Journal of Advances in Mathematics and Computer Science

35(1): 23-38, 2020; Article no.JAMCS.50040

ISSN: 2456-9968

(Past name: British Journal of Mathematics & Computer Science, Past ISSN: 2231-0851)

Use of Signed Permutations in Cryptography

Iharantsoa Vero Raharinirina[1*]

[1] *Department of Mathematics and Computer Science, Faculty of Sciences, University of Antananarivo, BP 906 Antananarivo 101, Madagascar.*

Author's contribution

The sole author designed, analyzed, interpreted and prepared the manuscript.

Article Information

DOI: 10.9734/JAMCS/2020/v35i130237

Editor(s):

(1) Raducanu Razvan, Assistant Professor, Department of Applied Mathematics, Al. I. Cuza University, Romania.

Reviewers:

(1) Francisco Bulnes, Tecnolgico de Estudios Superiores de Chalco, Mexico.

(2) G. A. Sathish Kumar, Sri Venkateswara College of Engineering, India.

Complete Peer review History: http://www.sdiarticle4.com/review-history/50040

Received: 10 November 2019

Accepted: 16 January 2020

Published: 13 February 2020

Original Research Article

Abstract

In this paper we consider cryptographic applications of the arithmetic on the hyperoctahedral group. On an appropriate subgroup of the latter, we particularly propose to construct public key cryptosystems based on the discrete logarithm. The fact that the group of signed permutations has rich properties provides fast and easy implementation and makes these systems resistant to attacks like the Pohlig-Hellman algorithm. The only negative point is that storing and transmitting permutations need large memory. Using together the hyperoctahedral enumeration system and what is called subexceedant functions, we define a one-to-one correspondence between natural numbers and signed permutations with which we label the message units.

Keywords: Signed permutation; public key system; discrete logarithm problem; hyperoctahedral enumeration system; subexceedant function.

2010 Mathematics Subject Classification: 11T71, 68R15, 94A60, 97F60.

**Corresponding author: E-mail: vero.raharinirina@gmail.com, ihvero@yahoo.fr;*

1 Introduction

1.1 Basic notion

Cryptography: Cryptography has, as its etymology, kryptos from the Greek, meaning hidden, and graphein, meaning to write. By definition, cryptography is the study of methods for sending messages in secret (namely, in enciphered or disguised form) so that only the intended recipient can remove the disguise and read the message (or decipher it). The original message is called the plaintext, and the disguised message is called the ciphertext.

Messages and transformations: The plaintext and ciphertext are written in some alphabet (usually, but not always, they are written in the same alphabet) consisting of a set of a certain number of letters. The term "letter" (or "character") can refer not only to the familiar A-Z, but also to numerals, blanks, punctuation marks, or any other symbols that we choose to use when sending messages.

The plaintext and ciphertext are broken up into message units. A message unit might be a single letter, a pair of letters (digraph), a triple of letters (trigraph), or a block of $n \in \mathbb{N}$ letters.

The process of transforming plaintext into ciphertext is called encryption or enciphering. The reverse process of turning ciphertext into plaintext, which is accomplished by the recipient who has the knowledge to remove the disguise, is called decryption or deciphering.

Cryptosystem: A cryptosystem is composed of a set consisting of enciphering transformations and the corresponding set of deciphering transformations.

The first step in inventing a cryptosystem is to "label" all possible plaintext message units and all possible ciphertext message units by means of mathematical objects from which functions can be easily constructed.

Purpose of encryption: The purpose of encryption is to change data in such a way that only an authorized recipient is able to reconstruct the plaintext. This allows us to transmit data without worrying about it getting into unauthorized hands. Authorized recipients possess a piece of secret information (called the key) which allows them to decrypt the data while it remains hidden from everyone else.

1.2 Background

Before the 1970's, to cipher or decipher a message, two users of cryptographic system must safely exchange private key. New keys could be periodically distributed so as to keep the enemy guessing. In 1976, W. Diffie and M. Hellman [1] discovered an entirely different type of cryptosystem for which all of the necessary information to send an enciphered message is publicly available without enabling anyone to read the secret message. With this kind of system called public key, it is possible for two parties to initiate secret communications without exchanging any preliminary information or ever having had any prior contact.

Public key cryptography is typically used for generating secret keys for symmetric cryptographic sessions and for digital signatures. The security of public key cryptosystems is based on the hardness of some mathematical problems [2, Chapter 5]. As a public key cryptosystem consists of a private key (the deciphering key) that is kept secret and a public key (the enciphering key) which is accessible to the public, then the straightforward way to break the system is to draw the private key from the public key. Therefore, the required computation cost is equivalent to solving these difficult mathematical problems.

Until recently, public key cryptography in the industry was almost exclusively dominated by RSA [3]. Over the past few years, curve based cryptography [4, 5] has gained enormous popularity. Its security depends on the intractability of the discrete logarithm problem. The Diffie-Hellman [1] key-exchange protocol contained the basic original ideas for public-key cryptosystem based on this mathematical problem. In a paper [6], Taher ElGamal observes that by varying the Diffie-Hellman key agreement protocol slightly, one can obtain an other encryption based on the discrete logarithm.

In this paper, we present a new idea on how it is possible to perform ElGamal encryptions as well on the signed permutations group. The first objective of this subject will be to understand the hyperoctahedral group arithmetic for cryptographic purposes. The second objective would be to address theoretical implementation aspects of hyperoctahedral group cryptography.

2 Discrete Logarithm in Signed Permutations Group

Let us denote by:

- $[n]$ the set $\{1, \cdots, n\}$,
- $[\pm n]$ the set $\{-n, \cdots, -1, 1, \cdots, n\}$,
- $\mathcal{S}_n$ the symmetric group of degree n.

2.1 Signed permutations group

Definition 2.1. *A bijection $\pi : [\pm n] \longrightarrow [\pm n]$ satisfying $\pi(-i) = -\pi(i)$ for all $i \in [\pm n]$ is called "signed permutation".*

We can also write a signed permutation π in the form

$$\pi = \begin{pmatrix} 1 & 2 & \dots & n \\ \varepsilon_1\sigma_1 & \varepsilon_2\sigma_2 & \dots & \varepsilon_n\sigma_n \end{pmatrix} \text{ with } \sigma \in \mathcal{S}_n \text{ and } \varepsilon_i \in \{\pm 1\} .$$

Under the ordinary composition of mappings, all signed permutations of the elements of $[n]$ form a group $\mathcal{B}_n$ called hyperoctahedral group of rank n. We write $\pi^k = \underbrace{\pi \circ \cdots \circ \pi}_{k\text{-times}}$ for an integer k and $\pi \in \mathcal{B}_n$ and when we multiply permutations, the leftmost permutation acts first. For example,

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & -3 & 4 & 2 \end{pmatrix} \circ \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & -2 & 4 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & -4 & 1 & -2 \end{pmatrix} .$$

As stated in the work of Victor Reiner [7], a signed permutation decomposes uniquely into a product of commuting cycles just as permutations do.

Example 2.1. *The disjoint cycle form of $\theta = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 3 & 6 & -2 & 7 & -5 & -1 & 4 \end{pmatrix}$ is*

$$\theta = \underbrace{\begin{pmatrix} 1 & 3 & 2 & 6 \\ 3 & -2 & 6 & -1 \end{pmatrix}}_{\theta_1} \underbrace{\begin{pmatrix} 4 & 7 \\ 7 & 4 \end{pmatrix}}_{\theta_2} \underbrace{\begin{pmatrix} 5 \\ -5 \end{pmatrix}}_{\theta_3} .$$

Definition 2.2. *The order of a signed permutation π is the smallest positive integer m such that $\pi^m = \iota$ where ι denotes the identity permutation.*

The proof of the following theorem can be found in [8].

Theorem 2.2. *The order of a permutation written in disjoint cycle form is the least common multiple of the orders of the disjoint cycles.*

Theorem 2.3. *Let* $C = \begin{pmatrix} i_0 & i_1 & \dots & i_{\ell-2} & i_{\ell-1} \\ \varepsilon_0 i_1 & \varepsilon_1 i_2 & \dots & \varepsilon_{\ell-2} i_{\ell-1} & \varepsilon_{\ell-1} i_0 \end{pmatrix}$ *where* $\varepsilon_i \in \{\pm 1\}$, *an* ℓ*-cycle in the signed permutations group. The order of* C *is:*

$$|C| = \begin{cases} \ell & \text{if } \varepsilon_0\varepsilon_1 \dots \varepsilon_{\ell-1} = 1 \\ 2\ell & \text{if } \varepsilon_0\varepsilon_1 \dots \varepsilon_{\ell-1} = -1 \end{cases}. \tag{1}$$

Proof. In the symmetric group, the cycle $\sigma = \begin{pmatrix} i_0 & i_1 & \dots & i_{\ell-2} & i_{\ell-1} \\ i_1 & i_2 & \dots & i_{\ell-1} & i_0 \end{pmatrix}$ of length ℓ has order ℓ so the order of C is a multiple of ℓ. For $j = 0, \dots, \ell-1$ and $m \in \mathbb{N} \setminus \{0\}$, we have

$$C^m(i_j) = \varepsilon_{(j) \bmod \ell}\ \varepsilon_{(j+1) \bmod \ell}\ \dots\ \varepsilon_{(j+m-1) \bmod \ell}\ i_{(j+m) \bmod \ell}\ .$$

$$\begin{array}{lll} \text{For } m = 1, & C(i_j) = & \varepsilon_j\ i_{(j+1) \bmod \ell} \\ \text{For } m = 2, & C^2(i_j) = & \varepsilon_j\ C(i_{(j+1) \bmod \ell}) \\ & C^2(i_j) = & \varepsilon_j\ \varepsilon_{(j+1) \bmod \ell}\ i_{(j+2) \bmod \ell} \\ & \vdots & \\ \text{For } m = \ell, & C^\ell(i_j) = & \varepsilon_j\ \varepsilon_{(j+1) \bmod \ell}\ \dots\ \varepsilon_{(j+\ell-1) \bmod \ell}\ i_{(j+\ell) \bmod \ell}\ . \\ & C^\ell(i_j) = & \varepsilon_0\ \varepsilon_2\ \dots\ \varepsilon_{\ell-1}\ i_j \end{array}$$

We have $\varepsilon_j\ \varepsilon_{(j+1)\bmod \ell}\ \dots\ \varepsilon_{(j+\ell-1)\bmod \ell} = \varepsilon_0\ \varepsilon_2\ \dots\ \varepsilon_{\ell-1}$ because in the one hand, for all integers k_1, k_2 such that $0 \leqslant k_1 \leqslant k_2 \leqslant \ell-1-j$, we have $j \leqslant k_1 + j \leqslant k_2 + j \leqslant \ell - 1$ and in the other hand, for all integers k_1, k_2 such that $\ell - j \leqslant k_1 \leqslant k_2 \leqslant \ell - 1$, we have

$$\ell \leqslant k_1 + j \leqslant k_2 + j \leqslant \ell + j - 1 \leqslant \ell + \ell - 2 < 2\ell$$

$$0 \equiv \ell \mod \ell \leqslant (k_1 + j) \mod \ell \leqslant (k_2 + j) \mod \ell \leqslant (\ell - j - 1) \mod \ell \equiv j - 1.$$

If $\varepsilon_0\varepsilon_1 \dots \varepsilon_{\ell-1} = 1$ then $C^\ell(i_j) = i_j$ for all $j \in \{0, \dots, \ell-1\}$, that is C has order ℓ.

If $\varepsilon_0\varepsilon_1 \dots \varepsilon_{\ell-1} = -1$ we take $C^{2\ell}(i_j) = \varepsilon_0\ \varepsilon_2\ \dots\ \varepsilon_{\ell-1} C^\ell(i_j) = (\varepsilon_0\ \varepsilon_2\ \dots\ \varepsilon_{\ell-1})^2\ i_j\ = i_j$ for $j \in \{0, \dots, \ell-1\}$, so C has order 2ℓ. ☐

2.2 Calculating the discrete logarithm

Definition 2.3. *The discrete logarithm problem in the finite group* G *to the base* $g \in G$ *is the problem : given* $y \in G$, *of finding an integer* x *such that* $g^x = y$, *provided that such an integer exists (in other words, provided that* y *is in the subgroup generated by* g*).*

If we really want our random element y of G to have a discrete logarithm, g must be a generator of G.

Example 2.4. *Let* $G = \left(\mathbb{Z}/19\mathbb{Z}\right)^*$ *be the multiplicative group of integers modulo* 19. *The successive powers of* 2 *reduced* mod 19 *are :* $2, 4, 8, 16, 13, 7, 14, 9, 18, 17, 15, 11, 3, 6, 12, 5, 10, 1$. *Let* g *be the generator* 2, *then the discrete logarithm of* 9 *to the base* 2 *is* 8.

The known algorithms for the discrete logarithm problem (DLP) can be categorized as follows:

1. algorithms which work in arbitrary groups, e.g., exhaustive search [9], the baby-step giant-step algorithm [10], Pollard's rho algorithm [11],
2. algorithms which work in arbitrary groups but are especially efficient if the order of the group has only small prime factors, e.g., Pohlig-Hellman algorithm [12], and
3. the index-calculus algorithms [11, 13] which are efficient only in certain groups.

Many improvements are made for solving the discrete logarithm problem [14, 15].

The index-calculus algorithm is the most powerful method known for computing discrete logarithms. The technique employed does not apply to all groups, but when it does, it often gives a subexponential-time algorithm. A description of this algorithm in the general setting of a cyclic group G follows.

The index-calculus algorithm requires the selection of a relatively small subset S of elements of G, called the factor base, in such a way that a significant fraction of elements of G can be efficiently expressed as products of elements from S. The described algorithm proceeds to precompute a database containing the logarithms of all the elements in S, and then reuses this database each time the logarithm of a particular group element is required.

The description of this algorithm is incomplete for two reasons. Firstly, a technique for selecting the factor base S is not specified. Secondly, a method for efficiently generating relations of the form (2) and (3) is not specified. The factor base S must be a subset of G that is small (so that the system of equations to be solved in step 3 is not too large), but not too small (so that the expected number of trials to generate a relation (2) or (3) is not too large). Suitable factor bases and techniques for generating relations are known for some cyclic groups including the field $\mathbb{Z}_p^*$, $\mathbb{F}_{2^m}^*$ and, moreover, the multiplicative group $\mathbb{F}_q^*$ of a general finite field $\mathbb{F}_q$.

Index-calculus algorithm for discrete logarithms in cyclic groups

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

1. (Select a factor base S) Choose a subset $S = \{p_1, \dots, p_t\}$ of G such that a significant proportion of all elements in G can be efficiently expressed as a product of elements from S.
2. (Collect linear relations involving logarithms of elements in S)

   2.1 Select a random integer $k, 0 \leq k \leq n-1$ and compute α^k.

   2.2 Try to write α^k as a product of elements in S :

$$\alpha^k = \prod_{i=1}^{t} p_i^{c_i}, c_i \geq 0. \tag{2}$$

   If successful, take logarithms of both sides of equation (2) to obtain a linear relation

$$k = \sum_{i=1}^{t} c_i \log_\alpha p_i (\mod n). \tag{3}$$

   2.3 Repeat steps 2.1 and 2.2 until $t+c$ relations of the form (3) are obtained (c is a small positive integer, e.g. $c = 10$, such that the system of equations given by the $t+c$ relations has a unique solution with high probability).

3. (Find the logarithms of elements in S) Working modulo n, solve the linear system of $t+c$ equations (in t unknowns) of the form (3) collected in step 2 to obtain the values of $\log_\alpha p_i, 1 \leq i \leq t$.

4. (Compute y)

    4.1 Select a random integer $k, 0 \leq k \leq n-1$ and compute $\beta.\alpha^k$.

    4.2 Try to write $\beta.\alpha^k$ as a product of elements in S :

$$\beta.\alpha^k = \prod_{i=1}^{t} p_i^{d_i}, d_i \geq 0. \tag{4}$$

    If the attempt is unsuccessful then repeat step 4.1. Otherwise, taking logarithms of both sides of equation (4) yields $\log_\alpha \beta = (\sum_{i=1}^{t} d_i \log_\alpha p_i - k) \mod n$, thus, compute $y = (\sum_{i=1}^{t} d_i \log_\alpha p_i - k) \mod n$ and return(y).

Definition 2.4. *Let $\mathcal{B}_n$ be the hyperoctahedral group of rank n. The hyperoctahedral discrete logarithm problem to the base $\beta \in \mathcal{B}_n$ is the problem, given $\pi \in \mathcal{B}_n$, of finding an integer x such that $\pi = \beta^x$ if such x exists.*

Let $<\beta>= \{\beta^i \mid i = 1, \ldots, |\beta|\}$ be the cyclic subgroup of $\mathcal{B}_n$ generated by $\beta \in \mathcal{B}_n$.

If $\pi \notin <\beta>$ then the equation $\pi = \beta^x$ has no solution.

Therefore, discussion is restricted to $\pi \in <\beta>$ and $1 \leqslant x \leqslant |\beta|$.

The signed permutation β can be selected in such a way that the least common multiple of the lengths of the disjoint cycles of β be very large. For a large value of n, the order of $<\beta>$ can be very large so algorithms which work in arbitrary groups such as exhaustive search, the baby-step giant-step algorithm, Pollard's rho algorithm with running times of $\mathcal{O}(|\beta|)$, $\mathcal{O}(\sqrt{|\beta|})$ and $\mathcal{O}(\sqrt{|\beta|})$ respectively, are inefficient to solve the equation $\pi = \beta^x$.

If the order of the group $<\beta>$ is B-smooth for a reasonably small B, that is, if $|\beta|$ is a smooth integer, then discrete logarithms in $<\beta>$ can be efficiently computed by the method of Pohlig-Hellman.

Definition 2.5. *Let n be a positive real number. We say that n is smooth if all of the prime factors of n are small.*

Definition 2.6. *Let B be a positive real number. An integer is said to be B-smooth if it is not divisible by any prime greater than B.*

The following theorem allows to generate the base β so that the order of the subgroup $<\beta>$ of $\mathcal{B}_n$ has arbitrary smoothness.

Theorem 2.5. *Let p be a prime. For a large integer n, a cyclic subgroup G of $\mathcal{B}_n$ which its order is p-smooth and is not $(p-1)$-smooth can be constructed.*

Proof. Let n be a large integer and $p \leq n$ a prime. There exists a p-cycle $\gamma_p \in \mathcal{B}_n$. Let $\gamma_p, C_1, C_2, \ldots, C_k \in \mathcal{B}_n$ be disjoint cycles of length respectively $p, l_1, l_2, \ldots, l_k$ with

$$1 \leq l_i \leq p \text{ for } i = 1, \ldots, k \text{ and } \sum_{i=1}^{k} l_i \leq n - p\ .$$

Let us now consider the signed permutation $\pi = \gamma_p C_1 C_2 \ldots C_k \in \mathcal{B}_n$ of order

$$|\pi| = lcm(|\gamma_p|, |C_1|, |C_2|, \ldots, |C_k|)\ .$$

Let q be a prime such that $q|lmc(|\gamma_p|, |C_1|, |C_2|, \ldots, |C_k|)$. We show that $q \leq p$. Let us suppose that $q > p$. We obtain $l_i < p < q$ which implies $q \nmid |\pi|$ so $q \leq p$. We have just shown that every prime q dividing the order of π is smaller than p, that is, $|\pi|$ is p-smooth. The order $| < \pi > | = |\pi|$ of the cyclic subgroup $< \pi >$ of $\mathcal{B}_n$ generated by π is p-smooth but it is not $(p-1)$-smooth because $p \mid | < \pi > |$ and $p > p - 1$. $\square$

Theorem 2.5 provides high flexibility in selecting a subgroup of hyperoctahedral group $\mathcal{B}_n$ on which algorithms such as the Pohlig-Hellman algorithm is computationally inefficient.

The index-calculus algorithm requires the selection of the factor base which is a relatively small subset S of elements of $< \beta >$ such that a significant fraction of elements of $< \beta >$ can be efficiently expressed as products of elements from S. Then, collect linear relations involving logarithms of elements in S.

Assume $\beta = C_1 \ldots C_k$ where $C_1, \ldots, C_k$ are disjoint cycles. All the elements of $< \beta >$ can be expressed as products of $C_1, \ldots, C_k$.

Although the set $\{C_1, \ldots, C_k\}$ is small, it is not a subset of $< \beta >= \{\beta^i \mid i = 1, \ldots, |\beta|\}$ in general because there may exists an $i \in [k]$ such that $C_i \notin < \beta >$. For example, $\theta_1 \notin < \theta >$, $\theta_2 \notin < \theta >$ and $\theta_3 \notin < \theta >$ for the signed permutation θ in example 2.1. Moreover, even if the set $\{C_1, \ldots, C_k\}$ is a subset of $< \beta >$, one can only obtain a relation of the form (3) : let an integer $0 \leq d \leq |\beta| - 1$,

$$\begin{aligned} \beta^d &= \prod_{i=1}^{k} C_i^d \\ d &= \sum_{i=1}^{k} d \log_\beta C_i (\mod |\beta|) \\ 1 &= \sum_{i=1}^{k} \log_\beta C_i (\mod |\beta|). \end{aligned}$$

It is clear that this relation does not cover the logarithms of $C_1, \ldots, C_k$ so the set $\{C_1, \ldots, C_k\}$ can not be chosen as factor base whereas the factor base can be chosen as the set

$$S = \left\{ b_i = C_i^{(\left|\prod_{j \in [k] \setminus \{i\}} C_j\right| \mod |C_i|)}, i = 1, \ldots, k \right\}.$$

It is obvious that the logarithm of b_i to the base β is $\left|\prod_{j \in [k] \setminus \{i\}} C_j\right|$. The same result is obtained by collecting linear relations involving logarithms of elements in S as described in step 2 of the index-calculus algorithm for discrete logarithms in cyclic groups given previously, and pursuing step 3 of this algorithm.

An example which illustrates index-calculus algorithm in the cyclic subgroup of $\mathcal{B}_n$, generated by a signed permutation of $\mathcal{B}_n$, on a problem with artificially small parameters follows.

Example 2.6. *Let select $\alpha \in \mathcal{B}_{23}$ as follows:*

$$\alpha = \underbrace{\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\ -2 & 3 & -4 & 5 & -6 & 7 & -8 & 9 & -10 & 11 & -12 & 13 & -1 \end{pmatrix}}_{\alpha_1}$$

$$\underbrace{\begin{pmatrix} 14 & 15 & 16 \\ 15 & -16 & 14 \end{pmatrix}}_{\alpha_2} \underbrace{\begin{pmatrix} 17 & 18 & 19 & 20 & 21 & 22 & 23 \\ -18 & 19 & -20 & 21 & -22 & 23 & -17 \end{pmatrix}}_{\alpha_3}.$$

α_1, α_2 and α_3 are disjoint cycles.
From theorem 2.2 and theorem 2.3, we have $|\alpha| = lcm(26, 6, 7) = 546$. Consider

$$\tau = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\ 7 & 8 & 9 & 10 & 11 & 12 & 13 & 1 & -2 & 3 & -4 & 5 & -6 \end{pmatrix}$$
$$\begin{pmatrix} 14 & 15 & 16 \\ -14 & -15 & -16 \end{pmatrix}\begin{pmatrix} 17 & 18 & 19 & 20 & 21 & 22 & 23 \\ -18 & 19 & -20 & 21 & -22 & 23 & -17 \end{pmatrix}.$$

Then, $\log_\alpha \tau$ is computed as follows, using the index-calculus technique.

1. *The factor base is chosen to be the set $S = \{\alpha_1^{16}, \alpha_2^2, \alpha_3\}$.*
2. *The following four relations involving elements of the factor base are obtained (unsuccessful attempts are not shown):*

$$\begin{aligned} \alpha^2 &= \alpha_1^2\alpha_2^2\alpha_3^2 &&= (\alpha_1^{16})^5\alpha_2^2\alpha_3^2 && \text{because } 16(5) = 3(|\alpha_1|) + 2 \\ \alpha^{16} &= \alpha_1^{16}\alpha_2^{16}\alpha_3^{16} &&= \alpha_1^{16}(\alpha_2^2)^2\alpha_3^2 && \text{because } 16 \equiv 4 \mod |\alpha_2| \text{ and } 16 \equiv 2 \mod |\alpha_3| \\ \alpha^4 &= \alpha_1^4\alpha_2^4\alpha_3^4 &&= (\alpha_1^{16})^{10}(\alpha_2^2)^2\alpha_3^4 && \text{because } 16(10) = 6(|\alpha_1|) + 4 \\ \alpha^{26} &= \alpha_1^{26}\alpha_2^{26}\alpha_3^{26} &&= \alpha_2^2\alpha_3^5 && \text{because } |\alpha_1| = 26, 26 \equiv 2 \mod |\alpha_2| \text{ and } 26 \equiv 5 \mod |\alpha_3|. \end{aligned}$$

   These relations yield the following four equations involving the logarithms of elements in the factor base (for convenience of notation, let $b_1 = \alpha_1^{16}, b_2 = \alpha_2^2$ and $b_3 = \alpha_3$) :

$$\begin{aligned} 2 &\equiv 5\log_\alpha b_1 + \log_\alpha b_2 + 2\log_\alpha b_3 && (\mod 546) \\ 16 &\equiv \log_\alpha b_1 + 2\log_\alpha b_2 + 2\log_\alpha b_3 && (\mod 546) \\ 4 &\equiv 10\log_\alpha b_1 + 2\log_\alpha b_2 + 4\log_\alpha b_3 && (\mod 546) \\ 26 &\equiv \log_\alpha b_2 + 5\log_\alpha b_3 && (\mod 546). \end{aligned}$$

3. *Solving the linear system of four equations in three unknowns (the logarithms $x_i = \log_\alpha b_i$) yields the values $\log_\alpha b_1 = 42, \log_\alpha b_2 = 182$ and $\log_\alpha b_3 = 78$.*
4. *Suppose that the integer $k = 3$ is selected. Since*

$$\begin{aligned} \tau.\alpha^3 &= \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\ 10 & -11 & 12 & -13 & 1 & -2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{pmatrix} \\ &\quad \begin{pmatrix} 14 & 15 & 16 \\ 14 & 15 & 16 \end{pmatrix}\begin{pmatrix} 17 & 18 & 19 & 20 & 21 & 22 & 23 \\ 21 & 22 & 23 & 17 & -18 & 19 & -20 \end{pmatrix} \\ &= \alpha_1^{22}\alpha_3^4 \\ \tau.\alpha^3 &= (\alpha_1^{16})^3\alpha_3^4 \quad \text{because } 16(3) = |\alpha_1| + 22, \end{aligned}$$

   it follows that $\quad \log_\alpha \tau = (3\log_\alpha b_1 + 4\log_\alpha b_3 - 3) \mod 546 = 435$.

3 Method for Representing Integers by Signed Permutations

3.1 Converting natural number in hyperoctahedral system

Definition 3.1. *Hyperoctahedral number system is a system that expresses all natural number n of $\mathbb{N}$ in the form :*

$$n = \sum_{i=0}^{k(n)} d_i.B_i\ , \text{ where } k(n) \in \mathbb{N},\ d_i \in \{0, 1, 2, \cdots, 2i+1\} \text{ and } B_i = 2^i i!\ . \tag{5}$$

This definition is motivated by the fact that B_i is the cardinal of the hyperoctahedral group $\mathcal{B}_i$.

To denote the non-negative integer n of the equation (5) in the hyperoctahedral system, we use by convention the $(k+1)$-digits representation

$$d_k d_{k-1} \cdots d_2 d_1 d_0 \ .$$

Theorem 3.1. *Every positive integer has an unique representation in the hyperoctahedral system.*

Before giving the proof of this theorem, these are some properties of the hyperoctahedral system.

Lemma 1. *If $n = d_k \cdots d_1 d_0$ is a number in hyperoctahedral system, then*

$$0 \leqslant n \leqslant B_{k+1} - 1$$

Proof. Since $0 \leqslant d_i \leqslant 2i+1$, we have

$$0 \leqslant \sum_{i=0}^{k} d_i B_i \leqslant \sum_{i=0}^{k} (2i+1) B_i \ .$$

Recall that $B_i = 2^i i!$,

$$\begin{aligned} (2i+1)B_i &= (2(i+1-1)+1)B_i \\ &= (2(i+1)-1)B_i \\ &= 2(i+1)B_i - B_i \\ &= B_{i+1} - B_i \ . \end{aligned}$$

Therefore,

$$\sum_{i=0}^{k} (2i+1) B_i = \sum_{i=0}^{k} (B_{i+1} - B_i) = B_{k+1} - B_0 = B_{k+1} - 1 \ .$$

□

Lemma 2. *Let $n = d_k \cdots d_1 d_0$ be a number in hyperoctahedral system, then*

$$d_k B_k \leqslant n < (d_k + 1) B_k \ .$$

Proof. Let $m = d_{k-1} \cdots d_1 d_0$. From lemma 1, $0 \leqslant m < B_k$. We also have $n = d_k B_k + m$, hence

$$d_k B_k \leqslant n < B_k + d_k B_k = (1 + d_k) B_k \ .$$

□

Now, we can proceed to the demonstration of theorem 3.1 which states the one-to-oneness between non negative integers and hyperoctahedral base numbers.

Proof. Let $a_n \cdots a_1 a_0$ with $a_n \neq 0$ and $b_m \cdots b_1 b_0, b_m \neq 0$ be two representations of a positive integer N in hyperoctahedral system. First, $b_m \geq 1$ and $a_n \geq 1$ imply

$$B_m \leq b_m B_m \leq b_m \cdots b_1 b_0 \ \text{ and } B_n \leq a_n \cdots a_1 a_0.$$

The relation $n = m$ is immediate because if $n < m$ then

$$B_m \geq B_{n+1} > a_n \cdots a_1 a_0$$

by lemma 1 so

$$b_m \cdots b_1 b_0 > a_n \cdots a_1 a_0 \ .$$

Similarly, if $m < n$ then

$$a_n \cdots a_1 a_0 > b_m \cdots b_1 b_0 \ .$$

We get a contradiction. Consequently $n = m$ and $a_i = b_i$ for all $i \in \{0, 1, \ldots, n\}$ by induction and by the unicity of the expression $N = a_n B_n + r_n$ with $r_n = a_{n-1} \cdots a_1 a_0 < B_n$. □

To express a positive integer n in the hyperoctahedral system, one proceeds with the following manner. Start by dividing n by 2 and let d_0 be the rest r_0 of the expression

$$n = r_0 + (2)q_0 \ .$$

Divide q_0 by 4, and let d_1 be the rest r_1 of the expression

$$q_0 = r_1 + (4)q_1 \ .$$

Continue the procedure by dividing q_{i-1} by $2(i + 1)$ and taking $d_i := r_i$ of the expression

$$q_{i-1} = r_i + 2(i + 1)q_i$$

until $q_l = 0$ for some $l \in \mathbb{N}$. In this way, we obtain $n = d_l : d_{l-1} : \cdots : d_1 : d_0$ and we also have

$$n = d_0 + 2\left(d_1 + 4 \cdot (d_2 + 2(3) \cdot (d_3 + \cdots))\right).$$

Example 3.2 (Conversion is made by programming in PARI/GP). *In the hyperoctahedral system, the integer* $m = 1976620216402300889624487751 50$ *is represented by* $h_m = 41 : 40 : 33 : 24 : 33 : 6 : 33 : 24 : 1 : 13 : 14 : 7 : 16 : 15 : 13 : 4 : 4 : 11 : 1 : 7 : 3 : 3 : 0$.

Now let $n = d_{k-1} : d_{k-2} : \cdots : d_1 : d_0$ be a number in hyperoctahedral system. By definition 3.1, one way to convert n to the usual decimal system is to calculate

$$d_{k-1}2^{k-1}(k-1)! + \cdots + d_1.2 + d_0 \ .$$

In practice, one can use this algorithm:

```
Input :   An integer $d_{k-1} : d_{k-2} : \cdots : d_1 : d_0$ in hyperoctahedral system.
Output :              An integer $d$ in decimal system.

1. initiate the value of $d$ :    $d \leftarrow d_{k-1}$
2. for $i$ from $k-1$ to 1 do :   $d \leftarrow d.2.i + d_{i-1}$
3. return $d$
```

Example 3.3. *To convert the number* $7 : 0 : 2 : 3 : 1$ *to the decimal system, multiply 1, 3, 2, 0, 7 respectively by* B_0*,* B_1*,* B_2*,* B_3*,* B_4*, after that, add the results:*

$$7(384) + 0 + 2(8) + 3(2) + 1 = 2711 \ .$$

We obtain the same result with :

$$\begin{aligned} d_4 = &\quad 7\ , \\ 7(2)4 + d_3 = &\quad 56 + 0 = \quad 56\ , \\ 56(2)3 + d_2 = &\quad 336 + 2 = \quad 338\ , \\ 338(2)2 + d_1 = &\quad 1352 + 3 = \quad 1355\ , \\ 1355(2)1 + d_0 = &\quad 2710 + 1 = \quad 2711\ . \end{aligned}$$

3.2 A bijection between the subexceedant functions and permutations

Definition 3.2. *A subexceedant function* f *on* $[n]$ *is a map* $f : [n] \longrightarrow [n]$ *such that*

$$1 \leqslant f(i) \leqslant i \text{ for all } i \in [n] \ .$$

We will denote by $\mathcal{F}_n$ the set of all subexceedant functions on $[n]$, and we will represent a subexceedant function f over $[n]$ by the word $f(1)f(2)\cdots f(n)$.

Example 3.4. *These are the sets $\mathcal{F}_n$ for $n = 1, 2, 3$:*

$$\begin{aligned}\mathcal{F}_1 &= \{1\}\\ \mathcal{F}_2 &= \{11, 12\}\\ \mathcal{F}_3 &= \{111, 112, 113, 121, 122, 123\}.\end{aligned}$$

It is easy to verify that *card* $\mathcal{F}_n = n!$, since from each subexceedant f over $[n-1]$, one can obtain n distinct subexceedant functions over $[n]$ by adding any integer $i \in [n]$ at the end of the word representing f.

We will give a bijection between $\mathcal{S}_n$ and $\mathcal{F}_n$. Let be the map

$$\begin{array}{rcl}\phi : \mathcal{F}_n & \longrightarrow & \mathcal{S}_n\\ f & \longmapsto & \sigma_f = (1\, f(1))(2\, f(2))\cdots(n\, f(n))\end{array}.$$

Notice that there is an abuse of notation in the definition of σ_f . Indeed, if $f(i) = i$, then the cycle $(i\, f(i)) = (i)$ does not really denote a transposition but simply the identity permutation.

Lemma 3. *The map ϕ is a bijection from $\mathcal{F}_n$ onto $\mathcal{S}_n$.*

Proof. Since $\mathcal{S}_n$ and $\mathcal{F}_n$ both have cardinality $n!$, it suffices to prove that f is injective. Let f and g be two subexceedant functions on $[n]$. Assume that $\phi(f) = \phi(g)$ i.e. $\sigma_f = \sigma_g$. So we have:

$$(1\, f(1))(2\, f(2))\cdots(n\, f(n)) = (1\; g(1))(2\; g(2))\cdots(n\; g(n)). \tag{6}$$

Since $\sigma_f = \sigma_g$, then in particular $\sigma_f(n) = \sigma_g(n)$. By definition $\sigma_f(n) = f(n)$ and $\sigma_g(n) = g(n)$, so $f(n) = g(n)$. Let us multiply both members of equation (6) on the right by the permutation $(n\; f(n)) = (n\; g(n))$, we obtain:

$$(1\, f(1))(2\, f(2))\cdots(n-1 \quad f(n-1)) = (1\; g(1))(2\; g(2))\cdots(n-1 \quad g(n-1)).$$

Now, if we apply the same process to this equation, we obtain $f(n-1) = g(n-1)$. By iterating, we conclude that $f(i) = g(i)$ for all integers $i \in [n]$ and then $f = g$. ☐

Let σ be a permutation of the symmetric group $\mathcal{S}_n$ and f be the inverse image of σ by ϕ. Then f can be constructed as below:

1. Set $f(n) = \sigma(n)$.
2. Multiply σ on the right by $(n\; \sigma(n))$ (this operation consists in exchanging the image of n and the image of $\sigma^{-1}(n)$), we obtain a new permutation σ_1 having n as a fixed point. Thus σ_1 can be considered as a permutation of $\mathcal{S}_{n-1}$. Then set $f(n-1) = \sigma_1(n-1)$.
3. In order to obtain $f(n-2)$, apply now the same process to the permutation σ_1 by multiplying σ_1 by $(n-1 \quad \sigma_1(n-1))$. Iteration determines $f(i)$ for all integers i of $[n]$.

3.3 Mapping hyperoctahedral base numbers to signed permutations

Let $d_{n-1}\; d_{n-2}\; \cdots\; d_1\; d_0$ be a n-digits number in hyperoctahedral system. That is $d_i \in \{0, 1, 2, \cdots, 2i+1\}$ for $i = 0, \cdots, n-1$.

Writing $d_i = 2q_i + r_i$ where $r_i \in \{0, 1\}$ and $q_i \in \{0, \ldots, i\}$, gives

- the subexceedant function $f = f(1)\cdots f(n)$ with $f(i) = 1 + q_{i-1},\ i = 1, \ldots, n$

- and the sequence $(\varepsilon_1, \ldots, \varepsilon_n)$ with $\varepsilon_i = (-1)^{r_{i-1}}$ for $i = 1, \ldots, n$.

So to each integer $d_{n-1}\, d_{n-2}\, \cdots\, d_1\, d_0$, we associate the signed permutation

$$\begin{pmatrix} 1 & 2 & \ldots & n \\ \varepsilon_1\sigma_1 & \varepsilon_2\sigma_2 & \ldots & \varepsilon_n\sigma_n \end{pmatrix}$$

where $\sigma = \sigma_1 \cdots \sigma_n$ is the permutation associated to the subexceedant function f by the map

$$\begin{array}{rcl} \phi : \mathcal{F}_n & \longrightarrow & \mathcal{S}_n \\ f & \longmapsto & \sigma_f = (1 f(1))(2 f(2)) \cdots (n f(n)) \end{array}.$$

Example 3.5. *Let the number of* 23 *digits* $h_m = 41 : 40 : 33 : 24 : 33 : 6 : 33 : 24 : 1 : 13 : 14 : 7 : 16 : 15 : 13 : 4 : 4 : 11 : 1 : 7 : 3 : 3 : 0$ *in hyperoctahedral system. Writing each digit as* $d_i = 2q_i + r_i$ *gives*

- *the subexceedant function*

$$f = 1 \quad 2 \quad 2 \quad 4 \quad 1 \quad 6 \quad 3 \quad 3 \quad 7 \quad 8 \quad 9 \quad 4 \quad 8 \quad 7 \quad 1 \quad 13 \quad 17 \quad 4 \quad 17 \quad 13 \quad 17 \quad 21 \quad 21$$

with $f(i) = 1 + q_{i-1},\ i = 1, \ldots, 23$

- *and the sequence*

$$(\varepsilon_1, \ldots, \varepsilon_{23}) = (1, -1, -1, -1, -1, -1, 1, 1, -1, -1, 1, -1, 1, -1, -1, 1, -1, 1, 1, 1, -1, 1, -1)$$

with $\varepsilon_i = (-1)^{r_{i-1}}$ *for* $i = 1, \ldots, 23$.

The permutation associated to the subexceedant function f by the map ϕ *is the product of cycles* (1 1)(2 2)(3 2)(4 4)(5 1)(6 6)(7 3)(8 3)(9 7)(10 8)(11 9)(12 4)(13 8)(14 7)(15 1)(16 13)(17 17) (18 4)(19 17)(20 13)(21 17)(22 21)(23 21) *from left to right. We associate the signed permutation*

$$\mu = \left(\begin{array}{ccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\ 5 & -11 & -2 & -12 & -15 & -6 & 10 & 3 & -14 & -16 & 9 & -18 & 8 \end{array} \right.$$
$$\left. \begin{array}{cccccccccc} 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 \\ -7 & -1 & 20 & -19 & 4 & 22 & 13 & -17 & 23 & -21 \end{array} \right)$$

to the number $41 : 40 : 33 : 24 : 33 : 6 : 33 : 24 : 1 : 13 : 14 : 7 : 16 : 15 : 13 : 4 : 4 : 11 : 1 : 7 : 3 : 3 : 0$.

Now, let $\pi = \begin{pmatrix} 1 & 2 & \ldots & n \\ \varepsilon_1\sigma_1 & \varepsilon_2\sigma_2 & \ldots & \varepsilon_n\sigma_n \end{pmatrix}$ be a signed permutation. As ϕ is a bijection, from π we have

$$f = f(1) \cdots f(n) = \phi^{-1}(\sigma) \text{ and } r_{i-1} = \begin{cases} 0 & \text{if } \varepsilon_i = 1 \\ 1 & \text{if } \varepsilon_i = -1 \end{cases} \text{ for } i = 1, \ldots, n.$$

The digits $d_i = 2(f(i+1)-1)+r_i,\ i = 0, \ldots, n-1$ form the hyperoctahedral number $d_{n-1}\, d_{n-2}\, \cdots\, d_1\, d_0$. It is easy to verify that $d_i \in \{0, \ldots, 2i+1\}$. We have

$$\begin{array}{rl} 1 \leqslant f(i) \leqslant i & \text{for } i = 1, \ldots, n \\ 0 \leqslant f(i+1) \leqslant i+1 & \text{for } i = 0, \ldots, n-1 \\ 0 \leqslant 2(f(i+1)-1) \leqslant 2i & \text{for } i = 0, \ldots, n-1 \\ 0 \leqslant d_i \leqslant 2i+1 & \text{because } 0 \leqslant r_i \leqslant 1. \end{array}$$

4 Cryptography on Hyperoctahedral Group

In many ways, the group of signed permutations is analogous to the multiplicative group of a finite field. For this purpose, cryptosystems based on the latter can be translated to systems using the hyperoctahedral group. To label the message units, mathematical objects such as integers, points or vectors on some curves are often used in cryptography. This section addresses public key cryptosystems based on the group of signed permutations so one uses message units with signed permutations as equivalents (according to section 3).

4.1 Analog of the Diffie-Helman key exchange

The Diffie-Hellman key exchange [1] originally used the multiplicative group of a finite field. It can be adapted for signed permutations group as follow.

Suppose that two users Alice and Bob want to agree upon a secret key, a random element of $\mathcal{B}_n$, which they will use to encrypt their subsequent messages to one another. They first choose a large integer n for the hyperoctahedral group $\mathcal{B}_n$ and select a signed permutation $\beta \in \mathcal{B}_n$ to serve as their "base" and make them public. Alice selects a random integer $0 < a < B_n$, which she keeps secret, and computes $\beta^a \in \mathcal{B}_n$ which she makes public. Bob does the same. He selects a random integer $0 < b < B_n$, and transmits β^b to Alice over a public channel. Alice and Bob agree on the secret key β^{ab} by computing $(\beta^b)^a$ and $(\beta^a)^b$ respectively.

4.2 Generalized ElGamal public key encryption procedure in signed permutations group

- **Key generation.** Each entity creates a private key and a corresponding public key as follows :
  1. Select a large integer n for the hyperoctahedral group $\mathcal{B}_n$.
  2. Generate a signed permutation $\beta \in \mathcal{B}_n$ such that $|\beta|$ is divisible by a very large prime $p \leq n$.
  3. Select a random integer $0 < u < |\beta|$ and computes β^u.
  4. Publish (β, β^u) as public key and keep u as private key.
- **Encryption.** User "A" encrypts a message M for user "B" as follows :
  1. Obtain B's authentic public key (β, β^b)
  2. Represent the message M as an element μ of the signed permutations group $\mathcal{B}_n$.
  3. Select a random integer $0 < a < |\beta|$.
  4. Compute $\gamma_1 = \beta^a$ and $\gamma_2 = \mu.(\beta^b)^a$
  5. Send the pair of signed permutations $\gamma = (\gamma_1, \gamma_2)$ to user "B".
- **Decryption.** To recover plaintext m from γ, user "B" should do the following :
  1. Use B's private key b to compute $(\gamma_1^b)^{-1} = ((\beta^a)^b)^{-1}$.
  2. Recover μ by multiplying γ_2 on the right by γ_1^{-b}.
  3. Recover M by computing the integer representation of μ.

Example 4.1 (ElGamal encryption with artificially small parameters)**.** ***Key generation.*** *Entity "B" selects to use the hyperoctahedral group $\mathcal{B}_{23}$ and its subgroup generated by the signed permutation α of order 546 in example 2.6. "B" chooses the private key $b = 121$ and computes α^{121}. B's public key is (α, α^b) with*

$$\alpha^b = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\ -5 & -6 & -7 & -8 & -9 & -10 & -11 & -12 & -13 & -1 & 2 & -3 & 4 \end{pmatrix}$$
$$\begin{pmatrix} 14 & 15 & 16 \\ 15 & -16 & 14 \end{pmatrix}\begin{pmatrix} 17 & 18 & 19 & 20 & 21 & 22 & 23 \\ -19 & -20 & -21 & -22 & -23 & -17 & -18 \end{pmatrix}.$$

Encryption. *User "A" requires to send the message " ATTACK AT DAWN " interpreted as the integer m of example 3.2 to user "B". m is represented by the signed permutation μ of example 3.5. "A" selects the integer $a = 14$ and computes*

$$\alpha^a = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\ 2 & -3 & 4 & -5 & 6 & -7 & 8 & -9 & 10 & -11 & 12 & -13 & 1 \end{pmatrix}\begin{pmatrix} 14 & 15 & 16 \\ -16 & -14 & 15 \end{pmatrix}$$

and

$$\mu.(\alpha^b)^a = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\ 9 & 2 & -6 & -3 & 14 & -10 & 1 & 7 & 16 & -15 & 13 & -18 & 12 \end{pmatrix.}$$
$$\begin{matrix} 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 \\ -11 & -5 & 20 & -19 & 8 & 22 & -4 & -17 & 23 & -21 \end{matrix}\Bigg).$$

Then, "A" sends the pair of signed permutations $(\alpha^a, \mu.(\alpha^b)^a)$ to user "B".

Decryption. *To decrypt, user "B" computes*

$$((\alpha^a)^b)^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\ 10 & -11 & 12 & -13 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{pmatrix}\begin{pmatrix} 14 & 15 & 16 \\ -15 & 16 & -14 \end{pmatrix}$$

and recovers μ by multiplying $\mu.(\alpha^b)^a)$ on the right by $((\alpha^a)^b)^{-1}$. "B" recovers m by computing the integer representation of μ.

4.3 Efficiency and security

It is easy to represent any data stream with a finite length by an unique signed permutation thanks to the one to one correspondence between signed permutations and natural numbers. The group operation in the signed permutation group, which is the ordinary composition of mappings, is easy to apply. On the hyperoctahedral group $\mathcal{B}_n$, it can be performed in time $\mathcal{O}(n)$. Optimized method for exponentiation makes the proposed Diffie-Hellman key exchange and ElGamal cryptosystem easy to implement. However, as we must work in a very large hyperoctahedral group, the need of large memory from the point of view implementation requires improvements.

For the Diffie-Hellman's key exchange above, a third party knows only the elements β ,β^a and β^b of $\mathcal{B}_n$ which are public knowledge. Obtaining β^{ab} knowing only β^a and β^b is as hard as taking the discrete logarithm a from β and β^a (or b knowing β and β^b). Then an unauthorized third party must solve the discrete logarithm problem to the base $\beta \in \mathcal{B}_n$ to determine the private key.

Breaking the generalized ElGamal cryptosystem above, that is, recovering μ knowing only β, β^b, β^a and $\mu.(\beta^b)^a$, amounts to finding β^{ab} and then multiplying $\mu.(\beta^b)^a$ on the right by $(\beta^{ab})^{-1}$. As we have already seen previously, to determine β^{ab}, an unauthorized third party needs the solution of the discrete logarithm problem in the cyclic subgroup $< \beta >$ of $\mathcal{B}_n$. However the fact that the signed permutations group is non commutative strengthens the security of this generalized ElGamal cryptosystem.

5 Conclusions

The main idea for implementation of the hyperoctahedral group cryptography is to establish a bijection between signed permutations and natural numbers. Multiplication and inversion in signed permutations group are very fast. The only negative point of the implementation is the need of large memory for storing and transmitting permutations. Subgroups of signed permutations group on which the Pohlig-Hellman's algorithm is unsuccessful to solve the discrete logarithm problem can be selected with a very easy and low cost method. This fact makes the proposed Diffie-Hellman key exchange and ElGamal cryptosystem in security while index-calculus algorithm treats without difficulty the DLP on these subgroups.

Acknowledgement

The author is gratefully indebted to Randrianarivony Arthur for his careful reading of the manuscript and his active interest in the publication of this paper. This paper was prepared with the help of the personal computer that the Simons Foundation pole PREMA (past name: PRMAIS) offered to the author.

Competing Interests

Author has declared that no competing interests exist.

References

[1] Diffie W, Hellman ME. New directions in cryptography. IEEE Transactions on Information Theory. 1976;IT-22:644-654.

[2] Bernhard Esslinger. Learning and Experiencing Cryptography with CrypTool and SageMath. 12th Ed., CrypTool Project; 2018.

[3] Rivest RL, Shamir A, Adleman LM. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM. 1978;21(2):120-126.

[4] Galbraith SD, Gaudry P. Recent progress on the elliptic curve discrete logarithm problem. Designs, Codes and Cryptography. 2016;78.1:51-72.

[5] Christophe Petit, Michiel Kosters, Ange Messeng. Algebraic approaches for the elliptic curve discrete logarithm problem over prime fields. IACR International Workshop on Public Key Cryptography. Springer Berlin Heidelberg. 2016;3-18.

[6] ElGamal T. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory. 1985;IT-31:469-472.

[7] Reiner V. Signed permutation statistics and cycle type. European Journal of Combinatorics. 1993;14:569-579.

[8] Rotman JJ. Advanced Modern Algebra. 1st Edition, Prentice Hall, New Jersey; 2002.

[9] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to algorithms. MIT Press, Second Edition; 2001.

[10] Shanks D. Class number, a theory of factorization, and genera. In Proceedings of Symposia in Pure Mathematics. 1969;20:415-440.

[11] Pollard JM. Monte Carlo methods for index computation ($\bmod p$). Mathematics of Computation. 1978;32:918-924.

[12] Pohlig SC, Hellman ME. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. IEEE Transactions on Information Theory. 1978;24:106-110.

[13] Adleman LM. A sub-exponential algorithm for the discrete logarithm problem with applications to cryptography. IEEE Annual Symposium on Foundations of Computer Science. 1979;20:49-74.

[14] Gopalakrishna Hejmadi Gadiyar, Ramanathan Padma and Vellore. The discrete logarithm problem over prime fields: the safe prime case. the smart attack, non-canonical lifts and logarithmic derivatives. Czechoslovak Mathematical Journal. 2018;68(143):1115-1124.

[15] Jung Hee Cheon, Taechan Kim. A new approach to the discrete logarithm problem with auxiliary inputs. LMS J. Comput. Math. 2016;19(1):1-15.
