DEFINITION OF MATHEMATICAL PROBLEM FOR CRYPTOGRAPHY USING DIRECT PRODUCT GROUP

D. Ali M. Sagheer[1] *and Nahidh Saleem* [2]

1)*Information System Department*, College of Computers, University of Anbar, Iraq. Email: ali_makki_sagheer@yahoo.com

2) Physics Department, College of Science, University of Anbar, Iraq.

المستخلص :

تعتمد انظمة التشفير على طريق مستمر من البصائر والطرق الجديدة من نظرية العدد، الهندسة الجبرية الحسابية، وفروع أخرى من الجبر. في هذه البحث نقترح مفهوماً جديداً في انظمة التشفير ذات المفتاح المعلن الذي يعتمد على زمرة الضرب المباشرالابيلية. زمرتين أبيلية يمكن أَن تستخدم الضرب المباشر لانشاء زمرة أبيلية من عمليات الإضافة و/ أَو الضرب. وتم اقتراح مسألة رياضية صعبة في الزمرة التي تم تعريفها اطلقنا عليها مسألة اللوغاريتِم المنفصل في زمرة الضرب المباشر. بعد ذلك صممنا انظمة تشفير ذات مفتاح معلن مستند على المسألة المقترحة. أيضاً الطريقة المقترحة تعطي أمنية مساوية لأمنية مسألة اللوغارتم المنفصل (DLP) مع حجم مفتاح أصغر، واصعب في الحل.

Abstract :

Cryptography depends on a continuing stream of new insights and methods from number theory, arithmetic algebraic geometry, and other branches of algebra. In this paper we proposes new concept in the public key cryptosystems that is depend on Direct Product Abelian Group. Two abelian groups can use direct product to construct an abelian group under addition and/or multiplication operations. There are a hard mathematical problem is proposed in the constructed group we call it Discrete Logarithm Problem of Direct Product Group. After that we design public key cryptosystems based on the suggested

problem. Also it appears to offer equal security for a far smaller bit size, with problem harder than DLP.

1. INTRODUCTION

Mathematics is amazing not only in its power and beauty, but also in the way that it has applications in so many areas. Group of numbers comprise a computational system within which one may clarify and study many kinds of mathematical problems. In this brief essay, we will describe the Direct Product Group of number system carefully and pose several computational problems for discussion. Then we will apply the system to develop the mathematical problem such as Discrete Logarithm Problem (DLP) of construction with straightedge and compass, we call it Discrete Logarithm Problem of Direct Product Group. Finally, we will apply the system to construct public key cryptosystems based on Direct Product Finite Field.

2. ALGEBRAIC PRELIMINARIES

As mathematicians, let us attempt to collect the core of these basic ideas in a useful definition, generalizing the notions of addition and multiplication of numbers. We do not attempt to define a set. However, we can attempt to be somewhat mathematically precise, and we describe our generalizing as functions rather than as rules. Recall that for any set S, the set S×S consists of all ordered pairs (a, b) for elements a and b of S.

Definition : a binary operation $*$ on a set S is a functions mapping S×S into S. for each $(a, b) \in S\times S$. we will denote the element $*((a, b))$ of S by $a * b$.

Definition : Let $*$ be a binary operation on S and let H be a subset of S. The subset H is closed under $*$ if for all a, b $\in$ H we also have $a * b$. In this case, the binary operation on H given by restricting $*$ to H is the induced operation of $*$ on H.

By our very definition of binary operation $*$ on S, the set S is closed under $*$.

Definition:- A binary operation $*$ on a set S is commutative if (and only if) $a * b = b * a$ for all $a * b \in S$.

Definition:- A binary operation on a set S is associative $(a * b) * c = a * (b * c)$ for all a, b, c $\in$ S.

Definition:- Let $(S, *)$ be a binary structure. An element e of S is an identity element for $*$ if $s * s = s * e = s$ for all $s \in S$.

Theorem: (Uniqueness of Identity Element) A binary structure $(S, *)$ has at most one identity element That is, if there is an identity element, it is unique.

Definition:- A group $(G, *)$ is a set G, closed under a binary operation $*$, such that following axioms are satisfied:

G_1: For all a, b, c $\in$ G, we have :

$(a * b) * c = a * (b * c)$ associativity of $*$.

G_2 : There is an element e in G such that for all $\chi \in G$,

$e * \chi = \chi * e = \chi$. identity element e for $*$.

G_3 : Corresponding to each $a \in G$, there is an element a' in G such that $a * a' = a' * a = e$. inverse a' of a.

Definition:- A group G is abelian if its binary operation is commutative.

Definition:- Let G be a group and let $a \in G$, Then the subgroup $\{a^n \mid n \in Z\}$ of G, is called the cyclic subgroup of G generated by a, and denoted by (a).

Definition:- An element a of a group G generate G and is generator for G if (a) = G. A group G is cyclic if there is some element a in G that generates G.

Theorem: (Theorem of Lagrange) Let H is Subgroup of a finite group G. Then the order of H is a divisor of the order of G.

Corollary: Every group of prime order is cyclic.

Theorem: order of an element of a group divides the order of the group.

3. DIRECT PRODUCTS GENERATED ABELIAN GROUPS

Let us a moment to review our present stockpile of groups. Starting with finite groups, we have the cyclic groups $Z_{n,}$ the symmetric groups S_n, of course we know that subgroup of these group exist. Turning to infinite groups, we have groups consisting of sets of numbers under the usual addition or multiplication, as for example, Z, R, and C under addition, and their nonzero element under multiplication, we have the U of complex numbers of magnitude 1 under multiplication, which is isomorphic to each of the groups R_C under addition modulo ${}_{C,}$ where ${}_{C}$ / R. We also have the groups S_A of all permutations of an infinite set A, as well as various groups formed from matrices.

One purpose of this section is to show a way to use known groups as building blocks to from more groups. The Klein 4- groups will be recovered in this way from the cyclic groups. Employing this procedure with the cyclic groups gives us a large class of abelian groups that can be shown to include all possible structure types for a finite abelian group. We start by generalizing.

Definition:- The Cartesian product of sets $S_{1,}$ $S_2, \dots S_n$ is the set of all ordered n-tuples $(a_1, a_2, \dots a_n)$, where $a_i \in S_i$ for $i = 1, 2, \dots, n$. The Cartesian product is denoted by either

$S_1 \times S_2 \times \dots \times S_n$ or by ΠS_i.

we could also define the Cartesian product of an infinite number of sets, but the definition is considerably more sophisticated and we shall not need it.

Now let it $G_{1,}$ $G_2, \dots G_n$ be groups, and let us use multiplicative notions for all the group operations. Regarding the G_i as sets, we can form, $\Pi_i^n{}_{=1} G_i$ Let us show that we can make $\Pi_i^n{}_{=1} G_i$ into a group by means of a binary operation of multiplication by direct products and finitely Generated Abelian Groups Components. Note again that we are being sloppy when we use the same notions for a group as for the set of element of the group.

Theorem Let $G_{1,}$ $G_2, \dots G_n$ be group For $(a_1, a_2, \dots a_n)$ and $(b_1, b_2, \dots b_n)$ in $\Pi_i^n{}_{=1} G_{i,}$ define $(a_1, a_2, \dots a_n)$ $(b_1, , \dots b_n)$ to be the element $(a_1b_1, a_2b_2, \dots a_nb_n)$ Then $\Pi_i^n{}_{=1} G_i$ is a group, the direct product of the groups G_i under this binary operation Groups.

Example:- Consider the group $Z_2 \times Z_3$, which has 2 . 3=6 elements, namely. (0, 0), (0, 1), (0, 2), (1, 0), (1, 1) and (1, 2). We clairn $Z_2 \times Z_3$ is cyclic. It is only necessary to find a generator. Let us try (1, 1). Here the operations is Z_2 and Z_3 are written additively, so we do the same in the direct product $Z_2 \times Z_3$.

$1(1, 1) = (1, 1)$
$2(1, 1) = (1, 1) + (1, 1) = (0, 2)$
$3(1, 1) = (1, 1) + (1, 1) + (1, 1) = (1, 0)$
$4(1, 1) = 3(1, 1) + (1, 1) = (1, 0) + (1, 1) = (0, 1)$
$5(1, 1) = 4(1, 1) + (1, 1) = (0, 1) + (1, 1) = (1, 2)$
$6(1, 1) = 5(1, 1) + (1, 1) = (1, 2) + (1, 1) = (0, 0)$

Theorem: $Z_m \times Z_n$ is cyclic and is isomorphic to Z_{mn} if and only if m and n are relatively prime, that is, the gcd of m and n is 1.
This section gives general introduction to group theory and basic definitions of group.

4. DIRECT PRODUCTS FINITE FIELD LAWS

The paper proposes a definition of a Direct Product Finite Field defined over several finite fields F_{q1}, F_{q2}, …. F_{qi}, denoted by DPF(F_{q1}, F_{q2}, …. F_{qi}). In this subsection, we shall introduce some of theoretic properties of the DPF(F_{q1}, F_{q2}, …. F_{qi}).

Defenition (Direct Product Finite Field): The (DPF(F_{q1}, F_{q2}, …. F_{qi}),+,·) is a finite field with the following group theoretic properties:
Let $a=(a_1, a_2,\ldots a_i)$, $b=(b_1, b_2,\ldots b_i)$, $c=(c_1, c_2,\ldots c_i)$, $a, b, c \in$DPF(F_{q1}, F_{q2}, …. F_{qi}), where, $a_1, b_1, c_1 \in F_{q1}$, $a_2, b_2, c_2 \in F_{q2}$, … and $a_i, b_i, c_i \in F_{qi}$.
Let we denote the Direct Product Finite Field DPF(F_{q1}, F_{q2}, …. F_{qi}) as DPF.

a:-Additive Properties

(1) Closure: $\forall a,b \in$ DPF, then $a+b \in$ DPF.
(2) Identity: $\forall a \in$ DPF, then $a+I_A=I_A+a=a$, so that $I_A=(0, 0,\ldots,0)$.
(3) Inverse: $\forall a \in$ DPF, then $a+(-a)=I_A$.

(4) *Associativity:* $\forall a,b,c\in$ DPF, then $(a+b)+c=a+(b+c)$.

(5) *Communicative:* $\forall a,b\in$ DPF, then $a+b=b+a$.

b:- *Multiplicative Properties*

(1) *Closure:* $\forall a,b\in$ DPF, then $a\cdot b\in$ DPF.

(2) *Identity:* $\forall a\in$ DPF, then $a\cdot I_M=I_M\cdot a=a$, so that $I_M=(1,1,\ldots,1)$.

(3) *Inverse:* $\forall a\in$ DPF, then $a\cdot a^{-1}=I_M$.

(4) *Associativity:* $\forall a,b,c\in$ DPF, then $(a\cdot b)\cdot c=a\cdot(b\cdot c)$.

(5) *Commutative:* $\forall a,b\in$ DPF, then $a\cdot b=b\cdot a$.

c:-*Addition and Multiplicative Properties*

(1) *Distributive:* $\forall a,b,c\in$ DPF, then $a\cdot(b+c)=a\cdot b+a\cdot c$.

(2) *No zero divisors:* if $a,b\in$ DPF and $a\cdot b=I_A$, then either $a_i=0$ or $b_i=0$.

Definition: Let a be an element of the group DPF and $a\neq I_A$. Then a is said to have order k if

$a^k=a\cdot a\ \ldots\ldots\ldots\cdot a=I_M$

k *product*

with $a^{k'}\neq I_M$ for all $1\leq k'\leq k$ (that is, k *is* the smallest integer such that $a^k=I_M$). If such a k exists, then, the subgroup of DPF is said to have finite order k, otherwise, it has infinite order.

Definition: From here on, for the multiplication operations on a DPF, for $k\in Z$, $a\in$ DPF, and $a\neq \boldsymbol{I_A}$, then:

$$\begin{array}{ll} a^k = a\cdot a\ \ldots\ldots\ldots\cdot a,\ (k\ \text{times}), & \text{for } k>0, \\ a^0 = I_M, & \text{and} \\ a^k = (a^{-1})^{-k}, & \text{for } k<0. \end{array}$$

Definition: The order of a DPF is defined as the number of element of DPF and denoted by #DPF.

If $a \in$ DPF is of order k, then

$$H = \{ a^i \mid 0 \leq i < k\text{-}1 \},$$

is a *subgroup* of DPF of order *k*.

Definition: Let $a \in$ DPF, and $a \neq \boldsymbol{I_A}$. Then a is said to generator element if

$$ord(a) = \#\text{DPF}$$

Then,

$$\text{DPF}^* = \{ a^k \mid 0 \leq k < \#\text{DPF} \text{ -1} \},$$

5. DISCRETE LOGARITHM PROBLEM OF DIRECT PRODUCT GROUP (DLPDPG)

One of the most interesting open problem in cryptography is the realization of a trapdoor on the discrete logarithm, in which to solve the DLP is hard only if published parameters are used, while it is easy by using a secret key (trapdoor key) [4].

The DLP can be defined on various finite groups as well as multiplicative group over a finite filed F_q [5], this idea can be extended to arbitrary groups and, in particular, to Direct Product Group. A typical example except the multiplicative group is the discrete logarithm problem on Direct Product elements over F_q.

Consider the equation $b=a^k$, where a and b are two elements in the DPF and k is an integer. It is relatively easy to calculate b given a and k, but determining the integer k from a multiple of the element a^k, even with the knowledge of a, b and DPF is a very difficult problem, we call it the (DLPDPG).

Definition (DLPDPG): For a Direct Product Finite Field DPF, let a, $b \in$ DPF, recall that in the DLPDPG to find an integer $k \in Z$, is such that $a^k = b$.

Since a Direct Product Finite Field DPF is made into Abelian group by a direct product operation. The exponential of an element on DPF actually refers to the repeated multplications. Therefore, $b=a^i$ is the i^{th} power of $a\in$DPF is the i^{th} multiple of a. The logarithm of b to the base a would be i (i.e. the inverse of exponentiation under DPF). The DLPDPG is of interest because its apparent intractability forms the basis for the security of new cryptographic schemes.

6. THE PROPOSED CRYPTOSYSTEMS

This section proposes cryptosystems that employs Direct Product Finite Field. It does not invent new cryptographic algorithm, but it is to implement existing public-key cryptosystem using Direct Product Group. The proposal is an analogues to the Diffie-Hellman key exchange protocol, analogues to ElGamal, Massey-Omura schemes. The modular multiplication operation for element at DPF in the proposed system is the counterpart of modular multiplication in RSA and ElGamal, and exponentiation of element in DPF is the counterpart of the modular exponentiation. To form cryptographic system using Direct Product Group, we need to find a "hard problem" corresponding to the difficulty of factoring the product of two prime or taking the discrete logarithm or elliptic curve discrete logarithm.

6.1 Exponentiation over DPF

The fundamental operation in cryptographic schemes is that the exponentiation of an element by an integer. If not the most confusing term, certainly the idea of multiplying element refers to computing $b=a^k$, where a and b are two elements in the DPF, group and k is an integer. This really means that we multiply a to itself k times.

Definition (Exponentiation of a element in DPF by an integer):

Given $k \in Z$, and a is an element belong to DPF, then:

$$a^k = a \cdot a \cdot \ldots\ldots\ldots \cdot a \qquad (k \ \text{times}) \ldots\ldots\ldots\ldots\ldots (1)$$

It is the dominant cost operation in the cryptographic schemed, and it dominates the execution time of cryptographic schemes, especially the representation of DLPDPG.

The algorithm that can be used to compute the element exponentiation in DPF, group is Repeated-Squaring and Multiplication or fast group operation Method.

6.2 Fast Group Operation Method

The most fundamental computation on DPF group is the multiplication operation as shown in equation(1) with k are very large positive integer, since the computation of a^k is so fundamental in all elements related computations and applications, it is desirable that such computations are carried out as fast as possible.

Remarkably enough, the idea of repeated squaring for fast exponentiation can be used almost directly for fast group operation on DPF.

Let $e_{n-1}\, e_{n-2} \ldots\ldots e_1\, e_0$ be the binary representation of k. then for i starting from n-1 down to 0 (e_{n-1} almost 1 and used for initialization), check whether or not e_i =1. If e_i =1, then perform a squaring and a multiplication group operation; otherwise, just perform a squaring operation. For example: compute a^{85}, since 85= 1010101, we get the following table:

Table 1 Compute a^{85} using fast group operation method

I	e_i	*value*	*Operations*	Status
6	e_6	1	a	Initialization
5	e_5	0	$a^2 = a^2$	Squaring
4	e_4	1	$^{(}a^{2)2}\cdot a = a^5$	Squaring and Multiplication
3	e_3	0	$((a^2)^2\cdot a)^2 = a^{10}$	Squaring
2	e_2	1	$(((a^2)^2\cdot a)^2)^2\cdot a = a^{21}$	Squaring and Multiplication
1	e_1	0	$((((a^2)^2\cdot a)^2)^2\cdot a)^2 = a^{42}$	Squaring
0	e_0	1	$(((((a^2)^2\cdot a)^2)^2\cdot a)^2)^2\cdot a = a^{85}$	Squaring and Multiplication

We have the following algorithm which implements this idea of repeated squaring and multiplication (fast group operation) for computing a^k, that is, it reduces the complexity of the computation of a^k from k to $\log k$.

Algorithm (Fast Group Operation)

Input: an element $a \in$ DPF and positive integer k.

Output: an element $b = a^k$.

```
1. Write $k$ in the binary expansion form $k = e_{n-1}\ e_{n-2}\ \ldots\ldots\ e_1\ e_0$ (Assume $k$ has $n$ bits)
2. Set $b = I_M$.
3. Compute $a^k$:
    3.1 For $i$ from $n$-1 down to 0 do
    3.2 $b = b^2$.
    3.3 if $e_i = 1$, then $b = b\cdot a$.
4. Output $b$: (now $b = a^k$).
```

7. PUBLIC-KEY CRYPTOSYSTEMS BASED ON DLPDPG

For any cryptographic system based on the DLP, there is an analogy to direct product finite field. In what follows, it'll introduce complex cryptosystems analogues to three widely used public-key cryptosystems, namely Diffie-Hellman key exchange system, the Massey-Omura, the El-Gamal public-key cryptosystems.

7.1. Analogy of the Diffie-Hellman Key Exchange System

This system is merely a method for exchanging keys; no messages are involved. Alice and Bob first publicly choose DPF. Then they publicly choose an element $b \in$DPF to serve as their "Base element". It is a generator of the key. To generate a key, Alice chooses random integer e between 1 and #DPF, and keeps it secret. She then computes $b^e \in$DPF and makes that public. Bob chooses his own secret random integer d between 1 and #DPF, and makes public $b^d \in$DPF. The secret key is then $b^{ed} \in$DPF. Both Alice and Bob can compute this key. For example, Alice knows b^d (public knowledge) and her own secret e. Charlie, on the other hand, only knows b, b^e and b^d. Without solving the DLPDPG, (finding d knowing b and b^d), there is no way for him to compute b^{ed} only knowing b^e and b^d. The following algorithm illustrates this manner.

Algorithm (Diffie-Hellman key exchange system with DLPDPG)

1. Initialization

- Alice and Bob publicly choose a direct product finite field DPF.
- They publicly choose a random "Base element" $b \in$DPF such that b generates a large subgroup of DPF.

2. Key generation

- Alice chooses a secret random integer *e*. She then computes $b^e \in$DPF.
- Bob chooses a secret random integer *d*. He then computes $b^d \in$DPF.
- Make b^e and b^d public and keep *e* and *d* secret.

3. Calculation of the secret key b^{ed}

- Alice computes the secret key $b^{ed}=(b^d)^e$.
- Bob computes the secret key $b^{ed}=(b^e)^d$.

There is no known fast way to compute b^{ed} if only knows *b*, b^e and b^d, which is DLPDPG.

6.2. Analogy of the Massey-Omura Cryptosystem

In this system the direct product finite field DPF have been made publicly known. Alice and Bob both select a random integer e_1 and e_2 between 1 and #DPF respectively with $gcd(e_1,$ #DPF$)=1$ and $gcd(e_2,$ #DPF$)=1$. They also compute their inverses $d_1 = e_1^{-1}$ *mod* #DPF (ie. $d_1e_1=1$ *mod* #DPF) and $d_2 = e_2^{-1}$ *mod* #DPF (ie. $d_2e_2=1$ *mod* #DPF), then, keep everything secret. If Alice wants to send the message *P* (i.e. PlainText, we represent the message as an element in DPF; each equivalent to element denoted by *P*) to Bob, she first sends him the message P^{e1}. This means nothing to Bob, since he does not know d_1. He ever, he can exponentiate it by his e_2 and send the message P^{e1e2} back to Alice. Then Alice can help unravel the message by exponentiating this new message by d_1 which sends $P^{e1e2d1} = P^{e2}$ back to Bob. Then Bob can exponentiate this message by d_2 to get the original message ($P^{e2d2}=P$). During this process Charlie sees P^{e1}, P^{e2}, and P^{e1e2}.

Without solving the DLPDPG –finding e_2 and then its inverse knowing P^{e1} and P^{e1e2}- there is no way for him to find *P*. The following algorithm illustrates this manner.

Algorithm (Massey-Omura Cryptosystem with DLPDPG)

1. **Initialization**
    - Alice and Bob publicly choose a complex finite field DPF.
    - They publicly known the order number of the DPF denoted by *N*.

2. **Key generation**
    - Alice chooses a secret random integer e_1 between 1 and *N*, such that $gcd(e_1, N)=1$. She then computes its inverse $d_1=e_1^{-1}$ *mod N.*
    - Bob chooses a secret random integer e_2 between 1 and *N*, such that $gcd(e_2, N)=1$. He then computes its inverse $d_2=e_2^{-1}$ *mod N.*
    - Keep e_1, d_1, e_2, and d_2 secret.
3. **Transmission procedure**

    Alice sends the message *P* to Bob as follows:

    - Alice computes P^{e1}, and sends it to Bob.
    - Bob computes P^{e1e2}, and sends it to Alice.
    - Alice computes $P^{e1e2d1} = P^{e2}$, and sends it to Bob.
    - Bob computes $P^{e2d2}= P$.

6.3. Analogy of the ElGamal Cryptosystem

In this system the direct product finite field DPF, and the “Base element” $b \in$DPF are public information. Bob randomly chooses an secret integer d ($1<d<\#DPF$), and publishes the element b^d. If Alice ants to send the message P_c (i.e. PlainText, we represent the message as an element in DPF; each equivalent to element denoted by P) to Bob, she will choose a secret random integer e ($1<e<\#DPF$) and send $(P \cdot b^{ed}, b^e)$ to Bob. Bob will then exponentiate the second element in the pair by d to get b^{ed}, the compute the inverse of the key b^{ed} to get $(b^{ed})^{-1}$ and multiply by the first element in the pair $P \cdot b^{ed}$ to find P. In the meantime, Charlie has only seen b^e and b^d. Without solving the DLPDPG (eg. finding d knowing b and b^e), there is no way for him to find P. The following algorithm illustrates this manner.

Algorithm (ElGamal Cryptosystem with DLPDPG)

1. Initialization

- Alice and Bob publicly choose a direct product finite field DPF.
- They publicly choose a random “Base element” $b \in$DPF such that b generates a large subgroup of DPF.

2. Key generation

- Bob chooses a secret random integer d in interval [2, #DPF].
- He then computes $Q=b^d$.
- Make Q public and keep d secret.

3. Encryption

Alice sends the message P to Bob as follows:

- Select random integer e in interval [2, #DPF].
- Compute b^e.
- Compute $K=Q^e$, (i.e. $K=b^{de}$).
- Compute ciphertext $C= P \cdot K$.

- Transmit the pair elements (C, b^e).

4. Decryption

Bob retrieves the message as follows:

- Compute $K= (b^e)^d$, (i.e. $K=b^{ed}$).
- Compute K^{-1}, then multiply K^{-1} with the ciphertext C:

$$P=C\cdot K^{-1}.$$

8. IMPLEMENTATION

The proposed system us programmed by MatLap Version 7 programming language on P4 PC with CPU of 3 G.B and RAM of 2 G.B. Then the methods is applied on different size messages, which takes plaintext and devided into blocks each block corrispond to element containd in DPF and computes the running time of the encryption and decryption of each messages. In this examples we take DPF as the direct product finite fields DPF(F_{263},F_{347}), the order of the prime field F_{263} is 263-1, the order of the prime field F_{347} is 347-1, then order of DPF is (263-1)*(347-1)= 90652 or its divisors.

A. Diffie-Hellman key exchange system with DLPDPG

1. Initialization

- Alice and Bob publicly choose a direct product finite fields DPF(F_{263}, F_{347}).
- They publicly choose a random "Base element" $b= (47, 19)\in$DPF such that b generates a large subgroup of DPF, the ord(b)= 45326.

2. Key generation

- Alice chooses a secret random integer e=5832. She then computes $b^e(47,19)^{5832}=(143,300)$.
- Bob chooses a secret random integer d=26481. He then computes $b^d=(47,19)^{26481}=(202,17)$.

- Make b^e and b^d public and keep e and d secret.

3. Calculation of the secret key b^{ed}

- Alice computes the secret key $b^{ed}=(b^d)^e=(202,17)^{5832}=(176, 29)$.
- Bob computes the secret key $b^{ed}=(b^e)^d=(143,300)^{26481}=(176, 29)$.

B. Massey-Omura Cryptosystem with DLPDPG

1. Initialization

- Alice and Bob publicly choose a direct product finite fields DPF(F_{263}, F_{347}).
- They publicly known the order number of the of DPF(F_{263}, F_{347}) denoted by $\#DPF=90652$.

2. Key generation

- Alice chooses a secret random integer $e_1=31831$, such that gcd(31831,90652)=1. She then computes its inverse $d_1=e_1^{-1}$ $mod\ \#DPF=31831^{-1}$ mod $90652=74735$.
- Bob chooses a secret random integer $e_2=8197$, such that gcd(8197,90652)=1. He then computes its inverse $d_2=e_2^{-1}$ $mod\ \#DPF=8197^{-1}$ mod $90652=54765$.
- Keep e_1, d_1, e_2, and d_2 secret.

3. Transmission procedure

Alice sends the message $P=(100,200)$ to Bob as follows:

- Alice computes $P^{e1}=(100,200)^{31831}=(44, 203)$, sends it to Bob.
- Bob computes $P^{e1e2}=(44,203)^{8197}=(54,240)$, sends it to Alice.
- Alice computes $P^{e1e2d1}=(54,240)^{74735}=(64,227)=P^{e2}$, sends it to Bob.
- Bob computes $P^{e2d2}=(64,227)^{54765}=(100,200)=P$.

C. ElGamal Cryptosystem with DLPDPG

1. **Initialization**

- Alice and Bob publicly choose a direct product finite fields DPF(F_{263}, F_{347}).
- They publicly choose a random “Base element” $b=(135,23)\in$ DPF(F_{263}, F_{347}) such that generates a large subgroup of DPF, the ord(b)= 45326

2. **Key generation**

- Bob chooses a secret random integer d=943.
- He then computes $Q=b^d=(135,23)^{943}=(85,32)$.
- Make Q public and keep d secret.

3. **Encryption**

Alice sends the message P=(100,200) to Bob as follows:

- Select random integer e=32741.
- Compute $b^e=(135,23)^{32741}=(177,226)$.
- Compute $K=Q^e=(b^d)^e=(85,32)^{32741}=(55,312)$ (i.e. $K=b^{de}$).
- Compute ciphertext $C=P\cdot K=(100,200)\cdot(55,312)=(240,287)$.
- Transmit the pair complexes (C, b^e)=[(240,287), (177,226)].

4. **Decryption**

Bob retrieves the message as follows:

- Compute $K=(b^e)^d=(177,226)^{943}=(55,312)$, (i.e. $K=b^{ed}$).
- Compute $K^{-1}=(55,312)^{-1}=(110,228)$ under DPF.
- Multiply K^{-1} with the ciphertext C:

$P=C\cdot K^{-1}=(240,287)\cdot(110,228)=(100,200)$.

8. THE COMPUTATIONAL COMPLEXITY

The Computational Complexity of the El-Gamal encryption/decryption algorithms using DLP compared to proposed problem DLPDPG is as follows:

1. **Encryption/Decryption using DLP:**
   - Let the size of the maximum order is n.
   - The complexity of the computing $q=b^e$ is:
     $T(Q)=T(b^e)=O(\log n)$ arithmetic (multiplication) operation, using Fast Exponential Algorithm [6].
     Then,
     $T(Q)=O(\log^3 n)$ bit operation.
     Also, $T(k)=O(\log^3 n)$ bit operation.
   - The complexity of the computing ciphertext $c=m*k$ is:
     $T(c)=T(m*k)=O(\log^2 n)$ bit operation, for each encryption block, suppose there are 100 blocks of message; then, $100T(c)=100T(m*k)=O(100\log^2 n)$.
     The overall Complexity is $O(100\log^2 n)+O(2\log^3 n)$.
2. **Encryption/Decryption using DLPDPG:**
   - Let the size of the maximum order of the finite field is n.
   - The complexity of the computing $Q=b^e$ is:
     $T(Q)=T(b^e)=O(2\log n)$ group operation, using Fast group operation method.
     Then,
     $T(Q)=O(2\log^3 n)$ bit operation.
     Also, $T(K)=O(2\log^3 n)$ bit operation.
   - The complexity of the computing ciphertext $C=P\cdot K$ is:
     $T(C)=T(P\cdot K)=O(2\log^2 n)$ bit operation, for each encryption block, suppose there are 100 blocks of message (are represented as 50 pairs of each pair correspond to complex number); then, $50T(C)=50T(P\cdot K)=O(100\log^2 n)$.
     The overall Complexity is $O(100\log^2 n)+O(4\log^3 n)$.

Finally, the computational complexity of the implementation of encryption/decryption function is same approximately for DLP and DPGDLP.

9. THE RUNNING TIME COMPARISON

The running time of the El-Gamal method with DLP over F_q and DLPDPG over DPF and the base element is obtained in the following Table 1. The order of the DPF(F_{q1}, F_{q2}) group of base element is (q_1-1) (q_2-1), then, the order of the (F_{72139}) group of base element 2998+3213i is q^2-1=24196560. The DLPDPG over F_{72139} is solved by 16 msec.

Therefore, we conclude the order of the DPF(F_q) group of base element is q^2-1 or its factors. The following table explain the running time of solving DLP over F_{4919}, F_{59011}, F_{699367}, $F_{2099221}$, and $F_{9999991}$, and DLPDPG over (F_{263}, F_{347}) , (F_{4919}, F_{4931}), (F_{59011}, F_{59021}), (F_{699367}, F_{699373}), ($F_{2099221}$, $F_{2099249}$), and ($F_{9999991}$, $F_{9999973}$).

Table 1 Running Time of solving DLP and DLPDPG in *msec*

Finite Field F_q	**DLP**			**Finite Field DPF**	**DLPDPG**		
	Base Number	Power	Running Time		Base element	Power	Running Time
F_{263}	81	241	0	F_{263}, F_{347}	(217, 321)	10193	172
F_{347}	67	319	0	F_{263}, F_{347}	(17, 63)	41392	719
F_{4919}	4593	3931	0	F_{4919}, F_{4931}	(2873, 1639)	7463293	137469
F_{59011}	55578	53934	16	F_{59011}, F_{59021}	(46483, 32983)	736549202	281860
F_{699367}	79521	679773	32	F_{699367}, F_{699373}	(327351, 82736)	41917354369	432170
$F_{2099221}$	1928733	2058253	406	$F_{2099221}$, $F_{2099249}$	(736422, 97462)	864523231317	782736
$F_{9999991}$	9128453	9888888	1860	$F_{9999991}$, $F_{9999973}$	(8234683, 3417832)	3435265289130	974320

There is a clear growth of the time execution when use the complex group DPF and increase as long as the finite field size is increased. This increasing with small numbers, what is happen when a large number is applied, such as 100 digit number, 200 digit number or more, the complexity is increased rapidly, show Figure 1.

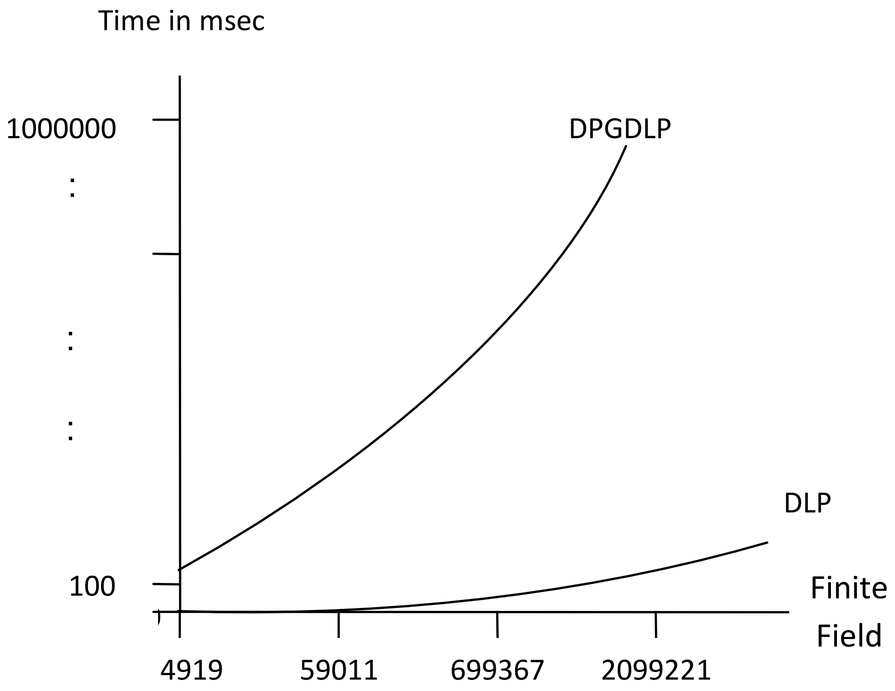

Figure 1: The complexity evaluation of DLP and DLPDPG

10. SECURITY OF PROPOSED CRYPTOSYSEMS

The complication associated with proposed cryptosystems comes from the wide variety of possible group structures of the element in the DPF and from the fact that modular operation under DPF is somewhat more complicated than classical modular multiplication.

The security of systems depends on how difficult it is to determine the integer d, given the complex number b and the complex a^d where $b=a^d$ mod q. This is referred to as the DLPDPG. Also that it appears to offer equal security for a far smaller bit size, because the size (order) of the DPF appears at most q^2 -1, that means the calculation is applied with q-bit size, while to solve the DLPDPF needs q^2-1 operations.

Therefore, the cryptanalyzer need to analysis and solve the DLPDPF to cryptanalysis the public based on it.

11. CONCLUSION

The project defined the DPF that proved as an Abelian group to use it in the proposed cryptosystems. Then, discover that the DPF group has a one way function similar to DLP and ECDLP, which DLPDPG. The construction of cipher system is based on the difficulty of solution of the DLPDPG that is a clear change in the cryptography and opens new windows for treatment with special group and new operations. There is a computational advantage in using the proposal with the shorter key length that reduces the overall calculations with secure system. The DLPDPG appears more complicated than DLP, because the element operations increase the complexity as long as the size is increased.

The DLPDPG over Direct Product Finite Field (F_{q1}, F_{q2}, …. F_{qi}) is more intractable than the DLP in F_q. It is this feature that makes cryptographic system based on the DLPDPG even more secure than that based on the DLP, because the DPF gives a large group over small field size. Since the group DPF(F_q, F_p) of order $q.p$-1 or its factors, therefore, some of the strongest algorithms for solving DLP cannot be adaptive to the DLPDPG.

REFERENCES

[1] Minhyong Kim, Why everyone should know number theory, Department Of Mathematics, University Of Arizona, Tucson, AZ85721, April, 1998.

[2] H. Anton, I. Bivens and S. Davis, *Calculus*, John Wiley & Sons Inc., 2002.

[3] R. L. Finney and G. B. Thomas, *Calculus*, Addison Wesley Pub., 1990.

[4] M. B. Nathanson, *Elementary Methods in Number Theory*, Graduate Text in Mathematics 195, Springer-Verlag, 2000.

[5] R. A. Mollin, *Number Theory and Applications*, NATOASI series 1989.

[6] S.Y. Yan, *Number Theory for Computing* , Springer-Verlag, 2000.

[7] J. M. Kizza, Computer Network Security, Springer Inc., 2005.

[8] N. Koblitz, *A Course in Number Theory and Cryptography*, Springer-Verlag, 1987.

[9] W. Stallings, *Cryptography and Network Security, Principle and Practice*, Addison Wesley, 1999.

[10] M. Stamp, *Information Security Principles and practice*, JohnWiley & Sons, Inc., 2006.

[11] G. Williams, *Linear Algebra with Applications*, 4th edition, Jones and Bartlett Publishers, Inc., 2001.