random integer k=247 and computes

$$124^{247}\equiv 291=\gamma \pmod{359}$$

and

$$101.205^{247}\equiv 288=\delta \pmod{359}.$$

Then, B sends (γ=291, δ=288) to A. Note that B has 359 different values for m to choose from Z_{359}. Finally, A computes $\gamma^{p-1-a}=291^{66}\equiv 216 \pmod{359}$ and recovers the original message m by computing

$$\gamma^{-a}.\delta\equiv (216).(288)\equiv 101 \pmod{359}.$$

Example 2: To generate the public-key, entity A generates an odd prime p=359 and computes the composite integer $n=2p^3=92536558$. Then, A chooses the generator α=7395 of the multiplicative cyclic group $Z_{92536558}^{*}$ and a=42514236. Now, computing
$\alpha^{a} \pmod n \equiv 7395^{42514236}\equiv 85784899 \pmod{92536558}$, we have A′s public-key is

$$(n=92536558, \alpha=7395, \alpha^{a}=85784899).$$

and A′s private key is a=42514236.
To encrypt the message m=1100110, where $m \in Z_{92536558}$, entity B selects a random integer k=35923064 and computes
$\gamma\equiv 7395^{35923064}\equiv 66976409 \pmod{92536558}$,
and
$\delta\equiv (1100110).(85784899)^{35923064}\equiv 63539874 \pmod{92536558}$. Then B sends (γ=66976409 and δ=63539874) to A.
To decrypt the sent message, A computes
$\gamma^{\varphi(n)-a}\equiv 66976409^{3625162}\equiv 25198413 \pmod{92536558}$,
and hence recovers
$m\equiv(25198413)(63539874)\equiv 1100110 \pmod{92536558}$
Since $m\equiv 1100110 \pmod{92536558}$ and $m \in Z_{92536558}$, then m=1100110.

Note that there are 92536559 values for m you can choose from the complete residue system modulo 92536558, $Z_{92536558}$.

Next, we describe the modifications of ElGamal public-key encryption to some of the cases described in the list of cyclic groups stated above, and to other new cases.

2.1 ElGamal Cryptosystem in the Domain of Gaussian Integers, Z[i]

El-Kassar (El-Kassar, 2001) considered the arithmetics in the domain of Gaussian integers to extend ElGamal cryptosystem from the integer arithmetics as follows: First, a Gaussian prime β is chosen. If $\beta=\pi$, where $q=\pi\,\pi'$ is prime integer of the form 4k+1, then $G\pi=\{a: 0\le a\le q-1\}=Zq$. This choice will be excluded since the calculations will be identical to those of the classical case. Hence, β is chosen to be a large prime integer p of the form 4k+3 so that $G_\beta=\{a+bi: 0\le a\le p-1,\ 0\le b\le p-1\}$, where the number of elements in G_β is $q(\beta)=p^2$ and in Gβ is $\varphi(\beta)=p^2-1$. Hence, the cyclic group used in the extend ElGamal cryptosystem has an order larger than the square of that used in the classical ElGamal cryptosystem with no additional efforts required for finding the prime p. Now, a generator of θ of G_β^{*} is chosen. Note that there are $\varphi(p^2-1)$ generators in G_β^{*}. A random positive integer *a* is then chosen so that the public key is (p, θ, θ^{a}). Since *a* is a power of θ, then a must be less than the order of the group power G_β^{*} which is p^2-1. This power, a, is the private key. To encrypt a message, we first represent it as an element m in G_β^{*}. Then, a random positive integer k is selected to be used as a power so that k is less than p^2-1. The encrypted message is $c=(\gamma, \delta)$ where $\gamma=\theta^{k}$ and $\delta=m.(\theta^{a})^{k}$. Note that the values of γ and δ must be elements of G_β and hence must be reduced modulo β. The message c is decrypted using the private-key a to compute $\gamma^{-a}.\delta$.

We note that the reduction modulo a Gaussian integer requires computational procedures that are more involved than those used in the reduction modulo an integer. However, since β was chosen to be a prime integer p=4k+3, then the reduction modulo β do not require computational procedures that are different from those used for the integers. In fact, to reduce a+bi modulo β, we find c, d with $0\le c, d\le p-1$ such that $c\equiv a \pmod p$ and $d\equiv b \pmod p$. Then $c+di \in G_\beta$ and $c+di \equiv a+bi \pmod\beta$. Hence, the reduction modulo β in Z [i] is done using integer reductions.

2.2 ElGamal Cryptosystem over Finite Fields

The generalized ElGamal public-key Crypto-system in the setting of a finite field F_q, Where $q=p^n$ for an odd prime integer p and a positive integer n, is based on working with the quotient ring $Z_p[x]/<h[x]>$ where h(x) is an irreducible polynomial over $Z_p[x]$ .We extend the ElGamal public-key cryptosystem to the setting of a finite field. It is well known that $Z_p[x]/<h[x]>$ is a field whose elements are the congruence classes modulo h(x) of polynomials in $Z_p[x]$ with degree this field by $\{a_0 + a_1 x +\ldots+a_{n-1} x^{n-1}:a_0,a_1,\ldots,a_{n-1} \in Z_p[x]\}$to be the complete residue system by A(h(x)) .Note that $Z_p[x]/<h[x]>$ is of order p^n and its nonzero elements from a cyclic group denoted by $U(Z_p[x]/<h[x]>)$. The order of U $(Z_p[x]/<h[x]>)$ is $\varphi(h(x)) = p^n-1$. Let α(x) be a generator of the cyclic group U $(Z_p[x]/<h[x]>)$. The elements in U $(Z_p[x]/<h[x]>)$ can be written as a power of the generator α(x). Hence, U $(Z_p[x]/ <h[x]>) = \{e, \alpha(x), \alpha(x)^2\ldots \alpha(x)^{p^n-1}\}$.

2.3 ElGamal Cryptosystem over Quotient Rings of Polynomials over Finite Fields

The ElGamal public-key cryptosystem is also extended in the setting of the cyclic group of the finite quotient ring $Z_p[x]/<f(x)>$, where p is an odd prime, and f(x) is a reducible polynomial of degree n over $Z_p[x]$ (Smith, 1985). In this case the ring $Z_p[x]/<f(x)>$ is not a field. But according to ElGamal public-key cryptosystem scheme we are only interested in the cyclic groups of units of such rings. Hence, throughout this section we are dealing with any finite fields of order p^n, where p is an odd prime and n is the degree of the reducible polynomial f(x). From a recent study on the structure of cyclic finite fields in (El-Kassar, 2002) by El-Kassar, Chihadi, and Zentout, we can deduce for any finite field F of order $q=p^n$, where p is a prime integer, the group of units U $(F[x]/<f(x)>)$ is cyclic and isomorphic to Z_{q-1} if and only if f(x) is linear. Also, U $(F[x]/<f(x)>)$ is cyclic and isomorphic to $Z_{p-1}\times Z_p$ if and only if $f(x) = h(x)^2$, where h(x) is linear. Hence, we conclude that in order that the group of units U $(Z_p[x]/<h(x)>)$ to be cyclic, h(x) must be irreducible or a square power of only one linear irreducible polynomial. That is, $h(x) = h_1(x)^2$, where $h_1(x) = ax+b$. This means that U $(Z_p[x]/<(ax+b)^2>)$ is cyclic. Moreover, we have that $Z_p[x]/<(ax+b)^2> \cong Z_p[x]/<x^2>$. Hence, we can say that the extension of the ElGamal scheme in this case turns to apply on the group of units of the ring $Z_p[x]/<x^2>$, of order $\varphi(x^2)=p^2-1$. We note that a polynomial f(x) in $Z_p[x]$ belongs to the cyclic group U $(Z_p[x]/<x^2>)$ if and only if (f(x), x) = 1. This is equivalent to saying that x does not divide f(x), where f(x) is a linear polynomial. Hence,

$$U(Z_p[x]/<x^2>) = \{c+dx \mid 1\le c\le p-1,\ 0\le d\le p-1\}.$$

For a detailed look at the algorithms of the extended ElGamal encryption scheme in the domain of Gaussian integers, finite fields and over quotient rings of polynomials over finite fields see (Otrok, 2003).

3 ELGAMAL PUBLIC-KEY SCHEME ATTACK

In order to attack any protocol that uses ElGamal public-key encryption scheme we have to solve the discrete logarithm problem. There are many algorithms for solving the discrete logarithm problem. The most popular algorithm is the Exhaustive Search with its baby-step giant-step algorithm.

3.1 Exhaustive Search

The most obvious algorithm for the discrete logarithm problem (Menezes, 1997) is to successively compute $\alpha^0, \alpha^1, \alpha^2\ldots$ until β is obtained. This method takes O (n) multiplications, where n is the order of α, and is therefore inefficient if n is large (i.e., in cases of cryptographic interest). The algorithm is as follows:

Algorithm 4: Exhaustive Search

INPUT: a generator α of a cyclic group G of prime order n, and an element $\beta \in G$.

OUTPUT: the discrete logarithm $x=\log_\alpha\beta$.

1. Set k=0.
2. Set $\beta=\alpha^k$. If $\beta=x^a$ then return k.
3. Set k=k+1, then return with new k; $0\le k\le n-1$, until $\beta=x^a$ is reached.

3.1.1 Baby-step Giant-step Algorithm

Let $m= [\sqrt{n}]$, where n is the order of α. The baby-step giant-step algorithm is a time-memory trade-off of exhaustive search and is based on the following observation. If $\beta=\alpha^x$, then one can write x=im+j, where $0\le i,j\le m$. Hence, $\alpha^x=\alpha^{im}\alpha^j$, which implies

$\beta(\alpha^{-m})^i=\alpha^j$. This suggests the following algorithm for computing the discrete logarithm $x=\log_\alpha\beta$.

Algorithm 5: The Baby-step algorithm for computing discrete logarithms

INPUT: a generator α of a cyclic group G of order n, and an element $\beta \in G$.

OUTPUT: the discrete logarithm $x=\log_\alpha\beta$.

1. Set $m=[\sqrt{n}]$.
2. Construct a table with entries (j, α^j) for $0\leq j\leq m$. Sort this table by second component. (Alternatively, use conventional hashing on the second component to store the entries in a hash table; placing an entry, and searching for an entry in the table takes constant time.)
3. Compute α^{-m} and set $\gamma=\beta$.
4. For i from 0 to m-1 do the following:
   4.1 Check if γ is the second component of some entry in the table.
   4.2 If $\gamma=\alpha^j$ then return $(x=im+j)$.
   4.3 Set $\gamma=\gamma.\alpha^{-m}$.

The (Baby-step giant-step algorithm) requires storage for $O(\sqrt{n})$ group elements. The table takes $O(\sqrt{n})$ multiplications to construct, and $O(\sqrt{n}\lg n)$ comparisons to sort. Having constructed this table, step 4 takes $O(\sqrt{n})$ multiplications and $O(\sqrt{n})$ table look-ups. Under the assumption that a group multiplication takes more time than log n comparisons, the running time of Baby-step giant-step algorithm is $O(\sqrt{n})$ group multiplications.

4 TESTING AND EVALUATION

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

4.1 ElGamal based Algorithms

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

1. Classical ElGamal.
2. Classical ElGamal with n of the form $2p^t$.
3. ElGamal with Gaussian numbers.
4. ElGamal with irreducible polynomials.
5. ElGamal with reducible polynomials.

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

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

1. All programs are reliable; they can encrypt and decrypt any message.
2. The complexity for each of the algorithms is $O(n^2)$.
3. The reducible polynomial cryptosystem is reliable but it took considerable time to generate a key and to encrypt a message. This does not mean that it is inefficient because it is more secure than the other algorithms. This will be shown later in the attack section.
4. The irreducible polynomial program worked well but only on specific examples. This is due to the fact that it is difficult to generate a random irreducible polynomial according to a prime number *p*.

After 25 runs, we can conclude that:

a- The time needed to find the key, to encrypt and to decrypt for the classical, modified $2p^t$ and Gaussian is approximately negligible compared to the time needed for the polynomials.

b- For the reducible and irreducible polynomials the time needed to encrypt a message is greater than the time needed to find the key or to decrypt a message.

4.2 Attack Algorithm

In order to attack any protocol that uses ElGamal public-key encryption scheme we have to solve the discrete logarithm problem. We enhanced the Baby-step giant-step algorithm to work with the modified algorithms.

To test the security of the algorithms, we implemented attack schemes and applied them on the classical and modified cryptosystem algorithms. After running these attack algorithms, we observed the following:

1. All the attack programs are reliable so that they can hack an encrypted message by finding the private key.
2. The $2p^t$ algorithm is probably stronger than the classical algorithm because we have an unknown

power t. Moreover, it needs t times to attack this algorithm compared to the classical one.

3. The Gaussian algorithm is probably stronger than the classical algorithm since its attack algorithm needs double the time needed to attack the classical one.

4. Perhaps the most difficult one to attack is in the polynomial domain. This is due to the fact that mathematically it is complex and needs considerable computing time to find the modulus of a given polynomial with respect to a certain irreducible or reducible polynomial and with respect to a given prime number.

5 CONCLUSION

In this work, we presented the classic ElGamal cryptosystem and four modifications to it, namely, the ElGamal cryptosystem in Zn, in the domain of Gaussian integers, Z[i], over finite fields, 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 ElGamal 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 discrete logarithm problem that these algorithms utilize. We modified the Baby-step Giant-step algorithm to handle the modified algorithms. We observed that the polynomial domain algorithm was the most challenging to attack due to mathematical complexity.

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

REFERENCES

Cross, J. T. 1983. The Euler′s φ -function in the Gaussian integers, *American Mathematics Monthly* 90, pp. 518-528.

Diffie, W. and Hellman, M. E. 1978. New directions in cryptography, *IEEE Transaction on Information Theory*, IT-22, pp. 472-492, 1978.

ElGamal, T. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Transactions on Information Theory* IT-31, pp. 469-472.

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

El-Kassar, A. N., Rizk M., Mirza N., and Awad, Y. 2001. ElGamal public key cryptosystem in the domain of Gaussian integers, *International Journal of Cryptography*, Volume 7, No. 4.

Haraty, R. and Al-Kassar, A. N. 2003. ElGamal public key cryptosystem using reducible polynomials over a finite field, to appear.

Kenneth, A. R. 1988. Elementary number theory and its *applications, Technical Report, AT&T Bell* Laboratories in Murray Hill, New Jersey.

Menezes, A. J., Van Oorshot, and Vanstone, P. 1997. CSA *Handbook of applied cryptography*, CRC Press.

Otrok, H. 2003. Security testing and evaluation of cryptographic algorithms, M.S. Thesis, Lebanese American University.

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