Public key cryptosystem based on the discrete logarithm problem

Adullah. A. Shihadeh

Abstract

In mathematics, specifically in abstract algebra and its applications, discrete logarithms are group-theoretic analogues of ordinary logarithms. The problem of computing both problems are difficult, algorithms from one problem are often adapted to the other, and the difficulty of both problems has been exploited to construct various cryptgraphic systems. This work describes an application of the discrete logarithm that is cryptosystem based on the discrete logarithm problem. The paper defined at first the classical diffie-hellman key exchange, then extend this cryptosystem to the domain of gaussian integer that make it more difficult to break. Then we mention the elgamal public key cryptosystem at the domain of integer number, to extend it later to the domain of gaussian interger also to be more difficult to break that is the main objective in this work.

Key words: diffie-hellman key exchange, elgamal cryptosystem, gaussian integer

Introduction

Cryptography is the study of techniques that can be used to disguise a message so that only the intended recipient can remove the disguise and read it. It describe the elgamal public-key cryptosystem and the diffie-hellman key exchange and the then extends these cryptosystem over the domain of gaussian integers. The main purpose of this paper is to use the gaussian integers; the set of all complex numbers $a+bi$ with $a, b \in \mathbb{Z}$ in the Gaussian integers, that can help the cryptosystem to be more secure. As a part of finding public-key cryptography, diffie-hellman proposed a concrete scheme for obtaining a common private key using an authentic but not secret channel. This private key can then for instance be used for encryption with a symmetric encryption algorithm. In our work we begining with a study of the classical diffie-hellman key exchange that use the integer number, then we extend it using the domain of gaussian integer.

In 1985 a powerful and practical public-key scheme was produced by taher elgamal. The classical ELGamal encryption scheme is one of the most widely used public-key cryptosystems. It is described in the setting of the multiplicative group $\mathbb{F}_p^*$ of the finite field $\mathbb{F}_p$ the field integers modulo a prime integer $p$. Many aspects of the arithmetics over the domain of integers $\mathbb{Z}$ can be carried out to the domain of gaussian integers $\mathbb{Z}[i]$, however the computational details of the arithmetics in $\mathbb{Z}[i]$ are different from those of $\mathbb{Z}$. In this paper, we extended the computational procedures behind the elgamal algorithm using arithmetics.
module gaussian integers. First we review the classical elgamal algorithm. Then we modify the elgamal algorithm to the domain of gaussian integers. Dr. Taher elgamal (born 18 August 1995) is an egyptian american cryptographer. Elgamal is sometimes written as el gamal or elgamal. In 1985, elgamal published a paper titled: AA public key cryptosystem and a signature scheme based on discrete logarithms@, in which he proposed the design of the elgamal discrete logarithm cryptosystem and of the elgamal signature scheme. The latter scheme became the basis for digital signature algorithm (DSA) adopted by national Institute fo Standards and Technology (NIST) as the digital signature standard (DSS). He also participated in the >SET= credit card payment protocol, plus a number of Internet payment scheme.

1 Discrete Logarithm Problem

Discrete logarithms are perhaps simplest to understand in the group \( \mathbb{Z}_p^* \), the set of integers \{1,...,p-1\} under multiplication module the prime \( p \).

If we want to find the \( k^{th} \) power of one of the numbers in this group, we can do so by finding its \( k^{th} \) power as an integer and then finding the remainder after division by \( p \).

This process is called discrete exponentiation. For example, consider \( \mathbb{Z}_{17}^* \). To compute \( 3^4 \) in this group, we first compute \( 3^4 = 81 \), and then we divide 81 by 17, obtaining a reminder of 13. Thus \( 3^4 = 13 \) in the group \( \mathbb{Z}_{17}^* \).

Discrete logarithm is just the inverse operation: Given that \( 3^k \equiv 13 \pmod{17} \), what is the value of \( k \) that makes this true? Actually, there are infinitely many answers, due to the modular nature of the problem; we typically seek the least nonnegative answer, which is \( k=4 \).

Definition 1:

Let \( G \) be a finite cyclic group, and \( \theta \in G \) be a generator of \( G \). The discrete logarithm of some element \( g \in G \), denoted \( \log_{\theta} g \), is the unique integer \( a;0\leq a<|G| \), such that \( g = \theta^a \).

If \( \theta \) is not a generator, the notion of the discrete logarithm of \( g \) to the base \( \theta \) is extended to be the smallest positive integer \( x \), such that \( g = \theta^x \), if it exists. The following facts, known from ordinary logarithms, also hold for discrete logarithms.

Theorem 1:

Let \( G = \langle \theta \rangle \) be a cyclic group of order \( n \), and let \( a \) and \( b \) be elements of \( G \). Then

1. \( \log_{\theta} ab \equiv \log_{\theta} a + \log_{\theta} b \pmod{n} \)
2. \( \log_{\theta} a^x \equiv x \log_{\theta} a \pmod{n} \)

Proof:

(1) Now \( \theta^{\log_{\theta} ab} \equiv ab \), and \( \theta^{\log_{\theta} a + \log_{\theta} b} \equiv \theta^{\log_{\theta} a} \cdot \theta^{\log_{\theta} b} \equiv ab \). Hence \( \theta^{\log_{\theta} ab} \equiv \theta^{\log_{\theta} a + \log_{\theta} b} \), and since \( \theta \) is a generator, then \( \log_{\theta} ab \equiv \log_{\theta} a + \log_{\theta} b \pmod{n} \).

(2) \( \theta^{\log_{\theta} a^x} \equiv a^x \), and \( \theta^{x \log_{\theta} a} \equiv (\theta^{\log_{\theta} a})^x \equiv a^x \).

Then \( \theta^{\log_{\theta} a^x} \equiv \theta^{x \log_{\theta} a} \), and so \( \log_{\theta} a^x \equiv x \log_{\theta} a \pmod{n} \). ■

Definition 2:

The discrete logarithm problem (DPL) in a finite cyclic group \( G \), with a generator \( \theta \) and an element \( g \),
is to find the integer \( a; 0 \leq a \leq |G| - 1 \), such that \( g = \theta^a \) holds.

Computing discrete logarithms is apparently difficult (no efficient algorithm is known), while the inverse problem of discrete exponentiation is not (it can be computed efficiently using exponentiation by squaring, for example). This asymmetry is analogous to the one between integer factorization and integer multiplication. Both asymmetries have been exploited in the construction of cryptographic systems.

Popular choices for the group \( G \) in discrete logarithm cryptography are the cyclic groups \( \mathbb{Z}_p^* \) where \( p \) is an odd prime. Newer cryptography applications use discrete logarithms in multiplicative group \( G_\beta^* = \mathbb{Z}_\beta^* \) the group of units gaussian integer modula a gaussian prime \( \beta \), with the multiplication binary operation.

We will mention now some of cryptosystem which are based on the discrete logarithm problem as elgamal cryptosystem, and diffie-hellman key exchange, and we will use the multiplication \( \mathbb{Z}_p^* \) and then we will modify these cryptosystems to the domain of gaussian integers.

### 2 Diffie-Hellman Key Exchange:

Let \( G \) be a cyclic group of order \( q \) and let \( \theta \in G \) be a generator of \( G \) such that computing discrete logarithm in \( G \) is infeasible. Furthermore, let \( Y^a = \theta^a \) and \( Y^b = \theta^b \) be the public keys. To derive a common secret key \( k \), Ali and Basem exchange their public keys over the authentic channel and raise the partners public key to the power of their own secret key and thereby get \( k = (Y^a)^{x_\theta} = (Y^b)^{x_\theta} \) the common private key. Any adversary now have the public keys \( Y^a \) and \( Y^b \) and his problem is to compute \( k \), this problem is called diffie-hellman problem (DHP).

#### 1 Diffie-Hellman key exchange over \( \mathbb{Z}_p^* \) (Classical Diffie-Hellman key exchange):

The classical Diffie-Hellman key exchange can be described as follow: Let \( p \) be a large odd prime, then \( \mathbb{Z}_p \) is field under addition and multiplication module \( p \), and so \( \mathbb{Z}_p^* \) is a cyclic group generated by some generator \( \theta \).

Suppose that Ali and Basen want to generate a common private key using an authentic but not secret channel.

1. Ali and Basen first choose a generator \( \theta \in \mathbb{Z}_p^* \).
2. Every one of them now choose his own secret key, Ali choose \( x_\theta \) and Basen chooses \( x_\theta \), where \( 1 < x_\theta, x_\theta < p-2 \).
3. In this step Ali computes his public key \( Y^a = \theta^{x_\theta} \pmod{p} \), and sends it to Basem.
4. Basem also makes the same thing, he computes his public key \( Y^b = \theta^{x_\theta} \pmod{p} \), and sends it to Ali.

5. To compute their private key \( k \), Ali raises \( Y^b \) to the power \( x_\theta \) to find \( k \equiv \theta^{x_\theta x_\theta} \pmod{p} \). Similarly Basem computes \( k \) by raising \( Y^a \) to the power \( x_\theta \) and finds the same value of \( k \), the common private key.

**Example 1:**

Ali and Basem decided to work over \( \mathbb{Z}_{53}^* \) and they chose the generator \( \theta = 20 \). Now Ali chooses his secret key \( x_\theta = 14 \) and computes his public key \( Y^a = 20^{14} = 17 \pmod{53} \), then he sends it to Basem. At the same time Basem does a similar work, he choose his secret key \( x_\theta = 30 \), and after he computes his public key \( Y^b = 20^{30} = 7 \pmod{53} \), then he sends it to Ali. Now each of them wants to compute their common public key \( k \), Ali computes it by \( k = 7^{14} = 46 \pmod{53} \), and Basem computes it by \( k = 17^{30} = 46 \pmod{53} \).
2 Diffie-Hellman key exchange over the domain of gaussian integers:

The algorithm here is similar to the algorithm in classical case but we choose the prime $\beta$ in this case from the domain of gaussian integers. First case; if $\beta=1+i$ or $\beta=1-i$ and this case is clearly excluded since the order of the group $G_\beta^*$ is very small. Second case: if $\beta=\pi$ where $q=\pi \equiv 1 \pmod{4}$, and so $|G_\beta^*|=\beta^2-1$, which gives a wider space to choose from and a harder problem to be solved.

When Ali selects his secret key $x_A$, and also Basem makes the same thing to choose his secret key $x_B$, in this case Ali and Basem have $p^2-2$ choices to choose their secret key since $1 \leq x_A, x_B \leq |G_\beta^*|-1 = p^2 - 2$, and this property makes this group better than the classical one.

To be able to generate a common private key using Diffie-Hellman key exchange over the domain of gaussian integers, Ali and Basem must follow these steps:

1. First the partners select a gaussian prime $\beta$ of the from $\beta=3 \pmod{4}$.
2. Then they choose a generator $\theta \in G_\beta^*$.
3. Ali and Basem choose their secret keys $x_A$ and $x_B$ respectively, such that $1 \leq x_A, x_B \leq |G_\beta^*|-1=p^2-2$.
4. Ali computes his public key; $Y_A=\theta^{x_B} \pmod{p}$ and sends it to Basem. Similarly Basem computes his public key $Y_B=\theta^{x_A} \pmod{p}$ and sends it to Ali.
5. Every one of them computes the common private key $k$, Ali computes $k=(Y_B)^{x_A} \equiv \theta^{x_Ax_B} \pmod{p}$ and Basem computes $k=(Y_A)^{x_B} \equiv \theta^{x_Bx_A} \pmod{p}$.

Example 2:

Ali and Basem choose the gaussian $\beta=103$ to work over $G_{103}$. Then they choose a generator $\theta=(23+31i) \in G_{103}$. Now Ali choose his secret key $x_A=200$, and similarly Basem chooses his secret key $x_B=320$. Now Ali computes his public key $Y_A=(23+31i)^{200}=93+63i \pmod{103}$, and sends it to Basem. At the same time Basem computes his public key $Y_B=(23+31i)^{320}=93+63i \pmod{103}$ and sends it to Basem. At the same time Basem computes his public key $Y_B=(23+31i)^{320}=93+63i \pmod{103}$ and sends it to Ali. Now both of them compute the common private key $k$, Ali computes $k=(37+58i)^{200}=5+31i \pmod{103}$, Basem computes $k=(93+63i)^{320}=5+31i \pmod{103}$.

3 ElGamal Public key Cryptosystem:

The elgamal encryption scheme is typically described in the setting of the multilicative group $\mathbb{Z}_q^*$. 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 operation in $G$ should be relatively easy to apply for efficiency and discrete logarithm problem in $G$ should be computationally infeasible for the security of the protocol that uses the elgamal public-key cryptosystem.

1 Classical ElGamal public-key Cryptosystem:

The classical elgamal cryptosystem can be described as follows. Let $p$ be a large odd prime integer and let $\mathbb{Z}_p = \{0, 1, 2, 3, ..., p-1\}$. Then $\mathbb{Z}_p$ is a field under addition and multiplication modulo $p$. $\mathbb{Z}_p^* = \{1, 2, 3, ..., p-1\}$ is a cyclic group generated by some generator $\theta=1$ whose order is equal to $p-1$. That is,
every element to \( \mathbb{Z}_p^* \) is a power of \( \theta \).

**Lemma 1:**

Let \( n \) be an integer and \( \gamma \in \mathbb{Z}_n^* \). Then \( \gamma^{\varphi(n)-a} \equiv (\mod n) \).

**Proof:**

By Euler’s theorem \( \gamma^{\varphi(n)} \equiv 1 \pmod{n} \). Now multiplying both sides by \( \gamma^{-a} = (\gamma^{-1})^a \), we get that \( \gamma^{\varphi(n)-a} \equiv \gamma^{-a} \pmod{n} \).

**Algorithm 1 (Classical ElGamal algorithm)**

**First:**

1. Ali generates his public key by
   1. Generating a large prime \( p \) and generator \( \theta \) of \( \mathbb{Z}_p^* \).
   2. Selecting a random integer \( a \), \( 1 < a < p/2 \) and computing \( \theta^a \pmod{p} \).
   3. Ali’s public key is \( (p, \theta, \theta^a) \); Ali’s private key is \( a \).

**Second:**

To encrypt the message \( M \), Basem should do the following:
1. Obtains Ali’s authentic public key \( (p, \theta, \theta^a) \).
2. Select a random integer \( k \), \( 2 < k < p-2 \).
3. Compute \( \gamma \equiv \theta^k \pmod{p} \) and \( \delta \equiv M.(\theta^k)^k \pmod{p} \).
4. Send the ciphertext \( C = (\gamma, \delta) \) to Ali.

**Finally:**

To decrypt the message \( C \), Ali should do the following:
1. Use the private key \( a \) to compute \( \gamma^a \pmod{p} \) (Note: Ali can solve \( \gamma^{p-1-a} \pmod{p} \), since \( \gamma^{p-1-a} = \gamma^{\varphi(n)-p} \equiv \gamma^{-a} \pmod{n} \) by the above lemma).
2. Recovers the message by computing \( \gamma^{-a} \delta \equiv (\theta^k)^{-a}M.(\theta^a)^k \equiv M \pmod{n} \).

**Example 3:**

In order to generate the public key, Ali selects the prime \( p = 359 \) and a generator \( \theta = 124 \) of \( \mathbb{Z}_{359}^* \). Ali chooses the private key \( a = 292 \) and computes \( \theta^a = 124^{292} = 205 \pmod{359} \). Therefore, Ali’s public key is \( (p = 359, \theta = 124, \theta^a = 205) \) and Ali’s private key is \( a = 292 \). To encrypt the message \( M = 101 \), Basem selects a random integer \( k = 247 \) and computes \( \gamma = 124^{247} = 291 \pmod{359} \) and \( \delta = 101.205^{247} = 288 \pmod{359} \). Then Basem sends \( \gamma = 291 \) and \( \delta = 288 \) to Ali. Finally, Ali computes \( \gamma^{p-1-a} = 291^{66} = 216 \pmod{359} \) and recovers \( M \) by computing \( 216.288 = 101 \pmod{359} \).

2 ElGamal public-key cryptosystem in the domain of Gaussian integers \( \mathbb{Z}[i] \)

As in the case of the classical method, the first step is to choose a prime number, and in this case it must to be a Gaussian prime \( \beta \). If \( \beta = \pi \) where \( \pi = \pi \pi \pi \) is prime integer of the form \( 4k+1 \), then \( G_\pi = \{ \alpha : 0 \leq \alpha \leq q - 1 \} \equiv \mathbb{Z}_q^* \). This choice will be excluded since a large prime integer \( p \) of the form \( 4k+3 \)
so that $G_{\beta} \cong \mathbb{Z}[i]$.

Next we give the algorithm of the extended elgamal public-key cryptosystem in the domain of gaussian integers.

Algorithm 2:

(ElGamal public-key in the domain of gussian integers)

First:

Ali generates his public key:

1. Generates a large random gaussian prime integer $\beta$ of the form $\beta = 4k + 3$, where $p$ is an odd prime integer.
2. Finds a generator $\theta$ of the multiplication group $G_{\beta}^*$.
3. Select a random integer $a$, $2 \leq a \leq \phi(\beta) - 1 = p^2 - 2$.
4. Computes $\theta^a \pmod{\beta}$.
5. Ali’s public-key is $(\beta, \theta, \theta^a)$ and Ali’s private key is $a$.

Second:

To encrypt the message $M \in G_{\beta}$, Basem should do the following:

1. Obtain Ali’s authentic public key $(\beta, \theta, \theta^a)$
2. Select a random integer $k$, $2 \leq k \leq \phi(\beta) - 1 = p^2 - 2$.
3. Computes $\gamma = \theta^k \pmod{\beta}$ and $\delta = M \cdot (\theta^a)^k \pmod{\beta}$
4. Send the ciphertext $(\gamma, \delta)$ to Ali.

Finally:

To decrypt this message, Ali should do the following:

1. Receive $(\gamma, \delta)$ from Basem.
2. Use the private key $a$ to compute $\gamma^{-a} \equiv \gamma^{\theta(\beta)-a} \pmod{\beta}$
3. Recover the message $M$ by computing $\gamma^{-a} \cdot \delta \pmod{\beta}$

Example 4:

Consider the following example using artificially small parameters. In order to generate the public-key, Ali select the gaussian prime $\beta = 359$ and a generator $\theta = 1 + 11i$ of $G_{359}^*$. Then Ali chooses the private key $a = 86427$ and computes: $\theta^a \equiv (1 + 11i)^{86427} \equiv 323 + 295i \pmod{359}$. Thus Ali’s public key is $(\beta = 3359, \theta = 1 + 11i, \theta^a = 323 + 295i)$. To encrypt the message $M = 101$, Basem selects a random integer $k = 115741$ and computes $\gamma = (1 + 11i)^{115741} \equiv 149 + 117i \pmod{359}$ and $\delta = 101 \cdot (323 + 295i)^{115741} \equiv 147 + 209i \pmod{359}$. Then Basem sends $\gamma = 149 + 117i$ and $\delta = 147 + 209i$ to Ali. Finally, Ali computes $\gamma^{-a} \equiv (149 + 117i)^{42453} \equiv 117 + 178i \pmod{359}$, and recovers $M$ by computing $(117 + 178i) \cdot (147 + 209i) \equiv 101 \pmod{359}$.

Conclusion:

In this work when we extend the cryptosystem to the domain of gaussian integer, this extension make the cryptosystem more secure and very difficult to be broken but at the same time this extension need to provide potential and need a more development computer to accomplishment this work. Cryptosystems which have been discussed in this paper, with the infinity race between the cryptographers and the cryptanalysts, these methods and their extensions will not be the end. We hope that this thesis will be a step toward designing a cryptosystem that is more secure and very hard to be broken. Julius caesar (50 BC) is considered to be the first one who has designed a cryptosystem, as Julius caesar thought that his own cryptosystem was hard to be broken, nowadays his cryptosystem can be found in newspapers and magazines as puzzles for children to solve them. We may live to the day when we find this cryptosystems as puzzle or a computer game.
References


