Extensions of the Diffie-Hellman Key Agreement Protocol Based on Exponential and Logarithmic Functions

Zbigniew Lipiński[1] and Jolanta Mizera-Pietraszko[2](✉)

[1] University of Opole, Opole, Poland
zlipinski@uni.opole.pl
[2] Military University of Land Forces, Wroclaw, Poland
jolanta.mizera-pietraszko@awl.edu.pl

Abstract. We propose a method of constructing cryptographic key exchange protocols of the Diffie-Hellman type based on the exponential and logarithmic functions over the multiplicative group of integers modulo n. The security of the proposed protocols is based on the computational difficulty of solving a set of congruence equations containing a discrete logarithm. For the multiplicative group of integers modulo n we define the non-commutative group of their automorphisms. On the defined group we construct non-commutative key exchange protocol similar to the Anshel-Anshel-Goldfeld key exchange scheme.

Keywords: Diffie-Hellman key agreement · Primitive roots of finite filed · Non-commutative cryptography

1 Introduction

The Diffie-Hellman key exchange is one of the most utilized key agreement protocols in the secure network communication. There are symmetric and asymmetric (public) key exchange variants of the protocol. In case of the symmetric Diffie-Hellman key exchange protocol the users A and B agree the modulus p being the prime number and the primitive root r from the multiplicative group $(\mathrm{Z}/(p))^*$ of integers modulo p. The user A selects the number a and the user B selects the number b from $(\mathrm{Z}/(p))^*$. Both communicating parties exchange the numbers $r^a \bmod p$ and $r^b \bmod p$. The common key is $k_{a,b} = (r^b)^a = (r^a)^b \bmod p$, [1,2]. Security of the protocol is based on computational difficulty in determining the value of discrete logarithm $\log_r(r^a) \bmod \varphi(p)$. The standard X9.42 defines the Diffie-Hellman public key exchange scheme in which the private-public keys is the pair $(a, g^a \bmod p)$, $a, g \in (\mathrm{Z}/(p))^*$, [2]. According to the standard, the modulo parameter should be a prime number of the form $p = jq + 1$, where q is a large prime and $j \geq 2$. The base g of the public key $g^a \bmod p$ is of the form $g = h^j \bmod p$, where h is any integer with $1 < h < p - 1$ such that $h^j \bmod p > 1$. The base g does not have to be a generator of the cyclic group

$(Z/(p))^*$. For $j = 2$ we obtain the so called 'safe prime' numbers $p = 2q + 1$, where q is prime. The set of primitive roots $S^{(r)}_{2q+1}$ can be obtained by removing from the group $(Z/(2q + 1))^*$ all elements raised to power two (the quadratic residues) and removing the element $2q$, which is the unique element such that $(2q)^2 = 1 \bmod (2q + 1)$.

For the symmetric Diffie-Hellman key exchange protocol the crucial problem is determination of the primitive root elements from the group $(Z/(p))^*$. Unfortunately, there is no general formula which allows to determine it. In the preliminary section we discus the problem of finding the primitive root elements in cyclic groups. In the Lemma we define the set of primitive roots in the group $(Z/(p))^*$ as the common part of the complement of the sets of p-residues, see the formula (2). This definition allows in an efficient way to construct algorithms for searching the primitive root elements in cyclic groups. In the same section we define four types of functions which permute the elements of the group $(Z/(p))^*$. In Sect. 4 we propose the method of constructing Diffie-Hellman like key agreement protocols based on defined exponential and monomial functions. The security of the discussed protocols is based on the computational difficulty of solving a set of congruence equations containing the discrete logarithm. The public key exchange variants of the proposed protocols can be obtained straightforward.

With each of the defined functions we associate the subgroup of symmetric group of $p-1$ elements. In Sect. 5 we use these groups to construct a symmetric key agreement protocol based on the idea of the Anshel-Anshel-Goldfeld (AAG) key agreement scheme [3,4].

2 Related Work

The idea of using the non-abelian groups in cryptography has its origin in the solutions of three famous problems in combinatorial group theory proposed by M. Dehn in 1911, [5]. These are the word problem, the conjugacy problem and the isomorphism problem for finitely presented groups, [6]. In [7] Wagner and Magyarik devised the first public-key protocol based on the unsolvability of the word problem for finitely presented groups. A non-deterministic public-key cryptosystem based on the conjugacy problem on braid group, similar to the Diffie-Hellman key exchange system, was proposed in [8]. The particular importance are the Anshel-Anshel-Goldfeld key agreement system and the public-key cryptosystem proposed in [3,4,9]. The authors used the braid groups where the word problem is easy to solve but the conjugacy problem is intractible. This is due to the fact that on braid groups the best known algorithm to solve the conjugacy problem requires at least exponential running time. Proposed in Sect. 5 the symmetric key agreement protocol is based on the idea of the Anshel-Anshel-Goldfeld (AAG) key agreement scheme.

In Sect. 4 we construct a familly of Diffie-Hellman like key agreement protocols based on exponential and monomial functions which permute the cyclic group $(Z/(p))^*$. This is one of many generalization of the Diffie-Hellman key exchange protocol. For example, in [10] it was proposed a generalization of the

Diffie-Hellman scheme called Algebraic generalization of Diffe-Hellman (AGDH). Its security is based on the hardness of a solution of the homomorphic image problem, which requires to compute the image of a given element under an unknown homomorphism of two algebras selected as a encryption platform. In [11] a matrix-based Diffie-Hellman key exchange protocol was proposed. The security of the proposed protocol is based on exploiting of a non-invertible public matrix in the key generating process. In [12] a polynomial time algorithm to solve the Diffie-Hellman conjugacy problem in braid groups was proposed. Security aspects of the algorithm was analysed and proved. In [13] was considered a Diffie-Hellman like key exchange procedure which security is based on the difficulty of computing discrete logarithms in a group matrices over noncommutative ring. In the algorithm the exponentiation is hidden by a matrix conjugation which ensures it security. In [14] a new computational problem called the twin Diffie-Hellman problem was proposed. The twin DH protocol allows to avoid the problem of an attack on the public keys exchanged in the standard Diffie-Hellman scheme. I. F. Blake and T. Garefalakis analyzed the complexity of the discrete logarithm problem, the Diffie-Hellman and the decision DH problem, [15]. The authors showed that if the decision DH problem is hard then computing the two most significant bits of the DH function is hard. In [16] the authors analyze complexity of the Group Computational Diffe-Hellman and Decisional Diffe-Hellman problems and their application in cryptographic protocols. The security of the group Diffe-Hellman key exchange is discussed, [17]. In [18] the authors apply the symbolic protocol analysis to cryptographic protocols which utilize the exponentiation calculations on cyclic groups. Using this analysis several security aspects of Diffie-Hellman like operations was proved.

3 Preliminaries

The multiplicative group $(\mathrm{Z}/(n))^*$ is the set of elements from the ring $\mathrm{Z}/(n)$ of integers modulo n coprime to n. The group $(\mathrm{Z}/(n))^*$ is cyclic if and only if n is of the type $n = 1, 2, 4, p^m, 2p^m$, where p is an odd prime number and $m \geq 1$, [19]. The generators of $(\mathrm{Z}/(n))^*$ are called primitive roots. The order of a primitive root $r \in (\mathrm{Z}/(n))^*$ is equal to the Euler totient function $\varphi(n)$, i.e., $\varphi(n)$ is the smallest number such that $r^{\varphi(n)} = 1 \bmod n$. By $S_n^{(r)}$ we denote the set of all primitive roots in $(\mathrm{Z}/(n))^*$. The cardinality of the set $S_n^{(r)}$ equals to $\varphi(\varphi(n))$. By $(\mathrm{Z}/\varphi(n))^*$ we denote the multiplicative group of integers coprime to $\varphi(n)$, i.e., $(\mathrm{Z}/\varphi(n))^* = \{k \in \mathrm{Z}/\varphi(n) : \gcd(k, \varphi(n)) = 1\}$. For any $i \in (\mathrm{Z}/(n))^*$ and $k \in (\mathrm{Z}/\varphi(n))^*$ the equation $i \cdot k = 0 \bmod \varphi(n)$ implies that $i = 0 \bmod \varphi(n)$. From this implication it follows that the set $\{i^k \bmod n : i \in (\mathrm{Z}/(n))^*\}$ is equal to $(\mathrm{Z}/(n))^*$ and the elements $r^k \bmod n$, for given primitive root r, generate the whole set

$$S_n^{(r)} = \{r^k \bmod n : k \in (\mathrm{Z}/\varphi(n))^*\}. \tag{1}$$

Let p be a prime number and $\varphi(p) = p_0^{a_0} p_1^{a_1} \cdots p_k^{a_k}$, where $p_0 = 2$. The element $g \in (\mathrm{Z}/(p))^*$ is a residue of degree p_i modulo p if there exists $a \in (\mathrm{Z}/(p))^*$ such that $a^{p_i} = g \bmod p$. By $((\mathrm{Z}/(p))^*)^{p_i}$ we denote the group of p_i-residues $\bmod p$ and by

$$(((\mathrm{Z}/(p))^*)^{p_i})^C = (\mathrm{Z}/(p))^* \setminus ((\mathrm{Z}/(p))^*)^{p_i}$$

the set of p_i-non-residues, i.e., the complement of $((\mathrm{Z}/(p))^*)^{p_i}$ in $(\mathrm{Z}/(p))^*$. The group $((\mathrm{Z}/(p))^*)^{p_i}$ is a cyclic of order $\varphi(p)/p_i$.

Lemma. For a prime number p and $\varphi(p) = p_0^{a_0} p_1^{a_1} \cdots p_k^{a_k}$ the set of primitive roots can be obtained from the formula

$$S_p^{(r)} = \bigcap_{i=0}^{k} (((\mathrm{Z}/(p))^*)^{p_i})^C. \tag{2}$$

Proof. Let $g = r^b, x = r^a \in (\mathrm{Z}/(p))^*$ and $x^{p_i} = g \bmod p$ for some root r. From $r^{a \cdot p_i} = r^b \bmod p$ it follows that $a \cdot p_i = b \bmod \varphi(p)$. If $\gcd(b, \varphi(p)) = 1$ then p_i does not divides b, which means that $b \in (\mathrm{Z}/\varphi(p))^*$ and $r^b \bmod p \in S_p^{(r)}$ $\diamond$.
The formula (2) states that for a primitive root $r \in (\mathrm{Z}/(p))^*$ the set of congruence equations $x^{p_i} = r \bmod p, i \in [0, k]$ is not solvable. By

$$G^p_{p_0 \cdots p_k} = \bigcap_{i=0}^{k} ((\mathrm{Z}/(p))^*)^{p_i} = ((\mathrm{Z}/(p))^*)^{p_0 p_1 \cdots p_k}$$

we denote the intersection of all p_i-residue groups $((\mathrm{Z}/(p))^*)^{p_i}$, $i \in [0, k]$. Because any root $r = r_0^e \bmod p$, where $r_0 \in S_p^{(r)}$ and $e \in (\mathrm{Z}/\varphi(p))^*$, then the group $G^p_{p_0 \cdots p_k}$ leaves the set of roots invariant and acts on $(\mathrm{Z}/\varphi(p))^*$ as translations

$$T_g(r_0^e) = g \cdot r_0^e = r_0^{e + \mathrm{ind}(g)} \bmod p,$$

where $g \in G^p_{p_0 \cdots p_k}$ and $\mathrm{ind}(g)$ is the index of g. The order of the group $|G_{p_0 p_1 \cdots p_k}| = \frac{\varphi(p)}{p_0 p_1 \cdots p_k}$.

One of the first algorithms which allow to find a primitive root was defined by Gauss. In the algorithm it is used the fact that for the element $g \in (\mathrm{Z}/(p))^*$ such that $g^{(p-1)/p_i} \neq 1$ the orders of $g^{(p-1)/p_i}$ and $g^{(p-1)/p_i^{a_i}}$ are equal to $\mathrm{Ord}_p(g^{(p-1)/p_i}) = p_i$ and $\mathrm{Ord}_p(g^{(p-1)/p_i^{a_i}}) = p_i^{a_i}$ respectively. If $p_i \neq p_j$, then by multiplying $g_i^{(p-1)/p_i^{a_i}}$ and $g_j^{(p-1)/p_j^{a_j}}$ one can obtain an element of order $\mathrm{Ord}_p(g_i^{(p-1)/p_i^{a_i}} g_j^{(p-1)/p_j^{a_j}}) = p_i^{a_i} p_j^{a_j}$.

Gauss's algorithm to find a primitive root.

1. Find the first p_i-non-residue, $i \in [0, k]$, i.e., the numbers g_i such that $g_i^{(p-1)/p_i} \neq 1 \bmod p$.
2. From the formula $g_i^{(p-1)/p_i^{a_i}} \bmod p$ determine the elements of order $p_i^{a_i}$.
3. The product $\prod_{i=0}^{k} g_i^{(p-1)/p_i^{a_i}} \bmod p$ is a primitive root.

As an example of application of the Gauss's algorithm we find a primitive root in $\mathrm{Z}/(p)^*$, where $p = 4441$. Because $p-1 = 2^3 \cdot 3 \cdot 5 \cdot 37$, for each prime $p_0 = 2, p_1 = 3,$

$p_2 = 5$, $p_3 = 37$ the first 2th non-residue is 7, the first 3th, 5th and 37th non-residue is 2. The element $r = 7^{4440/2^3} \cdot 2^{4440/3} \cdot 2^{4440/5} \cdot 2^{4440/37} = 2749 \bmod 4441$ is a primitive root.

E. Bach modified the Gauss's algorithm to the following form.

1. Find $B \geq 1$ such that $B \log B = C \log p$, $C = 30$.
2. Factor $p - 1 = p_0^{a_0} \dots p_s^{a_s} Q$, where $p_i < B$ and Q are free of primes $< B$.
3. For each $i = 0, \dots, s$ choose a prime $g_i \leq 2(\log p)^2$ such that $g_i^{(p-1)/p_i} \neq 1$.
4. For $g = \prod_{i=0}^{s} g_i^{(p-1)/p_i^{a_i}}$ construct the set
   $S = \{g \cdot q^{(p-1)/Q} \bmod p : q \text{ is prime and } q \leq 5\frac{(\log p)^4}{(\log\log p)^2}\}$.

In [20] E. Bach proved, assuming that the extended Riemann hypothesis is true, that the algorithm computes the set S of residues $\bmod p$ such that S contains a primitive root and the cardinality of the set $|S|$ is of order $O(\frac{(\log p)^4}{(\log\log p)^3})$. As an example of application of E. Bach algorithm we find a primitive root in $\mathrm{Z}/(p)^*$, where $p = 4441$. For the prime number $p = 4441$ take $C = 1$, $B = 5$, $p_0 = 2$, $p_1 = 3$ and write $p - 1 = 2^3 \cdot 3 \cdot Q$, where $Q = 5 \cdot 37$ and $g = 7^{4440/2^3} \cdot 2^{4440/3} = 2690 \bmod 4441$. For $q = 2$ the element $r = g \cdot q^{4440/Q} = 2690 \cdot 2^{4440/185} = 3355 \bmod 4441$ is a primitive root.

Table 1. The primitive roots in $\mathrm{Z}/(p)^*$, $p - 1 = 2^3 \cdot 3 \cdot 5 \cdot 37$.

	p_i-residue mod p	p_i-non-residue mod p	primitive root
$p_0 = 2$	1, 2, 3, ...	7, 11, 13, ...	$(2 \cdot 3) \cdot 13$, ...
$p_1 = 3$	1, 6, ...	2, 3, 4, 5, 7, 9, 10, 12, 13, ...	$6 \cdot 13$, ...
$p_2 = 5$	1, 6, ...	2, 3, 4, 5, 8, 9, 10, 11, 12, 13, ...	$6 \cdot 13$, ...
$p_3 = 37$	1, 13, ...	2, 3, ...	$13 \cdot (2 \cdot 3)$, ...

From the formula (2) and the fact that the product of two k-residues is k-residue and the product of k-residue and k-non-residue is k-non-residue one can find a non-prime primitive root following the rule. For each $i \in [0, k]$ find a pair (g_i, g_i') of elements $g_i \in ((\mathrm{Z}/(p))^*)^{p_i}$ and $g_i' \in (((\mathrm{Z}/(p))^*)^{p_i})^C$ such that $\forall_{i \in [1,k]}\ g_i \cdot g_i' = g_0 \cdot g_0' \bmod p$, then the common value is a primitive root. In the Table 1 it was shown how to find a primitive root using proposed rule for $p = 4441$. The best know unconditional estimate for the smallest primitive root is due to Burgess [21]. Based on Burgess's results in [22] Shparlinski constructed a deterministic algorithm which finds a primitive root in a finite field F_p in time $O(p^{\frac{1}{4}+\epsilon})$, for any $\epsilon > 0$. V. Shoup in [23], assuming the extended Riemann hypothesis proved that the least primitive root g_p is bounded by a polynomial in $\log p$

$$g_p = O(\omega^4(1 + \log \omega)^4(\log p)^2) = O((\log p)^6),$$

where $\omega \equiv \omega(p - 1)$ is the number of distinct prime factors of $p - 1$. E. Bach conjectured that the least primitive root for prime p is $O((\log p)(\log \log p))$, [24].

Let us define four invertible functions on $(\mathrm{Z}/(p))^*$ determined by elements of the group

$$\begin{cases} R(x) = r^x \bmod p, & r \in S_p^{(r)}, \\ E(x) = x^e \bmod p, & e \in (\mathrm{Z}/\varphi(p))^*, \\ M(x) = m \cdot x \bmod p, & m,\ x\ \in (\mathrm{Z}/(p))^*, \\ C_n(x) \bmod p, & \gcd(n, p^2 - -1) = 1, \end{cases} \tag{3}$$

where p is a prime number, $C_n(x)$ are Chebyshev polynomials of the first kind, [25]. The exponential function $R(x)$ is determined by the primitive root r of the group $(\mathrm{Z}/(p))^*$. The monomial function $E(x)$ is determined by the element e from $(\mathrm{Z}/\varphi(p))^*$. Each of the function (3) defines permutation of the set $(\mathrm{Z}/(p))^*$. By R, E, M, C_n we denote the permutation matrices determined by the functions $R(x)$, $E(x)$, $M(x)$, $C_n(x)$. Composition of these functions corresponds to the multiplication of permutation matrices. For example, the composition of two functions $R_1(x)$ and $R_2(x)$, defined by $R_1 \circ R_2(x) = R_2(R_1(x))$, corresponds to the multiplication of matrices $R_1 R_2$. We denote by $G_p^{(r)}$ the permutation group determined by the exponential functions $R(x)$. By $G_p^{(e)}$, $G_p^{(m)}$, $G_p^{(C)}$ we denote the groups generated by the functions $E(x)$, $M(x)$ and $C_n(x)$ respectively. The group $G_p^{(r)}$ is non-abelian, and $G_p^{(e)}$, $G_p^{(m)}$ and $G_p^{(C)}$ are abelian subgroups of $G_p^{(r)}$ for prime $p > 11$.

4 Extensions of Diffie-Hellman Protocol

The first extension of the Diffie-Hellman key exchange protocol utilises the exponential and monomial functions defined in (3).

The DH $_{r,e}$ algorithm.

1. *The users* A *and* B *agree the primitive root* r *from* $(\mathrm{Z}/(p))^*$ *and the element* $e \in (\mathrm{Z}/\varphi(p))^*$.
2. *The user* A *selects a number* a *, calculates* $r^{a^e} \bmod p$ *and sends it to* B. *The user* B *selects a number* b*, calculates* $r^{b^e} \bmod p$ *and sends it to* A.
3. *For given* a^e *and* r^{b^e} *the user* A *calculates*

$$k_A = (r^{b^e})^{a^e} = r^{b^e a^e} = r^{(ba)^e} \bmod p.$$

For given b^e and r^{a^e} *the user* B *calculates*

$$k_B = (r^{a^e})^{b^e} = r^{a^e b^e} = r^{(ab)^e} \bmod p.$$

The common key is $k_{a,b} = r^{(ab)^e} \bmod p$.

Let's illustrate the algorithm with a simple example on the $(\mathrm{Z}/(4441))^*$ group. Let $r = 21, e = 53, a = 121, b = 61 \in (\mathrm{Z}/(4441))^*$ then $a^e = 3535$, $r^{b^e} = 3209$ and $b^e = 972$, $r^{a^e} = 862 \mod 4441$. The common key is $k_A = 3209^{3535} = k_B = 862^{972} = 385 \mod 4441$.

The attacker can intercept the numbers r, e and $u = r^{a^e} \bmod p$. To decipher the number $a^e \bmod p$ he has to solve the logarithmic equation $\log_r(u) =$

$w \bmod \varphi(p)$. Using encryption for multiple exponents $\{e_i\}_i$ we force the attacker to decipher the secret a, which require to solve set of equations $\log_r(u) = w \bmod \varphi(p)$, $w = a^{e_i} \bmod p$ with two unknown parameters w and a.

The next modification of the Diffie-Hellman protocol utilizes composition of the exponential functions from (3).

The DH $_{2r}$ algorithm.

1. *The users* A *and* B *agree two primitive roots* r_1, r_2 *from* $(\mathrm{Z}/(p))^*$.
2. *The user* A *selects a number* a *, calculates* $r_1^{r_2^a} \bmod p$ *and sends it to* B.
   The user B *selects a number* b *, calculates* $r_1^{r_2^b} \bmod p$ *and sends it to* A.
3. *For given* $r_1^{r_2^b} \bmod p$ *and* $r_2^a \bmod p$ *the user* A *calculates*
$$k_A = (r_1^{r_2^b})^{r_2^a} = r_1^{r_2^a r_2^b} = r_1^{r_2^{a+b}} \bmod p.$$
   For given $r_1^{r_2^a} \bmod p$ *and* $r_2^b \bmod p$ *the user* B *calculates*
$$k_B = (r_1^{r_2^a})^{r_2^b} = r_1^{r_2^a r_2^b} = r_1^{r_2^{a+b}} \bmod p.$$
   The common key is $k_{a,b} = r_1^{r_2^{a+b}} \bmod p$.

The following example illustrates how the algorithm works. Let $r_1 = 44, r_2 = 94, a = 121, b = 61 \in (\mathrm{Z}/(4441))^*$ then $r_1^{r_2^b} = 939$, $r_2^a = 3386$ and $r_1^{r_2^a} = 2593$, $r_2^b = 3031$, $\bmod 4441$. The common key is $k_A = 939^{3386} = k_B = 2593^{3031} = 450 \bmod 4441$.

The attacker has the three numbers r_1, r_2 and $u = r_1^{r_2^a} \bmod p$. To decipher the secret a he has to solve the set of two equations for discrete logarithm $\log_{r_1}(u) = w \bmod \varphi(p)$ and $\log_{r_2}(w) = a \bmod \varphi(p)$ for unknown variables w and a.

The third modification of the Diffie-Hellman protocol utilizes the inverse of the exponential function and the monomial functions from (3).

The DH $_{e,\log}$ algorithm.

1. *The users* A *and* B *agree the primitive root* r *from* $(\mathrm{Z}/(p))^*$ *and the element* $e \in (\mathrm{Z}/\varphi(p))^*$.
2. *The user* A *selects a primitive root* r_a*, calculates* $(\log_{r_a}(r))^e \bmod \varphi(p)$ *and sends it to* B.
   The user B *selects a primitive root* r_b*, calculates* $(\log_{r_b}(r))^e \bmod \varphi(p)$ *and sends it to* A.
3. *For given* $(\log_{r_b}(r))^e$ *and* $(\log_r(r_a))^e$ *the user* A *calculates*
$$k_A = (\log_{r_b}(r))^e(\log_r(r_a))^e = (\log_{r_b}(r_a))^e \bmod \varphi(p).$$
   For given $(\log_{r_a}(r))^e$ *and* $(\log_r(r_b))^e$ *the user* B *calculates*
$$k_B = (\log_{r_a}(r))^e(\log_r(r_b))^e = (\log_{r_a}(r_b))^e \bmod \varphi(p).$$
   Because $(\log_{r_b}(r_a))^e = (\log_{r_a}(r_b))^{-e} \bmod \varphi(p)$ *the common key is*
$$k(r_a, r_b) = (\log_{r_b}(r_a))^e \bmod \varphi(p).$$

As an example we will calculate the common key $k(r_a, r_b)$ on the $(\mathrm{Z}/(4441))^*$ group. Let $e = 53, r = 21, r_a = 104, r_b = 168 \in (\mathrm{Z}/(4441))^*$ then $\log_{r_b}(r) = 2527$, $\log_r(r_a) = 1913$, $\log_{r_a}(r) = 3377$, $\log_r(r_b) = 1063 \mod 4440$. Bacause $k_A = 2527^{53}\ 1913^{53} = 2231$ and $k_B = 3377^{53}\ 1063^{53} = 3431 \mod 4440$ then the common key is $k(r_a, r_b) = 2231 \mod 4440$.

The attacker has primitive root r, exponent e and $u_a = (\log_{r_a}(r))^e \bmod \varphi(p)$. To determine r_a it has to solve the set of equations $u_a = (w_a)^e \bmod \varphi(p)$ and $w_a = \log_{r_a}(r) \bmod \varphi(p)$ for two unknown variables w_a and r_a.

5 The Key Agreement Protocol on Permutation Group

Let us denote by $V((\mathrm{Z}/(p))^*)$ the set of ordered $\varphi(p)$-tuples with elements from the group $(\mathrm{Z}/(p))^*$. By $v_0 = (1, \ldots, \varphi(p)) \in V((\mathrm{Z}/(p))^*)$ we denote the tuple ordered in ascending order. The functions from (3) define the permutation of $V((\mathrm{Z}/(p))^*)$

$$\begin{cases} Rv_0 = (r^1, \ldots, r^{p-1}) \bmod p, \;\; r \in S_p^{(r)}, \\ Ev_0 = (1^e, \ldots, (p-1)^e) \bmod p, \;\; e \in (\mathrm{Z}/\varphi(p))^*, \\ Mv_0 = (m \cdot 1, \ldots, m \cdot (p-1)) \bmod p, \;\; m \in (\mathrm{Z}/(p))^*, \\ C_n v_0 = (C_n(1), \ldots, C_n(p-1)) \bmod p, \;\; \gcd(n, p^2 - -1) = 1. \end{cases} \tag{4}$$

The sets of matrices (4) we denote as $S_p^{(R)}$, $S_p^{(E)}$, $S_p^{(M)}$ and $S_p^{(C)}$ respectively. From the composition rules of the functions $R_i \circ R_j(x) = r_j^{r_i^x} \bmod p$ follows the non-commutativity of the group $G_p^{(r)}$ generated by the matrices R, i.e., $R_i R_j \neq R_j R_i \in G_p^{(r)}$. From the composition rules of the monomials $E_i \circ E_j(x) = (x^{e_i})^{e_j} = x^{e_i e_j} \bmod p$, $x \in (\mathrm{Z}/(p))^*$, $e_i, e_j \in (\mathrm{Z}/\varphi(p))^*$, follows the commutativity of the group $G_p^{(e)}$, i.e., $E_i E_j = E_j E_i$. The matrices M_i, $i \in [1, \varphi(p)]$ generate the finite abelian group $G_p^{(m)}$ isomorphic to $(\mathrm{Z}/(p))^*$, called the automorphism group of $(\mathrm{Z}/(p))^*$, [26]. From the equation $R \circ E(x) = r^{xe} = (r^e)^x \bmod p$ it follows that $RE \in S_p^{(R)}$ which implies that $G_p^{(e)} \subset G_p^{(r)}$. The formula (1) written in terms of matrices has the form

$$S_p^{(R)} = \{RE_i\}_{i=1}^{\varphi(\varphi(p))}.$$

From the equations $E \circ M(x) = M(E(x)) = mx^e \bmod p$ and $M \circ E(x) = E(M(x)) = (mx)^e \bmod p$ it follows that the abelian groups $G_p^{(e)}$ and $G_p^{(m)}$ does not commute, i.e., $[G_p^{(e)}, G_p^{(m)}] \neq 0$. For $n = 11$ the group $G_{11}^{(r)}$ is the alternating group of the set of ten elements. For prime number $p > 11$ the group $G_p^{(r)}$ is the symmetric group $\mathrm{Sym}((\mathrm{Z}/(p))^*)$ of the set $(\mathrm{Z}/(p))^*$. The orders of the groups $G_p^{(e)}$ and $G_p^{(m)}$ are equal to $\varphi(\varphi(p))$ and $\varphi(p)$ respectively.

The construction of the symmetric cryptographic key on the group $G_p^{(r)}$ was motivated by the AAG symmetric key exchange protocol, [3,4]. The cryptographic key exchange we define by the formula

$$k_{a,b} = a_0 b_{N+1} a_{N+1}^{-1} b_0^{-1},$$

where $a_0, b_{N+1}, a_{N+1}, b_0 \in G_p^{(r)}$. The construction of the protocol can be explained on the following example. The users A and B agree the set of elements $S^{(g)} = \{g_{0,1}, g_{1,1}, g_{1,2}, g_{2,1}\}$ from the group $G_p^{(r)}$. The user A selects the set $S_a = \{a_0, a_1, a_2, a_3\}$, such that $a_3 = g_{0,1}g_{1,1}g_{2,1}$, calculates

$$S_a^{(g)} = \{u_{0,1} = a_0 g_{0,1} a_1^{-1}, \begin{array}{c} u_{1,1} = a_1 g_{1,1} a_2^{-1}, \\ u_{1,2} = a_1 g_{1,2} a_2^{-1}, \end{array} u_{2,1} = a_2 g_{2,1} a_3^{-1}\}$$

and sends $S_a^{(g)}$ to B. Similarly, the user B selects the set $S_b = \{b_0, b_1, b_2, b_3\}$, such that $b_3 = g_{0,1}g_{1,2}g_{2,1}$, calculates

$$S_b^{(g)} = \{w_{0,1} = b_0 g_{0,1} b_1^{-1}, \begin{array}{c} w_{1,1} = b_1 g_{1,1} b_2^{-1}, \\ w_{1,2} = b_1 g_{1,2} b_2^{-1}, \end{array} w_{2,1} = b_2 g_{2,1} b_3^{-1}\}$$

and sends $S_b^{(g)}$ to A. The user A, using the formula $w_{0,1}w_{1,1}w_{2,1} = b_0 a_3 b_3^{-1}$, calculates the common key $k_{a,b}^{-1} = (b_0 a_3 b_3^{-1} a_0^{-1})^{-1}$. The user B, using the formula $u_{0,1}u_{1,2}u_{2,1} = a_0 b_3 a_3^{-1}$, calculates the common key $k_{a,b} = (a_0 b_3 a_3^{-1}) b_0^{-1}$. In this trivial example, the attacker can easily calculate the cryptographic key $k_{a,b}$ because for a given matrices g and u from the sets $S^{(g)}$, $S_a^{(g)}$ to determine the unknown variable a_0 it's enough to solve the set of equations

$$\begin{cases} a_0 g_{0,1} = u_{0,1} a_1, \\ a_1 g_{1,1} = u_{1,1} a_2, \\ a_1 g_{1,2} = u_{1,2} a_2, \\ a_2 g_{2,1} = u_{2,1} a_3. \end{cases}$$

Because we know the formula $a_3 = g_{0,1}g_{1,1}g_{2,1}$ the set of equations can be trivially solved. In the proposed algorithm, we hide the last variable in the set S_a in order to make it difficult to find the element a_0 and the key $k_{a,b}$. To do this, we introduce a graph G_{2N}^{cipher} which nodes are elements of the set

$$S_x^{(g)} = \{u_{0,1} = x_0 g_{0,1} x_1^{-1}, \begin{array}{c} u_{i,1} = x_i g_{i,1} x_{i+1}^{-1}, \\ u_{i,2} = x_i g_{i,2} x_{i+1}^{-1}, \end{array} u_{N,1} = x_N g_{N,1} x_{N+1}^{-1}\}_{i=1}^{N-1},$$

where $x_i, g_{i,1}, g_{i,2} \in G_p^{(r)}$. Example of the such graph G_6^{cipher} build of six nodes is shown on Fig. 1. A path

$$\begin{tikzcd}
 & x_1 g_{1,1} x_2^{-1} \arrow[r] \arrow[ddr] & x_2 g_{2,1} x_3^{-1} \arrow[dr] & \\
x_0 g_{0,1} x_1^{-1} \arrow[ur] \arrow[dr] & & & x_3 g_{3,1} x_4^{-1} \\
 & x_1 g_{1,2} x_2^{-1} \arrow[r] \arrow[uur] & x_2 g_{2,2} x_3^{-1} \arrow[ur] &
\end{tikzcd}$$

Fig. 1. The graph G_6^{cipher} for generation of the cryptographic key.

$$p_a(u) = (u_{0,1}, \ldots, (u_{i,1} || u_{i,2}), \ldots, u_{N,1})$$

from the root node $u_{0,1}$ to the leaf node $u_{N,1}$ defines uniquely the product of matrices

$$x_0 = u_{0,1} \dots (u_{i,1}||u_{i,2}) \dots u_{N,1}, \;\; i \in [1, N-1], \tag{5}$$

from which follows the form of the a_{N+1} element

$$x_{N+1} = g_{0,1} \dots (g_{i,1}||g_{i,2}) \dots g_{N,1}, \;\; i \in [1, N-1]. \tag{6}$$

The expression $(g_{i,1}||g_{i,2})$ means that in the product (6) either the matrix $g_{i,1}$ or $g_{i,2}$ appears, depending on the selected path in G_{2N}^{cipher}. The number of possible values for the matrix $x_{N,1}$ is equal to the number of all possible paths from the root node $u_{0,1}$ to the leaf node $u_{N,1}$. For the graph G_{2N}^{cipher} build of $2N$ nodes there are 2^{N-1} such paths. For sufficiently large number of nodes in G_{2N}^{cipher} one can regard the matrix x_{N+1} as unknown parameter. On Fig. 1, it is shown the graph G_6^{cipher} build of six nodes, $N = 3$, in which there are four paths between the root and the leaf node. To calculate the key $k_{a,b}$, calculated for two paths $p_a(u)$ and $p_b(w)$ in the graph G_{2N}^{cipher}, the attacker should solve the set of matrix equations

$$\begin{cases} x_i \; g_{i-1,1}^{-1} g_{i-1,2} = u_{i-1,1}^{-1} u_{i-1,2} \; x_i, \\ x_i \; g_{i,1} g_{i,2}^{-1} = u_{i,1} u_{i,2}^{-1} \; x_i, \quad i \in [2, N], \end{cases} \tag{7}$$

where $g_{i,1}, g_{i,2}, u_{i,1}, u_{i,2}$ are known parameters. For a given solution $x_i = a_i$ of (7), using the formulas for $u_{i,1}$, $u_{i,2}$ from $S_a^{(g)}$ and $x = a_i^{-1} x_i$ we obtain the following set of matrix equations

$$\begin{cases} x \; g_{i-1,1}^{-1} g_{i-1,2} = g_{i-1,1}^{-1} g_{i-1,2} \; x \\ x \; g_{i,1} g_{i,2}^{-1} = g_{i,1} g_{i,2}^{-1} \; x, \quad i \in [2, N] \end{cases} \tag{8}$$

If the solution of (8) is trivial then the solution of (7) is unique. For a non-trivial solution g_0 of the equations (8), also any matrix g_0^k is a solution of (8). The two solutions a_i and a_i' of (7) are related by the formula

$$a_i' = a_i \; (g_0)^k, \quad k \in [1, |g_0|],$$

where g_0 is the solution of (8), i.e., belongs to the centralizer $C_{g_{i,1} g_{i,2}^{-1}}$ of the element $g_{i,1} g_{i,2}^{-1}$ and $|g_0|$ is order of the element g_0. If centralizers $C_{g_{i,1} g_{i,2}^{-1}}$, $i \in [2, N]$, are nontrivial then the solution of the set of equations (7) is not unique. The security of the proposed algorithm depends on the difficulty of finding the proper solution of the equations (7), i.e., depends on the number of solutions of (7). Below we give the detailed description of the algorithm.

The key agreement algorithm in $G_p^{(r)}$.

1. *The users* A *and* B *agree the set of elements* $S^{(g)} = \{g_{0,1}, g_{i,1}, g_{i,2}, g_{N,1}\}_{i=1}^{N-1}$ *from the group* $G_p^{(r)}$ *with non trivial centralizers* $C_{g_{i,1} g_{i,2}^{-1}}$.
2. *The user* A *selects the set* $S_a = \{a_i\}_{i=0}^{N+1} \subset G_p^{(r)}$ *and the path* $p_a(u)$ *in the graph* G_{2N}^{cipher}, *such that formula* (5) for a_0 *is satisfied. The user*

A *calculates* $S_a^{(g)} = \{u_{0,1}, u_{i,1}, u_{i,2}, u_{N,1}\}_{i=1}^{N-1}$ *and sends the set* $S_a^{(g)}$ *to the user* B. *Similarly, the user* B *selects the set* $S_b = \{b_i\}_{i=0}^{N+1} \subset G_p^{(r)}$, *and the path* $p_b(w)$ *in the graph* G_{2N}^{cipher}, *such that formula* (5) *for* b_0 *is satisfied. The user* B *calculates* $S_b^{(g)} = \{w_{0,1}, w_{i,1}, w_{i,2}, w_{N,1}\}_{i=1}^{N-1}$ *and sends the set* $S_b^{(g)}$ to A.

3. *For the selected path* $p_a(u)$ *the user* A *calculates* $k_{a,b}^{-1} = ((b_0 a_{N+1} b_{N+1}^{-1}) a_0^{-1})^{-1}$.
   For the selected $p_b(w)$ *the user* B *calculates* $k_{a,b} = (a_0 b_{N+1} a_{N+1}^{-1}) b_0^{-1}$.
   The common key is $k_{a,b}$.

We apply the proposed algorithm to the group $G_{17}^{(r)}$, which is isomorphic to the symmetric group of the set of $(\mathrm{Z}/(17))^*$. The graph G_8^{cipher}, $N = 4$, is build of eight nodes. There are 2^3 paths in G_8^{cipher}. The agreed set $S^{(g)}$ and selected secret sets S_a and S_a are

$$\begin{aligned}
&S^{(g)} = \{E_3, E_5, E_7, E_9, E_{11}, E_{13}, E_{15}, E_3\},\\
&S_a = \{R_{14}, R_{12}, R_{11}, R_{10}, R_7, E_3E_5E_{11}E_{15}E_3\},\\
&S_b = \{R_3, R_5, R_7, R_{10}, R_{11}, E_3E_7E_9E_{13}E_3\},
\end{aligned}$$

where the matrices $R_j \in G_{17}^{(r)}$ are determined by the primitive roots $j \in S_{17}^{(r)}$ and $E_i \in G_{17}^{(e)}$. The paths selected by the users A and B are

$$\begin{aligned}
&p_a(u) = (u_{0,1}, u_{1,1}, u_{2,2}, u_{3,2}, u_{4,1}),\\
&p_b(w) = (w_{0,1}, w_{1,2}, w_{2,1}, w_{3,1}, w_{4,1}),
\end{aligned}$$

where $u_{i,j} = a_i g_{i,j} a_{i+1}^{-1}$ and $w_{i,j} = b_i g_{i,j} b_{i+1}^{-1}$, $i \in [0, 4]$, $j = 1, 2$. The common key calculated by communicating parties is given by the formula

$$k_{a,b} = R_{14}(E_3E_7E_9E_{13}E_3)(E_3E_5E_{11}E_{15}E_3)^{-1}R_3^{-1},$$

and can be written in matrix form $k_{a,b}v_0 = (3, 6, 9, 12, 15, 2, 5, 8, 11, 14, 1, 4, 7, 10, 13, 16)$, where $v_0 = (1, \ldots, 16)$.

6 Conclusions

We proposed three cryptographic key exchange protocols of the Diffie-Hellman type based on the exponential and logarithmic functions over the multiplicative group of integers modulo prime number p. The security of the proposed protocols is based on the computational complexity in solving a set of congruence equations containing the discrete logarithm. For the multiplicative group of integer numbers modulo p we constructed the non-commutative group $G_p^{(r)}$ of their automorphisms. On the defined group we constructed a non-commutative key exchange protocol similar to the Anshel-Anshel-Goldfeld key exchange scheme. The security of the proposed protocols is based on the difficulty of finding path in a defined cipher graph G_{2N}^{cipher} build of $2N$ nodes and solution of the set of certain matrix equations in $G_p^{(r)}$.

References

1. Diffie, W., Hellman, M.E.: New Directions in Cryptography. IEEE Trans. Inform. Theor. **IT-22**(6), 644–654 (1976)
2. Rescorla, E.: Diffie-Hellman Key Agreement Method, RFC 2631, http://www.rfc-editor.org (1999)
3. Anshel, I., Anshel, M., Goldfeld, D.: An algebraic method for public-key cryptography. Math. Res. Lett. **6**, 1–5 (1999)
4. Anshel, I., Anshel, M., Goldfeld, D.: Non-abelian key agreement protocols. Discrete Appl. Math. **130**, 3–12 (2003)
5. Dehn, M.: Über unendliche diskontinuierliche Gruppen. Math. Annalen **71**, 116–144 (1911)
6. Myasnikov, A., Shpilrain, V., Ushakov, A.: Non-commutative Cryptography and Complexity of Group-theoretic Problems, Mathematical Surveys and Monographs, vol. 177, AMS (2011)
7. Wagner, N.R., Magyarik, M.R.: A public-key cryptosystem based on the word problem. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 19–36. Springer, Heidelberg (1985). https://doi.org/10.1007/3-540-39568-7_3
8. Ko, K.H., Lee, S.J., Cheon, J.H., Han, J.W., Kang, J., Park, C.: New public-key cryptosystem using braid groups. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 166–183. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44598-6_10
9. Anshel, I., Anshel, M., Fisher, B., Goldfeld, D.: New key agreement protocols in braid group Ccyptography. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 13–27. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45353-9_2
10. Partala, J.: Algebraic generalization of Diffe-Hellman key exchange. J. Math. Cryptol. **12**, 1–21 (2018)
11. Chefranov, A. G., Mahmoud, A. Y.: Commutative Matrix-based Diffie-Hellman-Like Key-Exchange Protocol. In: Proceedings of the 28th International Symposium on Computer and Information Sciences In: E. Gelenbe, R. Lent (eds.), Springer, pp. 317–324, (2013). https://doi.org/10.1007/978-3-319-01604-7_31
12. Cheon, J.H., Jun, B.: A polynomial time algorithm for the braid Diffie-Hellman Conjugacy Problem. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 212–225. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_13
13. Eftekhari, M.: A Diffie-Hellman key exchange protocol using matrices over non-commutative rings. Groups Complex. Cryptol. **4**, 167–176 (2012)
14. Cash, D., Kiltz, E., Shoup, V.: The twin Diffie-Hellman problem and applications. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 127–145. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_8
15. Blake, I.F., Garefalakis, T.: On the complexity of the discrete logarithm and Diffie-Hellman problems. J. Complexity **20**, 148–170 (2004)
16. Bresson, E., Chevassut, O., Pointcheval, D.: The group Diffie-Hellman problems. In: Nyberg, K., Heys, H. (eds.) SAC 2002. LNCS, vol. 2595, pp. 325–338. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36492-7_21
17. Steiner, M., Tsudik, G., Waidner, M.: Diffe-Hellman key distribution extended to group communication. In: Proceedings of ACM CCS '96, ACM Press, pp. 31–37 (1996)
18. Dougherty, D. J., Guttman, J. D.: Symbolic Protocol Analysis for Diffie-Hellman, arXiv:1202.2168 (2012)

19. Niven, I. M., Zuckerman, H. S., Montgomery, H. L.: An introduction to the theory of numbers, John Wiley & Sons (1991)
20. Bach, E.: Comments on search procedures for primitive roots. Math. Comp. **66**(220), 1719–1727 (1997)
21. Burgess, D.A.: Character sums and primitive roots in finite fields. Proc. London Math. Soe. **3**(17), 11–25 (1967)
22. Shparlinski, I.: On finding primitive roots in finite fields. Theor. Comput. Sci. **157**, 273–275 (1996)
23. Shoup, V.: Searching for Primitive Roots in Finite Fields. Math. Comput. **58**(197), 369–380 (1992)
24. Bach, E., Shallit, J.: Algorithmic number theory, Volume I: Efficient Algorithms, MIT Press (1996)
25. Lidl, R., Niederreiter, H., Cohn, P. M.: Finite Fields, Cambridge University Press (1997)
26. Rose, H.E.: A Course on Finite Groups, Springer-Verlag (2009). https://doi.org/10.1007/978-1-84882-889-6