A COMPARATIVE STUDY OF ELGAMAL BASED CRYPTOGRAPHIC ALGORITHMS

RAMZI A. HARATY*, A. N. EL-KASSAR**, AND HADI OTROK*

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 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* [2]. The concept of public-key cryptography was introduced in their work. This provided a new method for key exchange based on the intractability of discrete logarithm problems. At that time, the authors had no practical realization of a public-key encryption scheme; however, 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 [3] in 1985. El-Kassar et al. [5] and El-Kassar and Haraty [6] modified the ElGamal public-key encryption schemes from the domain of natural integers, Z, to two principal ideal domains, namely the

Date: September 1, 2003.

Key words and phrases. ElGamal cryptosystem, testing and evaluation.

This paper is in final form and no version of it will be submitted for publication elsewhere.

domain of Gaussian integers, $Z[i] = \{a + bi \mid a, b \in Z, i = \sqrt{-1}\}$, and the domain of polynomials over finite fields, $F[x]$, by extending the arithmetic needed for the modifications to these domains. In both cases, it was shown that the same prime modulus used in the classical ElGamal scheme can be used in the new settings to produce larger cyclic groups; hence, both the message space and the key space are enlarged without any additional effort. The larger key space makes the new schemes more secure and harder to break. Moreover, it was shown in both cases that the arithmetic are easy and efficient to apply.

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 among 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 is 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

2.1. Classical ElGamal Encryption Scheme.

The classical ElGamal encryption-decryption scheme is one of the most popular and widely used public-key cryptosystems. It is described in the setting of the multiplicative group Z_p^* of the field Z_p, the field of integers modulo a prime integer p. Let p be a large odd prime integer and let $Z_p = \{0, 1, 2, 3, ..., p-1\}$. Then Z_p is a ring under addition and multiplication modulo p. Since p is prime, Z_p is actually a field under these operations. Moreover, the multiplicative group of the ring of integers modulo p, $Z_p^* = \{1, 2, 3, ..., p-1\}$, is a cyclic group generated by some generator $\theta \neq 1$ whose order is equal to $p-1$. That is, every element of Z_p^* is a power of θ. Note that Z_p is a complete residue system modulo p and Z_p^* is a reduced residue system modulo p.

The ElGamal public-key cryptosystem scheme works as follows: Entity A generates the public-key by first generating a large random prime p, and then it finds a generator θ of Z_p^*. Then, A chooses randomly an integer a, $1 \leq a \leq p-2$, and computes $\theta^a (\mathrm{mod}\, p)$. The public-key is (p, θ, θ^a) and A's private-key is a. To encrypt the message m chosen from Z_p, entity B first obtains A's public-key (p, θ, θ^a). Then, B chooses a random integer k, where $2 \leq k \leq p-2$, and computes $\gamma = \theta^k$ $(\mathrm{mod}\, p)$ and $\delta = m.(\theta^a)^k$ $(\mathrm{mod}\, p)$. The ciphertext is $c = (\gamma, \delta)$. To decrypt the message c sent by B, A uses its own private-key a to recover the message m by computing $c = \gamma^{-a}.\delta$ $(\mathrm{mod}\, p)$.

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, $1 \leq a \leq p-2$, and compute $\theta^a (\bmod p)$.
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, $2 \leq k \leq p-2$.
4- Compute $\gamma = \theta^k \ (\bmod p)$ and $\delta = m \cdot (\theta^a)^k \ (\bmod p)$.
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 $\gamma^{p-1-a} \ (\bmod p)$
(note: $\gamma^{p-1-a} = \gamma^{-a} = \theta^{-ak} (\bmod p)$).
2- Recover the message m by computing $\gamma^{-a}.\delta \ (\bmod p)$.

Example 1. *In order to generate the public-key, entity A selects an odd prime $p = 359$ and finds a generator $\theta = 124$ of Z_{359}^*. Then, A chooses the private-key $a = 292$ and computes $124^{292} \equiv 205 = \theta^a (\bmod 359)$. Therefore, A's public-key is $(p = 359, \theta = 124, \theta^a = 205)$ and A's private-key is $a = 292$.*

To encrypt the message $m = 101$ chosen from Z_{359}, B selects a random integer $k = 247$ and computes

$$124^{247} \equiv 291 = \gamma (\bmod 359)$$

and

$$101.205^{247} \equiv 288 = \delta (\bmod 359).$$

Then, B sends $(\gamma = 291,\ \delta = 288)$ to A. Note that B has 359 different values for m to choose from Z_{359}. Finally, A computes

$$\gamma^{p-1-a} = 291^{66} \equiv 216 (\bmod 359)$$

and recovers the original message m by computing

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

The classical ElGamal encryption scheme, described in the setting of the multiplicative group Z_p^*, can be easily generalized to work in any finite cyclic group G. 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 [8] mentioned that some of 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 Jacobian of a hyperelliptic curve defined over a finite field, and the class group 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 θ. Assuming that G is written multiplicatively, a random integer a, $1 \leq a \leq n-1$, is selected and the group element θ^a is computed. $A's$ public-key is (θ, θ^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 (θ, θ^a), then select a random integer k, $1 \leq k \leq n-1$, and compute $\gamma = \theta^k$ and $\delta = m.(\theta^a)^k$. 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 $(\gamma^{-a}).\delta$.

Next, we describe the modifications of ElGamal public-key encryption in Z_n, where n is a composite integer, the domain of Gaussian integers $Z[i]$, and the domain of the rings of polynomials over finite fields $F[x]$.

2.2. **ElGamal Cryptosystem in Z_n.** In this scheme, the group G selected is Z_n^*, where n is not necessarily a prime. Since ElGamal public-key encryption scheme depends on the fact that the group selected is cyclic, it is important to know the positive integers n for which the multiplicative group Z_n^* of integer modulo n is cyclic. Although the ring Z_n modulo a composite integer is not a field, it is well known, that the multiplicative group Z_n^* is cyclic if and only if n is either 2, 4, p^t, or $2p^t$, where p is an odd prime and $t \geq 1$, see [7]. The order of Z_n^* is $\phi(n) = (p-1)p^{t-1}$, where $\phi(n)$ is Euler phi function. We note that if θ is a generator of Z_n^*, then $Z_n^* = \{\theta^i \mid 0 \leq i \leq \phi(n)-1\}$ and $b = \theta^k (\bmod n)$ is also a generator of Z_n^* if and only if $(k, \phi(n)) = 1$.

To describe the generalized scheme used in the above operation, three algorithms describing the modified ElGamal method restricted to the multiplicative group Z_n^* are needed.

To generate the public and private-keys, entity A should use the following algorithm:

Algorithm 4. *(Key generation for modified ElGamal public-key encryption in Z_n.)*

(1) *Generate a large random odd prime p and a positive integer t.*
(2) *Compute the composite integer n ($n = p^t$ or $n = 2p^t$) and the integer $\phi(n) = (p-1)p^{t-1}$.*
(3) *Find a generator θ of the cyclic multiplicative group of integers modulo n, $Z_n^* = \{\theta^i \mid 0 \leq i \leq \phi(n)-1\}$.*
(4) *Select a random integer a, $2 \leq a \leq \phi(n)-1$.*
(5) *Compute $\theta^a (\bmod n)$.*
(6) *$A's$ public-key is (n, θ, θ^a).*
(7) *$A's$ private-key is a.*

Note that the integer a in step 4 is chosen to be between 2 and $\phi(n)-1$ since a is a power of the generator θ of Z_n^*.

To encrypt a message m in Z_n, the complete residue system modulo n, Entity B should use the following algorithm:

Algorithm 5. *(Modified ElGamal public-key encryption in Z_n.)*

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

Note that k should be chosen to be an integer from 2 to $\phi(n) - 1$ since it will be used as a power of the generator θ.

To recover the plaintext m from the sent ciphertext c, entity A should use the following algorithm:

Algorithm 6. *(Modified ElGamal public-key decryption in Z_n.)*

(1) *Receive the ciphertext c from entity B.*
(2) *Use the private-key a to compute $\gamma^{\phi(n)-a} \pmod n$.*
(3) *Recover the plaintext m by computing $\gamma^{-a}.\delta \pmod n$.*

Example 2. *To generate the public-key, entity A generates an odd prime $p = 359$ and computes the composite integer $n = 2\ p^3 = 92536558$. Then, A chooses the generator $\theta = 7395$ of the multiplicative cyclic group $Z^*_{92536558}$ and $\theta = 42514236$. Now, computing $\theta^a (\bmod n) \equiv 7395^{42514236} \equiv 85784899 (\bmod 92536558)$, we have A's public-key is*

$$(n = 92536558, \theta = 7395, \theta^a = 85784899).$$

and A's private key is $\theta = 42514236$.

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 $\delta \equiv (1100110).(85784899)^{35923064} \equiv 63539874 (\bmod 92536558)$. Then B sends $(\gamma = 66976409$ and $\delta = 63539874)$ to A.

To decrypt the sent message, A computes $\gamma^{\phi(n)-a} \equiv 66976409^{3625162} \equiv 25198413 \pmod{92536558}$, and hence recovers $m \equiv (25198413) * (63539874) \equiv 1100110$ (mod 92536558). Since $m \equiv 1100110 (\bmod\ 92536558)$ 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, $Z_{92536558}$.

2.3. ElGamal Cryptosystem in the Domain of Gaussian Integers.

El-kassar et al. [5] extended ElGamal cryptosystem from the domain of integers to the domain of Gaussian integers as follows: First, a Gaussian prime β is chosen. Let G_β be a complete residue system modulo β. Then, G_β is a field under addition and multiplication modulo the Gaussian integer β. If $\beta = \pi$, where $q = \pi\overline{\pi}$ is prime integer of the form $4k + 1$, then $G_\beta = \{a : 0 \le a \le q - 1\} = Z_q$, see [1]. 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 $G_\beta = \{a + bi : 0 \le a \le p - 1, 0 \le b \le p - 1\}$, see [1]. The number of elements in G_β is p^2 and in G^*_β, its multiplicative group of units, is $\phi(\beta) = p^2 - 1$. Hence, the cyclic group used in the extended 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 $\phi(p^2 - 1)$ 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 $\gamma = \theta^k (\text{mod}\, \beta)$ and $\delta = m.(\theta^a)^k (\text{mod}\, \beta)$. 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 $c \equiv a (\text{mod}\, p)$ and $d \equiv b (\text{mod}\, p)$. Then $c + di \in G_\beta$ and $c + di \equiv a + bi (\text{mod}\, \beta)$. Hence, the reduction modulo β in $Z[i]$ is done using integer reductions.

2.4. **ElGamal Cryptosystem over Finite Fields.** The generalized ElGamal public-key cryptosystem 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 $Z_p[x]/\langle h(x)\rangle$, 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 $Z_p[x]/\langle h(x)\rangle$ is a field whose elements are the congruence classes modulo $h(x)$ of polynomials in $Z_p[x]$ with degree less than n. We identify this field by the complete residue system $A(h(x)) = \{a_0 + a_1 x + ... + a_{n-1}x^{n-1} \mid a_0, a_1, ..., a_{n-1} \in Z_p[x]\}$. Note that $Z_p[x]/\langle h(x)\rangle$ is of order p^n and its nonzero elements form a cyclic group denoted by $U(Z_p[x]/\langle h(x)\rangle)$. The order of $U(Z_p[x]/\langle h(x)\rangle)$ is $\phi(h(x)) = p^n - 1$. Let $\theta(x)$ be a generator of the cyclic group $U(Z_p[x]/\langle h(x)\rangle)$. The elements in $U(Z_p[x]/\langle h(x)\rangle)$ can be written as a power of the generator $\alpha(x)$. Hence, $U(Z_p[x]/\langle h(x)\rangle) = \{e, \theta(x), \theta(x)^2, ..., \theta(x)^{p^n-1}\}$.

2.5. **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 $Z_p[x]/\langle f(x)\rangle$, where p is an odd prime, and $f(x)$ is a reducible polynomial of degree n over $Z_p[x]$, see [6]. In this case the ring $Z_p[x]/\langle f(x)\rangle$ 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 quotient ring $Z_p[x]/\langle f(x)\rangle$ of order p^n, where p is an odd prime and n is the degree of the reducible polynomial $f(x)$. Using the characterization of the structure of the group unit of $Z_p[x]/\langle f(x)\rangle$ given in [10], El-Kassar et Haraty [4] obtained a characterization of all primes p and polynomials $f(x)$ for which $Z_p[x]/\langle f(x)\rangle$ is cyclic. Two particular cases of interest are as follows. For any finite field F of order $q = p^n$, where p is an odd 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 for the group of units $U(Z_p[x]/\langle h(x)\rangle)$ 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 $U(Z_p[x]/\left\langle (ax + b)^2\right\rangle)$ is cyclic. Moreover, we have that $Z_p[x]/\left\langle(ax+b)^2\right\rangle \simeq Z_p[x]/\left\langle x^2\right\rangle$. Hence, we can say that the extension of the ElGamal scheme in this case applies to the group of units of the ring $Z_p[x]/\left\langle x^2\right\rangle$, of order $\phi(x^2) = p^2 - 1$. We note that a polynomial $f(x)$ in $Z_p[x]$ belongs to the cyclic group $U(Z_p[x]/\left\langle x^2\right\rangle)$ 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, $U(Z_p[x]/\left\langle x^2\right\rangle) = \{c + dx \mid 1 \leq c \leq p-1, 0 \leq d \leq p-1\}$.

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 [9].

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, see [8]. The security of ElGamal encryption scheme depends on the intractability of the discrete logarithm problem. In the following, we describe some of the algorithms for solving the discrete logarithm problem in the general setting of a finite cyclic group G of order n generated by α. The discrete logarithm of an element β in G to the base α, denoted $log_\alpha\beta$, is the unique integer x, such that $\beta = \alpha^x$, where $0 \leq x \leq n-1$.

The groups considered in this paper are:

(1) The multiplicative group Z_p^*;of the integers modulo a prime p.
(2) The multiplicative group G_β^*; of the Gaussian integers modulo a prime β.
(3) The group of units Z_n, where n is a composite integer of the form $2p^t$.
(4) The multiplicative group $F^*$of a finite field F.
(5) The group of units of the quotient ring $Z_p[x]/ < x^2 >$.

Given a generator α of a finite cyclic group G of order n and an element β in G, the discrete logarithm problem is the problem finding the unique integer $x, 0 \leq x \leq n-1$, such that given $\beta = \alpha^x$. The known algorithms for solving the discrete logarithm problem are: exhaustive search, the baby-step giant-step algorithm, Pollard's rho algorithm, Pohlig-Hellman algorithm, and the index-calculus algorithms. The first three algorithms work in arbitrary groups. The baby-step giant-step algorithm is a time-memory trade-off of exhaustive search. Pollard's rho algorithm is a randomized algorithm. It requires a negligible amount of storage and has the same expected running time as the baby-step giant-step algorithm. The Pohlig-Hellman algorithm works in arbitrary groups but is especially efficient if the order of the group has only small prime factors. The index-calculus algorithms are efficient only in certain groups. Detailed descriptions and references of these algorithms can be found in [8].

In the following we describe the exhaustive search and the baby-step giant-step algorithms. These algorithms will be used in section 4 to evaluate the various versions of ElGamal cryptosystems.

3.1. Exhaustive Search.

An obvious algorithm for the discrete logarithm problem is to successively compute α^0,α^1, α^2, ... until β is obtained. If n is the order of α, the exhaustive search takes $O(n)$ multiplications, and is therefore inefficient if n is large (i.e., in cases of cryptographic interest). However, it can be used to compare the various ElGamal based cryptosystems having small parameters. The algorithm is as follows:

Algorithm 7. Exhaustive Search

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 $k = 0$.
(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 n be the order of a generator α and let $m = \lceil \sqrt{n} \rceil$, the smallest integer greater than or equal to $\sqrt{n}$. The baby-step giant-step algorithm is based on the following observation. Suppose that $\beta = \alpha^x$. Writing $x = km + j$, where $0 \le k,\ j \le m$, we have that $\alpha^x = \alpha^{km}\alpha^j$ and hence $\beta(\alpha^{-m})^k = \alpha^j$. This suggests the following algorithm for determining $x = \log_\alpha \beta$.

Algorithm 8. *Baby−step giant−step algorithm for finding discrete logarithms*

INPUT: α*, a generator of a cyclic group G of order n, and an element $\beta \in G$.*
OUTPUT: *the discrete logarithm $x = \log_\alpha \beta$, where $\beta = \alpha^x$.*

```
(1) Set $m = \lceil \sqrt{n} \rceil$.
(2) For $0 \le j \le m$, find $\alpha^j$ and construct a table with entries $(j,\ \alpha^j)$.
(3) Sort this table by second component.
(4) Find $\alpha^{-m}$ and set $\gamma = \beta$.
(5) For $k$ from $0$ to $m-1$ do the following:
        5.1 If $\gamma = \alpha^j$ then return $(x = km + j)$. Else, set $\gamma = \gamma.\alpha^{-m}$.
```

This algorithm requires storage for $O(\sqrt{n})$ group elements. To construct the table, $O(\sqrt{n})$ multiplications and $O(\sqrt{n}\lg n)$ comparisons to sort are required. Step 5 takes $O(\sqrt{n})$ multiplications and $O(\sqrt{n})$ table look-ups. Assuming that a group multiplication requires more time than $\log n$ comparisons, we have that the running time of Baby-step giant-step algorithm is $O(\sqrt{n})$ group multiplications.

Note that the performance can be improved by restricted exponents a special form. Usually, exponents having small Hamming weight are used. The Hamming weight of an integer is the number of ones in its binary representation. The following is an example of a modified Baby-step giant-step attack for Gaussian integers.

Example 3. $($*Baby − step giant − step algorithm for logarithms in* G^*_{11}

Let $p = 11$. The element $\alpha = 2 + 3i$ is a generator of G^*_{11}, of order $n = p^2 - 1 = 120$. Consider $\beta = 2 + 6i = (2 + 3i)^{67}$. Applying the Baby-step giant-step algorithm, $\log_{2+i} 2 + 6i \pmod{11}$ is computed as follows.

(1) *Set $m = \lceil \sqrt{120} \rceil = 11$.*
(2) *Construct a table whose entries are $(j,\ \alpha^j \pmod p)$ for $0 \le j < 11$:*
(3)

j	0	1	2	3	4	5	6	7	8	9	10
$(2+3i)^j$	1	$2+3i$	$6+i$	$9+9i$	$2+i$	$1+8i$	$8i$	$9+5i$	$3+4i$	$5+6i$	$3+5i$

(4) *This table can be sorted by second component using the linear order $\preceq$ defined by $a + bi \preceq c + di$ iff $(a < c)$ or $(a = c$ and $b < d)$. The table becomes:*

j	6	0	5	4	1	8	10	9	2	7	3
$(2+3i)^j$	$8i$	1	$1+8i$	$2+i$	$2+3i$	$3+4i$	$3+5i$	$5+6i$	$6+i$	$9+5i$	$9+9i$

(5) *Compute* $\alpha^{-1} = (2+3i)^{-1}(\text{mod}\,11) = 1+4i$ *and then compute* $\alpha^{-m} = (1+4i)^{11}(\text{mod}\,11) = 1+7i$.

(6) *Next,* $\gamma = \beta\alpha^{-mk}(\text{mod}\,11)$ *for* $k = 0, 1, 2, \ldots$ *is computed until a value in the second row of the above table is obtained. This yields the following table:*

i	0	1	2	3	4	5	6
γ	$2+6i$	$4+9i$	$7+4i$	$1+9i$	$4+5i$	2	$2+3i$

When $k = 6$, $\gamma = 2+3i$ *is the first value of* γ *appearing in the second row of the table in step 4 corresponding to* $j = 1$. *Finally, since* $\beta\alpha^{-29m} = 11834 = \alpha^{147}$, *then* $x = km + j = 6 \times 11 + 1 = 67$. *Therefore,* $\beta = \alpha^{67}$, *i.e., the discrete logarithm* $\log_{362} 9972 = 5425$.

4. Testing and Evaluation

In this section, we compare and evaluate the different classical and modified cryptosystems by showing the implementation of the cryptosystem algorithms with their running results. Also, we test the security of the algorithms by implementing different attack algorithms to crack the encrypted messages. All this is done using Mathematica 5.0 as a programming language and a centrino acer computer with 1.5 GHZ CPU and 256 MB DDRAM.

4.1. ElGamal based Algorithms.

Using Mathematica 5.0 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.

The various procedures were compared as follows.

a-A total of 25 runs of the various algorithms were conducted. In each run, a 20-digit random prime integer p of the form $4k+3$ was generated.

b-The same prime p was used for all algorithms.

c-For each method a public key was generated by finding a generator θ, a random integer a, and computing θ^a.

d-Using the public key (θ, θ^a), the same message $m = 12345678$ was encrypted by all algorithms to obtain the cypher text $c = (\gamma, \delta)$.

e-The decryption algorithms were then used to recover m.

f-All algorithms used the built-in Mathematica functions for modular arithmatic and for finding powers modulo an integer, Gaussian integer or a polynomial over Z_p.

g-The running times of the algorithm (Key generation, encryption, decryption) for each method were recorded.

Note that the cyclic groups used, and their corresponding orders, are:

(1) Classical ElGamal: Z_p^* of order p.
(2) Classical ElGamal with n of the form $2p^2$: Z_n^* of order $p^2 - p$.
(3) ElGamal with Gaussian integers: G_p^* of order $p^2 - 1$.
(4) ElGamal with reducible polynomials: $U(Z_p[x]/ < x^2 >)$ of order $p^2 - p$.
(5) ElGamal with irreducible polynomials: $U(Z_p[x]/ < x^2 + ax + b >)$ of order $p^2 - 1$.

Except for the classical ElGamal in the setting of the cyclic group Z_p^*, all cyclic groups used have comparable sizes. Hence, we expect the algorithms in the first case to be much faster. A different prime p having 40 digits could have been used for that case; but this would have been equivalent to case 2.

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. Figure 1 shows the average execution time of 25 runs of the various algorithms.

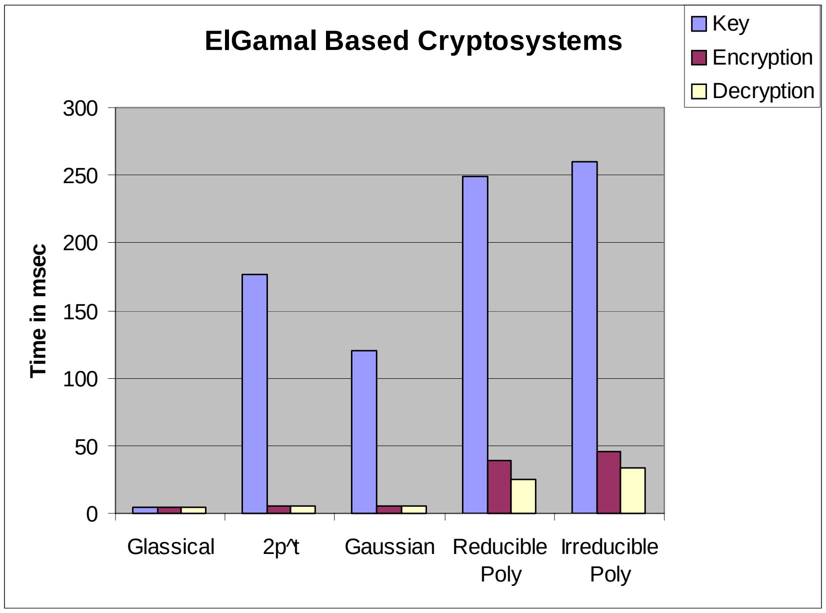

Figure 1: Average execution time for ElGamal Cryptosystems

Comparing these algorithms, we observe the following:

a- All programs are reliable; they can encrypt and decrypt any message.

b- The complexity for each of the algorithms is $O(n^2)$.

c- The reducible polynomial cryptosystem is reliable but took more 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. Also, these algorithms can be made more efficient by modifying the built-in Mathematica functions to take advantage of the arithmetic modulo x^2.

d- The irreducible polynomial program in the setting $Z_p[x]/ < x^2 + ax + b >$ worked well but requires more time. The encryption and decryption execution time for the irreducible polynomial scheme is slightly more than that of the reducible case. This is due to the additional computations needed for the arithmetic modulo the polynomial $x^2 + ax + b$.

e-The key generation time for the irreducible polynomial scheme is considerably more than that of the other methods. This is due to the fact that an irreducible polynomial must be generated before a generator θ is found. On average, it took about 0.4 sec to generate a quadratic irreducible polynomial for a 20-digit prime number.

Note that in Figure 1, the time of generating the irreducible polynomial is not included. Figure 2, shows the average execution time if the time of finding an irreducible polynomial is included in the key generating algorithm.

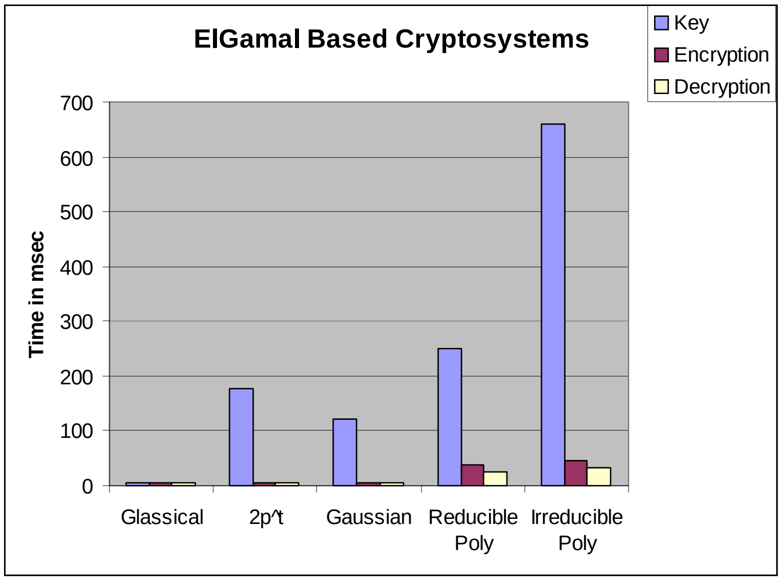

Figure 2: Average execution time for ElGamal Cryptosystems

From these results, we can conclude that:

a- The time for generating the key depends on finding the generator θ and not the prime p. The time for generating a prime number is negligible. The average time need for generating a 100-digit (recommended size) random prime is approximately 0.1 sec.

b- It took more time to find the key in the case of polynomials. This will not be a problem if common system-wide parameters are used. In such a case, all entities may elect to use the same cyclic group G and generator ϕ. Also, once a generator ϕ for a given prime p is found, all other generators can be easily obtained.

c- The time needed to encrypt and to decrypt the message for the classical, modified $2p^t$ and Gaussian is better than the time needed for the polynomials. However, the time is very reasonable even for larger primes.

d- Overall, the Gaussian integers methods performed very well. The algorithms can be made more efficient by modifying Mathematica built-in functions to take advantage of the arithmetic modulo the Gaussian prime p of the form $4k + 3$.

e- The key generation time in the case of Gaussian integers is less than that of the modified $2p^t$ method. This is due to the fact that the number of generators in $Z^*_{2p^2}$, which is $\phi(p(p-1))$, is almost always more than the number of generators in G^*_p, which is $\phi(p^2-1)$. In fact, among the first $200,000$ primes, there are only 7 primes p of the form $4k + 3$ for which $\phi(p^2-1) > \phi(p(p-1))$.

f-The reducible polynomial method is little slower but provide more security. The irreducible polynomial method is not recommended since it is as secure as the reducible case but requires more time especially in finding the key.

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 *Exhaustive search* and $Baby - step\ giant - step$ algorithms to work with the modified algorithms.

To test the security of the algorithms, we implemented and applied the attack schemes to the classical and modified cryptosystem algorithms. The ElGamal algorithm using irreducible polynomials was not tested since the attack time would be equivalent to that of the reducible polynomial case. For the exhaustive search algorithm, a random 3-digit prime p of the form $4k + 3$ was generated and a public key was obtained for each the four methods using the same prime. The average time it took to break the key for a total of 25 runs are shown in figure 3.

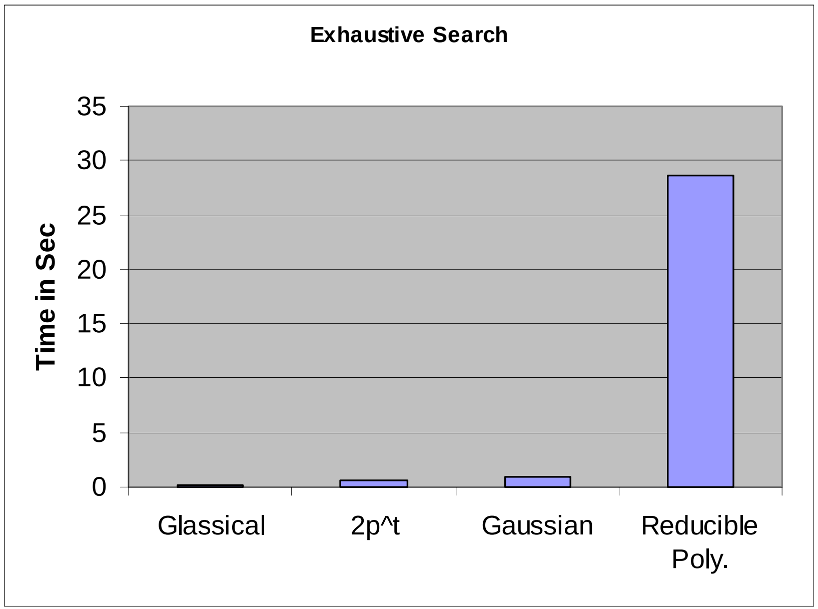

Figure 3. Attack Time: Exhaustive Search Algorithm

Attacking the ElGamal schemes using Baby-step giant-step algorithm, a random 4-digit prime p of the form $4k + 3$ was generated and a public key was obtained for each the four methods using the same prime. The average time it took to break the key for a total of 25 runs are shown in figure 4.

After running these attack algorithms, we observed the following:

a- All the attack programs are reliable so that they can hack an encrypted message by finding the private key.

b- The $2p^t$ algorithm is stronger than the classical algorithm because we have an unknown power t.

c- The Gaussian algorithm is stronger than the classical algorithm. The attack algorithm for Gaussian integers required more time than that of the $2p^2$ algorithm.

d- The most difficult algorithm to attack is in the polynomial domain. This is due to the fact that mathematically it is complex and needs considerable computing time to perform arithmetic modulo a given polynomial.

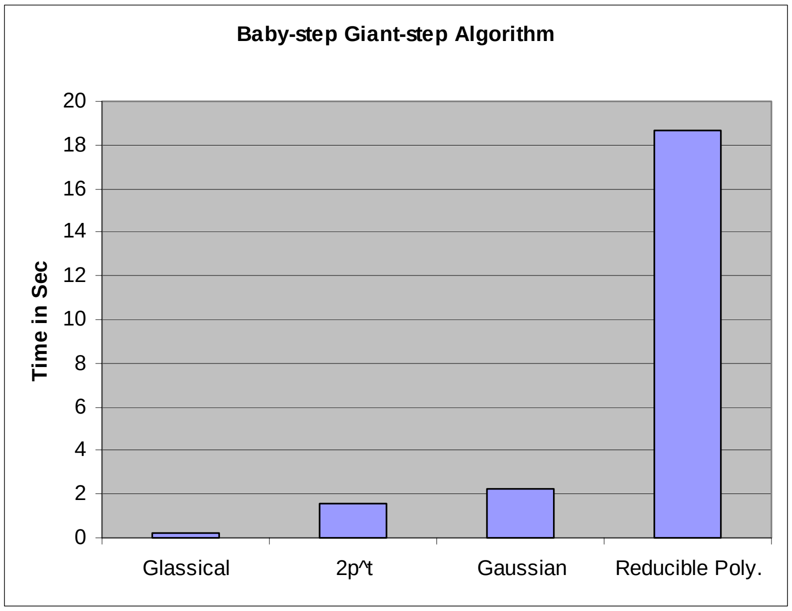

Figure 4. Attack Time: Baby-Step Giant-Step Algorithm

5. Conclusion

In this work, we presented the classic ElGamal cryptosystem and four modifications to it, namely, the ElGamal cryptosystem in Z_n, 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 classical ElGamal scheme is the weakest to attack and one of the modified methods should be used. The ElGamal scheme in the multiplicative group Z_n^*, where $n = 2p^2$, is the easiest to apply and the weakest among the modified method. The ElGamal scheme in the domain of Gaussian is superior to that of Z_n^* since it requires less time to generate a key, about the same time to encrypt and decrypt a message, and is more secure. The ElGamal scheme in the setting of a finite field has a disadvantage in finding an irreducible polynomial. The reducible polynomial scheme was the most challenging to attack due to the mathematical complexity of arithmetic modulo a polynomial.

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.

6. REFERENCES

[1] Cross, J. T. *The Euler's φ-function in the Gaussian Integers*, American Mathematics Monthly 90, pp. 518-528, 1983.

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

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

[4] El-Kassar, A. N., Chihadi H., and Zentout D. *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, 2002.

[5] El-Kassar, A. N., Rizk M., Mirza N., and Awad,Y. *ElGamal Public key cryptosystem in the domain of Gaussian integers*, International Journal of Applied Mathematics, Volume 7, No. 4, 2001.

[6] El-Kassar, A. N., and Haraty, Ramzi. ElGamal publickey cryptosystem using reducible polynomials over a finite field, Proceedings of the ISCA 13th International Conference on Intelligent and Adaptive Systems and Software Engineering, ISCA 2004, Nice, France, 189-194, 2004.

[7] Kenneth, A. R. *Elementary Number Theory and its Applications*, AT&T Bell Laboratories in Murray Hill, New Jersey, 1988.

[8] Menezes, A. J., Van Oorshot, and Vanstone, P. C. S. A. *Handbook of Applied Cryptography*, CRC press, 1997.

[9] Otrok, H. *Security Testing and Evaluation of Cryptographic Algorithms*, M.S. Thesis, Lebanese American University, June 2003.

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

*Lebanese American University P.O. Box 13-5053 Chouran, Beirut, Lebanon 1102 2801
E-mail address: rharaty@lau.edu.lb

**Beirut Arab University, Mathematics Department, Beirut, Lebanon
E-mail address: Ak1@bau.edu.lb