ON USING THE SYMMETRIC GROUP IN A CRYPTOSYSTEM

Jamal A. Othman
Iraq Commission for computer & informatics. Baghdad-Iraq.

Abstract:

This paper propose to overcome the negative point of the cryptosystem based on the symmetric group that had been suggested by (Doliskani,Ehsan and Zakerolhosseini)[1], which is the relatively large memory and bandwidth requirements for storing and transmitting permutations (symmetric group elements), our suggestion is to get benefit from the bijection between permutations and integers and modify the way we encrypt plaintext (m) from converting m to a permutation (as it is in there suggestion) to converting the permutation $(g^a)^k$ (which used to encrypt the plaintext (m) to an integer and then the cipher text ($m.(g^a)^k$) will be an integer and not a permutation which can be decrypted by multiply it with the number $((g^a)^k)^{-1}$ .By such modification we get the benefits of using the symmetric group elements in our public key such as non commutative, high computational speed and high flexibility in selecting keys which make the Discrete Logarithm Problem (DLP) resistant to attacks by algorithms such as Pohlig-Hellman.

Key words:
Public key cryptosystem, Discrete logarithm Problem (DLP), integer factorization problem (IFP), Permutation group, Factorial base number.

حول أستخدام زمره التناضر في نظام التشفير

جمال أحمد عثمان
الهيئه العراقيه للحاسبات والمعلوماتيه. بغداد–العراق.

الملخص

يحتاج نظام التشفير بأستخدام زمره التناضر المقترح من قبل الساده (دولسكاني ،أحسان وذاكر الحسيني)[1] لحجم ذاكره وعرض حزمه كبيرين نسبيا لغرض تشفير ومن ثم تناقل البيانات المشفره بموجب هذه الطريقه. يهدف البحث الى تجاوز هذا الجانب من خلال الاستفاده من التقابل الموجود بين عناصر زمره التناضر والاعداد الصحيحه لذلك أقترحنا في هذا البحث تحويل المفتاح العام $(g^a)^k$ لدى المستلم من عنصر من عناصر زمره التناضر الى عدد صحيح واستخدامه في التشفير بدل من تحويل الرساله (m) من عدد صحيح الى عنصر في زمره التناضر كما هو مقترح من قبل الساده (دولسكاني ،أحسان وذاكر الحسيني) وتكون الرساله المشفره التي يتم ارسالها($m.(g^a)^k$) عدد صحيح وليس عنصر من عناصر زمره التناضر يحوي عدد من الارقام اقل من عددها في حال تحويل مقطع الرساله (m) الى عنصر في زمره التناضر كما هو مقترح من قبلهم، بعد استلام مقطع الرساله من الطرف الاخر يتم فتح التشفير بضربه بالعدد $((g^a)^k)^{-1}$ فنحصل على الرساله (m). وبهذا نحصل على فائده استخدام زمره التناضر في نظام التشفير المتمثله بكونها زمره غير تبادليه يتم اختيار المفتاح فيه بمرونه ومقاومه للخورزميات المعروفه التي تستخدم في محاولات فك تشفير انظمه التشفير التي تعتمد على الزمر بمساحه ذاكره وعرض حزمه أصغر.

1. Introduction

There are two distinct paradigms for approaching security. One is the symmetric or private-key path, which can be viewed as a descendant of the very early attempts of security. Any such system assumes that, if two individuals wish to communicate securely, they must both possess a unique common secret key, which is used for both encryption and decryption. While many such systems have evolved over time, it is the difficulties of sharing and distributing the common key and keeping it a secret that leads to their vulnerability and the need for a new paradigm. By the early 1970's a significantly new and different perspective on security began to take shape contributing to the emergence of public-key cryptosystem. With public-key cryptosystem (PKC) each user has a specific private-key which only the user knows and a mathematically-related public-key which can be made public and freely distributed. Based upon the mathematical problem we can distinguish between two main kinds of public key cryptosystem, the first depend upon the intractability of the discrete logarithm problem (DLP) while the second depend upon the intractability of integer factorization problem (IFP).The cryptosystem proposed by (Doliskani, Ehsan and Zakerolhosseini) is of the first kind i.e. it depend upon the intractability of the discrete logarithm problem and it is a symmetric group based cryptosystem, we try in this paper to overcome some of the negative points in these proposed cryptosystem which they are the relatively large memory and bandwidth requirements. In the coming sections we will speak first briefly about subjects related to our work's so we will speak about public-key cryptosystem (PKC), the discrete logarithm problem (DLP), integer factorization problem (IFP), and the main topics in which our proposed modification are the symmetric group based cryptosystem and representing a symmetric group elements by integers and then we try to explain through an example in the last section the cryptosystem proposed by (Doliskani, Ehsan and Zakerolhosseini) and our proposal to modify the cryptosystem in the same example.

1.1 Public-Key cryptosystem

With Public-Key cryptosystem (PKC), each user has a specific private-key which only the user knows and a mathematically-related public-key which can be made public and freely distributed. Based upon the mathematical problem, we can distinguish between two main kinds of public key cryptosystem, the first depend upon the intractability of the Discrete Logarithm Problem (DLP) while the second depend upon the intractability of Integer Factorization Problem (IFP).

The first published work on Public Key cryptosystem was in a groundbreaking paper by Whitfield Diffie and Martin Hellman [2] titled "New Directions in Cryptography" in November, 1976. The paper described the key concepts of PKC .This paper revolutionized the world of cryptography and galvanized dozens of researchers around the world to work on practical implementations of a public key cryptography algorithm. It was the first practical method for establishing a shared secret over an unprotected communications channel in Diffie-Hellman key exchange, a finite field $GF(p)$ and a generator $g \in GF(p)$ are chosen and made public. Suppose that two users "A" and "B" wish to agree upon a key. User "A" selects a random integer $2 \leq x \leq p$-2, and transmits g^x to "B" over a public channel. User "B" also selects a random integer $2\leq y \leq p$-2, and transmits g^y to "A". The users "A" and "B" having common key g^{xy}, compute $(g^x)^y = g^{xy}$ and $(g^y)^x = g^{xy}$ respectively. Another important public key encryption algorithm is the ElGamal(3) encryption system which is an asymmetric key encryption algorithm for public-key cryptography based on the Diffie–Hellman key exchange. It was described by Taher Elgamal in 1985. The algorithm works as follows: User "A" selects a finite field GF (q) and a generator $g \in$ GF(q), and an integer x then he publishes (g, g^x) as the public-key and keeps x secret. User "B", who requires to send a message $m \in$ GF (q) to "A", selects an integer y, $2 \leq y \leq q-2$ randomly, computes $m.(g^x)^y = m.g^{xy}$, and sends the pair (g^y, $m.g^{xy}$) to "A". User "A" who knows x, recovers m by computing $m.g^{xy}(g^y)^{-x} = m.g^{xy}.\ g^{-xy} = m$. Finding an efficient discrete logarithm algorithm (DLP) would make this system unsecure, since g , g^x and g^y are all known, and x, y or g^{xy} can be computed. Another way for breaking this scheme is to compute g^{xy} from g^x and g^y , without computing either x or y. The best known attack against the above schemes, the index calculus method, is sub-exponential in nature. Elliptic Curve Cryptosystem (ECC) is another important public key cryptosystem, the first elliptic curve scheme was proposed by Koblitz (4) and Miller (5) independently, it is based on a group of points on an elliptic curve which are defined over a finite field. There is no sub exponential-time algorithm that could solve the Discrete Logarithm Problem (DLP) in these groups. On the other hand, the best known attack against Elliptic curve cryptosystems is exponential in nature. So the use of an elliptic curve group that is smaller in size maintains same level of security and offers potential reductions in bandwidth, storage, processing power, electrical power and message sizes.

The above cryptosystems are group based and depend upon the discrete logarithm problem (DLP). There are other important cryptosystem schemes which depend upon intractability of integer factorization (The integer factorization (IF) problem), RSA (Rivest, Shamir, Adleman)(6) is the most used one of them. It is based on the product of two prime numbers .The keys for the RSA algorithm are generated as follows: Choose two distinct uniformly at random prime numbers p and q Compute $n = pq$ (n is used as the modulus for both the public and private keys) Compute $\varphi(pq) = (p-1)(q-1)$. (φ is Euler's totient function), Choose an integer e such that $1 < e < \varphi(pq)$, e is released as the public key, Determine $d = e^{-1}$mod $((p-1)*(q-1))$, it will satisfy $d * e \equiv 1 \; mod \, \varphi(pq)$ and d is kept as the private key. User "A", transmits his public key (n, e) to User "B", and keeps the private key d secret. If User "B" wishes to send message m to User "A", He first turns m into an integer $< n$, He then computes the cipher text $c = m^e \; mod \, n$, to decrypt user “A" recover m from c by using his private key d throw the computation $m = c^d \; mod \, n$.

1.2. The Discrete Logarithm Problem (DLP)

The security of the group based cryptosystems depends on the Discrete Logarithm Problem (DLP), Let G denotes a cyclic group of order n and $\alpha \in G$ be a generator of G with $\beta \in G$. The discrete logarithm of β to the base α that denoted by $\log_\alpha \beta$ is an integer $0 \le x < n$, such that $\beta = \alpha^x$ the discrete logarithm problem can be stated as follows: Given $\beta \in G$, find an integer x that satisfies $\beta = \alpha^x$. The Discrete Logarithm Problem (DLP) considered being difficult (no efficient algorithms are known for non-quantum computers) but continuous improvements in computer processing power have increased the scope of the attacks and many algorithms developed to break the different cryptosystems security for the group based cryptosystem The simplest algorithm, the brute-force search is to compute $1, \propto^1, \alpha^2$ successively until β is Obtained. This algorithm will clearly find x. However, since it requires $O(n)$ of group operations, it would be inefficient for large n's. A faster algorithm is the baby-step giant-step algorithm (7) having a running time and memory requirement of $O(\sqrt{n})$, therefore it is a time-memory trade-off of the brute-force search method (8). An appropriate data structure for the implementation of this algorithm (brute-force search method) can be found in (9).

Pollard's r-algorithm is another algorithm (10). The expected running time of this algorithm is equal to the baby-step giant-step method, but its memory requirement is negligible. The Pollard's r-method can be parallelized so the expected number of steps required by each processor for the calculation of the discrete logarithm becomes $O(\sqrt{n}/t)$. The Pohlig-Hellman algorithm introduced by Pohlig and Hellman (11), is an algorithm that takes advantage of the factorization of order n of the group G, if $n = p_1^{\delta 1} p_2^{\delta 2} \ldots . p_k^{\delta k}$ where $\delta_i > 0, i = 1,2,\ldots.,k$ then the execution time of this algorithm is of $O(\sum_{i=1}^{k} \delta_i (\log n + \sqrt{p_i}))$ so if the order of the group is smooth integer then the algorithm is computationally efficient. The most efficient algorithm known to the date for solving the discrete logarithm over finite fields is the index-calculus algorithm also uses the idea of smooth numbers. It selects a small $B \subseteq G$, which is called the factor base, in such a way that a relatively large subset of elements of G can be expressed as the products of elements of B. Then the logarithms of elements of B are computed as follows:

1) Compute α^i where i is a random integer such that $0 \le i \le n-1$

2) If α^i can be expressed as $\alpha^i = \prod_{k=1}^{|B|} p_k^{\delta_k}$ where $\delta_k \ge 0$, $p_k \in B$ then taking logarithms of both sides we obtain $i = \sum_{k=1}^{|B|} \delta_k \log_\alpha p_k$, $\delta_k \ge 0$, $p_k \in B$ …… (1)

3) Repeat steps 1 and 2 till enough set of equations of the form Eq. (1) are obtained. Then by solving such system of equations we could find the logarithms of elements of B. At final stage, $\log_\alpha \beta$ is computed as follows:

- Compute $\beta\alpha^i$ where i is a random integer and $0 \le i \le n-1$
- If we could express $\beta\alpha^i$ as $\beta\alpha^i = \prod_{k=1}^{|B|} p_k^{\mu_k}$ $\mu_k \ge 0$ $p_k \in B$ …. (2)

  By taken logarithms of both sides we get what we looking for

  $\log_\alpha \beta = \sum_{k=1}^{|B|} u_k \log_\alpha p_k - i$ $where\ \mu_k \ge 0\ and\ p_k \in B$

  , If $\beta\alpha^i$ couldn't be expressed as in Eq. (2), go to step 1 and try another i .

The index-calculus algorithm is adopted specially for multiplicative group of finite field $GF(p^n)$, where p is a prime.

1.3. The Integer Factorization Problem (IFP)

Many important cryptosystem schemes depend upon intractability of (IFP), Factoring integers is quite an old challenge of great interest, most modern method of factoring are variants of Fermat's method in which we write the number n to be factored as a difference of two squares $n = x^2$ - y^2 ,and so we conclude that $n = (x-y)(x+y)$. for a large number n , as it is the case in the public key cryptography (PKC) in which the key is an integer composed of around 300 digits, finding the two numbers x and y with their squares differ by n is not an easy task and it is the cornerstone of the security of the PKC, Kraitchik came up with an interesting enhancement, instead of trying to find integers x and y such that $x^2 - y^2 = n$ it might be suffice to find x and y with $x^2 \equiv y^2 \; mod\; n$ and it is this enhancement that is at the basis of most modern factoring algorithms . Many sieving algorithms developed to improve finding these numbers starting from the linear sieve up to the algebraic sieve. A sieve algorithm searches a lot of numbers satisfying a certain property. Then it makes some tests systematically on all these numbers, and at the end keeps the ones that have passed all the tests successfully, so sieving is really the act of filtering. The general number field sieve (GNFS) considered being the state of the art for the factoring algorithms, the key idea is to use smooth numbers in a number rings deferent from Z. For the RSA public key cryptosystem which is depend upon factoring, if we could factor the integer n which is part of the public key (the public key used to be equal to (n, e)) to two prime numbers p and q such that $n = p * q$,then, we may recover the private key (d) from the relation $d = e^{-1} mod\left((p-1)*(q-1)\right)$ since we choose from the beginning the private key (d) to satisfy the relation $e\, d \equiv 1 \; mod\, (p-1)*(q-1)$.

2. Symmetric group based cryptosystem

The symmetric group considered to be one of the most widely used groups in modern algebra, Doliskani, Malekian and Zakeralhosseini suggest using a symmetric group in a cryptosystem, to define the symmetric group let Gn be the set of integers such that $\mathrm{Gn} = \{1,2, \dots\dots, \mathrm{n}\}$ the symmetric group Sn is the set of all bijections (one to one and onto functions) from Gn to its self i.e. $\mathrm{Gn} \rightarrow \mathrm{Gn}$ and the group operation is the ordinary functions composition, as many other groups there is no distinct mathematical solution for the (DLP) for the symmetric group, i.e. ,there is no distinct mathematical solution for the equation $\alpha^x = \beta$, $\alpha, \beta \in \mathrm{Sn}$ $, x \in Z \ge 0$. Let H_δ = { δ^i ,$i = 1,2 \dots\dots, |\delta|$ } be the cyclic subgroup of Sn generated by δ which will be used in the cryptosystem. Doliskani, Malekian and Zakeralhosseini discussed and examined different (DLP) for the symmetric group Sn (practically the subgroup H_δ) which we introduce in the last section. Since $|\mathrm{H}_\delta|$ can be very large for large values of n, Therefore general purpose algorithms such as the brute-force search, Pollard's r-method and the baby-step giant-step algorithm with execution times of $O(|H_\delta|)$, $O(\sqrt{|H_\delta|})$ and $O(\sqrt{|H_\delta|}/r)$ (r is the number of the used processors) respectively, are inefficient to solve the (DLP). The Pohlig-Hellman algorithm uses the smoothness of the order of δ, we can generate (construct) δ in a way such that $|\delta|$ resistant to attacks by Pohlig-Hellman logarithm (1).For the index-calculus method which considered to be the most efficient algorithm known to the date for solving the discrete logarithm over finite field there is no algorithm for selecting appropriate subset of the symmetric group Sn for the factor base and it is likely to be difficult to develop such an algorithm and this made the proposed cryptosystem hard to attack by this algorithm.

2.1. Representing a symmetric group elements by integers

To represent an element of the symmetric groups by an integer and vice versa a bijection between integers and elements of symmetric groups (each elements in the symmetric group is actually a permutation) is introduced which enables to represent the elements of symmetric group by integers depending upon a mixed radix number system which is called factoradic or factorial representation by converting a number less than n! to factorial representation, one obtains a sequence of n digits that can be converted to a permutation of n in a straightforward way the i-th digit from the right has base i, which means that the digit must be strictly less than i, and that its value to be multiplied by (i − 1)! (Its place value), so for instance the factoradic number 341010 Can be converted to a decimal number as follows:

$$3*5!\ +4*4!+1*3!+0*2!+1*1!+0*0!=463$$

There is a natural mapping between the integers $0,\ldots,n!\ -\ 1$ (or equivalently the numbers with n digits in factorial representation) and permutations of n elements in lexicographical order, when the integers are expressed in factoradic form. This mapping has been termed the Lehmer code (or inversion table). For example,$with\ n\ =\ 3$, such a mapping is:

Decimal	factoradic	permutation
010	000!	(0, 1, 2)
110	010!	(0, 2, 1)
210	100!	(1, 0, 2)
310	110!	(1, 2, 0)
410	200!	(2, 0, 1)
510	210!	(2, 1, 0)

By writing down the factoradic number of the permutation we can use ordinary arithmetic to convert that to any desired base. (12).

Our suggestion in this paper is to convert the symmetric group element which we get during cryptosystem session to the factoradic analog integer and then to a decimal integer and use that integer as a private a key used by both parties to encrypt and decrypt following El-Gamal scheme.

2.2. Example explaining Doliskani,Ehsan and Zakerolhosseini proposal via our proposal

We will introduce an example to explain the steps followed in the cryptosystem proposed by (Doliskani,Ehsan and Zakerolhosseini) and secondly we will present the same example with our modification, essentially the cryptosystem is a variant of Diffie-Hellman scheme was introduced by T. El-Gamal The algorithm performs as follows :

User "A" selects a generator g in the selected group (the symmetric group in our case) and an integer *a*. he then publishes (g, g^a) as the public-key and keeps *a* secret. User "B", who requires to send a message m to "A", selects an integer *k* randomly, computes *m.* $((g^a)^k)$ =*m.*g^{ak} , and sends the pair (g^k, *m.*g^{ak}) to "A". User "A" who knows *a*, recovers *m* by computing $m.g^{ak}.(g^k)^{-a}$=*m.*$g^{ak}.g^{-ak}$=*m.*

2.2.1 Example explaining Doliskani,Ehsan and Zakerolhosseini proposal:

In the following example we introduce the cryptosystem proposed by Doliskani, Ehsan and Zakerolhosseini which include three main steps , each steps consists of sub steps as follow:

Step (I): Choosing the keys by user "A".

User "A" chooses his public and privet keys as follow:

1. Selects n=100 for the symmetric group Sn, and generates g as follows:

$g = (0,\cdots,22)(23,\cdots,41)(42,\cdots,58)(59,\cdots,71)(72,\cdots,82)(83,\cdots,89)(90,\cdots,94)(95,96,97)(98,99).$

.

Thus the order of the cyclic subgroup H generated by g will be $|Hg|$=223092870

2. Publishes (g ,g^{546584}) as the public key while keeping a=546584 as the private key. Computation of g^{546584} is very easy, and can be performed using an algorithm such as Right to Left Exponentiation. User "A" has g^{546584}= 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 0, 1, 2, 3, 4, 5, 6,7, 8, 9, 10, 11, 34, 35, 36, 37, 38, 39, 40, 41, 23, 24, 25, 26,27, 28, 29, 30, 31, 32, 33, 42, 43, 44, 45, 46, 47, 48, 49, 50,
51, 52, 53, 54, 55, 56, 57, 58, 71, 59, 60, 61, 62, 63, 64, 65,66, 67, 68, 69, 70, 77, 78, 79, 80, 81, 82, 72, 73, 74, 75, 76,86, 87, 88, 89, 83, 84, 85, 94, 90, 91, 92, 93, 97, 95, 96, 98,
99.

Step (II): Sending the message from user "B" to "A".

When user "B" requires sending a message to "A" for example the message m= "The quick brown fox jumps over the lazy dog" to "A" he do the following sub steps:

1. Interpreted the message m as an integer m =11815744420664747200359014215611078 24987407741879290620375891615821186633473930719017441769795978975216 7.

2. Transform "m" to the factoradic form: m=13,63, 28, 32, 53, 57, 33, 2, 61, 18, 27, 5, 21, 9, 57, 23, 4, 13,50, 37, 23, 30, 25, 21, 34, 19, 12, 33, 37, 32, 28, 20, 26, 22,23, 31, 20, 28, 24, 29, 18, 26, 16, 13, 10, 0, 13, 16, 22, 12,21, 15, 2, 7, 13, 16, 5, 2, 4, 2, 3, 10, 5, 8, 2, 2, 4, 0, 1, 0, 1;.

3. Transform m to a permutation to get: m= 6, 11, 58, 1, 67, 17, 36, 43, 8, 35, 70, 3, 14, 55, 46, 60, 44, 49, 7, 64, 15, 48, 45, 38, 42, 47, 72, 10, 54, 16, 39, 62,29, 24, 41, 40, 31, 51, 22, 26, 20, 69, 68, 52, 65, 12, 19, 34,59, 25, 30, 56, 37, 50, 71, 4, 23, 66, 9, 21, 5, 27, 18, 61, 2,33, 57, 53, 32, 28, 63, 13, 73, 74, 75, 76, 77, 78, 79, 80, 81,82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97,98, 99, 0.

4. selects k=87493 and computes $(g^{546584})^{k}$= $(g^{546584})^{87493}$ =12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 0, 1, 2,3, 4, 5, 6, 7, 8, 9, 10, 11, 39, 40, 41, 23, 24, 25, 26, 27, 28,29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 42, 43, 44, 45, 46, 47,48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 69, 70, 71, 59, 60,61, 62, 63, 64, 65, 66, 67, 68, 78, 79, 80, 81, 82, 72, 73, 74,75, 76, 77, 83, 84, 85, 86, 87, 88, 89, 92, 93, 94, 90, 91, 97,95, 96, 98, 99;

5.compute $m\,.(g^{546584})^{k}$ = $(g^{546584})^{87493}$ = 18, 0, 58, 13, 64, 6, 33,43, 20, 32, 67, 15, 3, 55, 46, 70, 44, 49, 19, 61, 4, 48, 45,35, 42, 47, 78, 22, 54, 5, 36, 59, 26, 40, 38, 37, 28, 51, 11,23, 9, 66, 65, 52, 62, 1, 8, 31, 69, 41, 27, 56, 34, 50, 68, 16,39, 63, 21, 10, 17, 24, 7, 71, 14, 30, 57, 53, 29, 25, 60, 2,79, 80, 81, 82, 72, 73, 74, 75, 76, 77, 83, 84, 85, 86, 87, 88,89, 92, 93, 94, 90, 91, 97, 95, 96, 98, 99, 12; and g^{k}= g^{87493} = 1, 2,3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,21, 22, 0, 40, 41, 23,24, 25, 26, 27, 28, 29, 30, 31, 32, 33,34, 35, 36, 37, 38, 39, 53, 54, 55, 56, 57, 58,42, 43, 44, 45,46, 47, 48, 49, 50, 51, 52, 62, 63, 64, 65, 66, 67, 68, 69, 70,71, 59, 60, 61, 82, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 83,84, 85, 86, 87, 88, 89, 93, 94, 90,91, 92, 96, 97, 95, 99, 98.

Then "B" sends the pair
($m.(g^{546584})^{k}$, g^{k}) to "A".

Step (III): Decryption the message by user "A".
User "A" decrypts the message as follows:

1. compute $(g^{k})^{-a}$ = $(g^{k})^{-546584}$ = 11,12,13,14,15,16,17,18,19,20,21,22, 0,1, 2,3,4,5,6,7,8,9,10,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,23,24,25,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,62,63,64,65,66,67,68,69,70,71,59,60,61,77,78,79,80,81,82,72,73,74,75,76,83,84,85,86,87,88,89,93,94,90,91,92,96,97,95,98,99.

2. And finely getting the plain text m by computing $m.(g^{546584})^{k}$.$(g^{k})^{-546584}$ =6,11,58,1,67,17,36,43,8,35,70,3,14,55,46,60,44,49,7,64,15,48,45,38,42,47,72,10,54,16,39,62,29,24,41,40,31,51,22,26,20,69,68,52,65,12,19,34,59,25,30,56,37,50,71,4,23,66,9,21,5,27,18,61,2,33,57,53,32,28,63,13,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,0= m

2.2.2 Our modification through the same example

Our suggestion to modify Doliskani, Ehsan and Zakerolhosseini proposal through the same example as follow:

Step (I): Choosing the keys by user "A".

This step done as in Doliskani, Ehsan and Zakerolhosseini proposal.

Step (II): Sending the message from user "B" to "A".

Through this step we suggest to modify Doliskani, Ehsan and Zakerolhosseini proposal as follow:

1. Interpreted the message m as an integer m =1181574442066474720035901421561107824987407741879290620375891615821186633473930719017441769795978975216 7, we suggest to keep m as an integer and not to convert it to a factoradic and after to a permutation as they did in sub step 2&3 of step (II).

2. selects an integer k=87493 and computes $(g^{546584})^k$= $(g^{546584})^{87493}$ =12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 0, 1, 2,3, 4, 5, 6, 7, 8, 9, 10, 11, 39, 40, 41, 23, 24, 25, 26, 27, 28,29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 42, 43, 44, 45, 46, 47,48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 69, 70, 71, 59, 60,61, 62, 63, 64, 65, 66, 67, 68, 78, 79, 80, 81, 82, 72, 73, 74,75, 76, 77, 83, 84, 85, 86, 87, 88, 89, 92, 93, 94, 90, 91, 97,95, 96, 98, 99;

3. In this sub step we modify their proposal through converting $(g^{a})^k$ to an integer in two steps, firstly to a factoradic number and secondly to a decimal number (t) .The decimal number $t \leq 100!-1$ = 9332621544394415268169923885626670049071596826438162146859296389521759999322991560894146397615651828625369792082722375825118521091686399999999999999999999999 99. And we use t to encrypt the message m, so the number of digits to be transmitted in each session from user B to user A will be decreased from 289 digits (number of digits needed to represent$(g^k)^a$) to 158 digits (maximum number of digits needed to represent $t$in our example),and this is the main benefit which we get from our modification , Let t= 93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286253697920827223758251185210916863999999999999999999999999 0.

4. User "B" compute g^k= g^{87493} = 1, 2,3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,21, 22, 0, 40, 41, 23,24, 25, 26, 27, 28, 29, 30, 31, 32, 33,34, 35, 36, 37, 38, 39, 53, 54, 55, 56, 57, 58,42, 43, 44, 45,46, 47, 48, 49, 50, 51, 52, 62, 63, 64, 65, 66, 67, 68, 69, 70,71, 59, 60, 61, 82, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 83,84, 85, 86, 87, 88, 89, 93, 94, 90,91, 92, 96, 97, 95, 99, 98.To encrypt the message m he compute $m + t$ = 933262154439441526816992388562667004907159682643816214685929638952175999932300337663856706236285218763958540316097224990253731399789015891615821186633473929 7, Then he sends the pair ($m + t$, g^k) to "A".

Step (III): Decryption the message by user "A". User "A" decrypts the message as follows:

1. compute $(g^k)^a$ = $(g^{87493})^{546584}$ = 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 0, 1, 2,3, 4, 5, 6, 7, 8, 9, 10, 11, 39, 40, 41, 23, 24, 25, 26, 27, 28,29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 42, 43, 44, 45, 46, 47,48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 69, 70, 71, 59, 60,61, 62, 63, 64, 65, 66, 67, 68, 78, 79, 80, 81, 82, 72, 73, 74,75, 76, 77, 83, 84, 85, 86, 87, 88, 89, 92, 93, 94, 90, 91, 97,95, 96, 98, 99; converting $(g^k)^a$to an integer by converting to a factoradic and after to decimal number so user A get the same number t which user "B" get ,t= 93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286253697920827223758251185210916863999999999999999999999999 0 .And finely user "A"get the message m by computing $(m+t)-t=$ 118157444206647472003590142156110782498740774187929062037589161582118663347393071901744176979597897521 67 = m.

3. Conclusions

The relative large memory and bandwidth requirements for storing and transmitting permutations is a negative point of the cryptosystem proposed by (Doliskani,Ehsan and Zakerolhosseini), by our suggestion in this paper we overcome this negative point by converting the symmetric group element which we use for encryption and decryption to an integer getting use from the bijection between a permutation (group element) and a factoradic number which can be converted to a decimal integer which posses less digits then the permutation element, as we show in the above example.

References

1. Doliskani, J.N.; Malekian, E. and Zakerolhosseini, A., A Cryptosystem Based on the Symmetric Group Sn, *IJCSNS International Journal of Computer Science and Network Security*, 8(2): **2008**, 226-234.
2. Diffie, W. and Hellman M., **1976** New directions in cryptography, *IEEE Transactions on Information Theory*, 22, 644-654.
3. El-Gamal, T., (**1985**) A public key cryptosystem and a signature scheme

based on discrete logarithms, *IEEE Transactions on Information Theory,* (31): 469- 472.

4. Koblitz, N., (**1987**) Elliptic curve cryptosystems, Mathematics of Computation, (48): 203-209.
5. Miller V., (**1986**) Uses of elliptic curves in cryptography, Advances in Cryptology- CRYPTO '85, *Lecture Notes in Computer Science*, Springer-Verlag, (218): 417-426.
6. Rivest, R. L., Shamir, A. and Adleman L., (**1978**) A method for obtaining digital signatures and public key cryptosystems, *Communication of the ACM*, (21), 120-126.
7. Shanks. D., (**1969**) Class number, a theory of factorization and genera. *In Proceedings of Symposia in Pure Mathematics*, (20): 415-440.
8. HELLMAN, M.E., (**1980**) A cryptanalytic time-memory tradeoff, *IEEE Transactions on information Theory*, (26): 401-406.
9. Cormen, T. H.; Leiserson C. E.; Rivest R. L. and Stein C., (**2001**) *Introduction to Algorithms.* MIT Press, second edition, 325.
10. POLLARD, J.M., (**1978**) Monte Carlo methods for index computation (mod p), *Mathematics of Computation*, 32: 918-924.
11. POHLIG, S.C. AND HELLMAN, M.E. , (**1978**) An improved algorithm for computing logarithms over GF(p) and its cryptographic significance, *IEEE Transactions on Information Theory*, (24): 106-110.
12. Lehmer, D. H., (**1960**) Teaching combinatorial tricks to a computer, *Proc. Sympos. Appl. Math. Combinatorial Analysis*, (10) Amer. Math. Soc., Providence, R. I., :179-193.