A new key exchange protocol based on the decomposition problem

Vladimir Shpilrain and Alexander Ushakov

Abstract. In this paper we present a new key establishment protocol based on the decomposition problem in non-commutative groups which is: given two elements w, w_1 of the platform group G and two subgroups $A, B \subseteq G$ (not necessarily distinct), find elements $a \in A,\ b \in B$ such that $w_1 = awb$. Here we introduce two new ideas that improve the security of key establishment protocols based on the decomposition problem. In particular, we conceal (i.e., do not publish explicitly) one of the subgroups A, B, thus introducing an additional computationally hard problem for the adversary, namely finding the centralizer of a given finitely generated subgroup.

1. Introduction

In search of a more efficient and/or secure alternative to established cryptographic protocols (such as RSA), several authors have come up with public key establishment protocols as well as with complete public key cryptosystems based on allegedly hard *search problems* from combinatorial (semi)group theory, including the conjugacy search problem [**1, 15**], the homomorphism search problem [**14**], [**18**], the decomposition search problem [**5, 15, 17**], the subgroup membership search problem [**19**].

In this paper, we focus on the decomposition search problem which we subsequently call just the decomposition problem. The problem is: given two elements w, w_1 of the platform group G and two subgroups $A, B \subseteq G$ (not necessarily distinct), find elements $a \in A,\ b \in B$ such that $w_1 = awb$.

It is straightforward to arrange a key establishment protocol based on this problem (see [**5, 15, 17**]), assuming that $ab = ba$ for any $a \in A,\ b \in B$:

(0) One of the parties (say, Alice) publishes a random element $w \in G$ (the "base" element).

(1) Alice chooses $a_1, a_2 \in A$ (Alice's private keys) and sends $a_1 w a_2$ to Bob.

(2) Bob chooses $b_1, b_2 \in B$ (Bob's private keys) and sends $b_1 w b_2$ to Alice.

(3) Alice computes

$$K_a = a_1 b_1 w b a_2 b_2$$

and Bob computes

$$K_b = b_1 a_1 w a_2 b_2.$$

If $a_i b_i = b_i a_i$, then $K_a = K_b$ in G. Thus Alice and Bob have a *shared secret key.*

2000 *Mathematics Subject Classification.* 94A60, 20F05, 20F36, 68P25.

Research of the first author was partially supported by the NSF grant DMS-0405105.

Security of such a protocol will, of course, depend on a particular platform group G (at the very least, G has to be non-commutative). It appears that for braid groups (which are a popular choice for the platform), the so-called *length attacks* present a serious threat, see e.g. [**8, 12, 13, 16**].

In this paper, we introduce two new ideas that improve the security of key establishment protocols based on the decomposition problem:

(i) We conceal one of the subgroups A, B.

(ii) We make Alice choose her left private key a_1 from one of the subgroups A, B, and her right private key a_2 from the other subgroup. Same for Bob.

These two improvements together will obviously foil any length attacks. We give a complete description of our protocol in the following Section 2; here we just sketch the main idea.

Let G be a group and $g \in G$. Denote by $C_G(g)$ the *centralizer* of g in G, i.e., the set of elements $h \in G$ such that $hg = gh$. For $S = \{g_1, \ldots, g_k\} \subseteq G$, $C_G(g_1, \ldots, g_k)$ denotes the centralizer of S in G, which is the intersection of the centralizers $C_G(g_i), i = 1, ..., k$.

Now, given a public $w \in G$, Alice privately selects $a_1 \in G$ and publishes a subgroup $B \subseteq C_G(a_1)$ (we explain why computing B is easy). Similarly, Bob privately selects $b_2 \in G$ and publishes a subgroup $A \subseteq C_G(b_2)$. Alice then selects $a_2 \in A$ and sends $w_1 = a_1 w a_2$ to Bob, while Bob selects $b_1 \in B$ and sends $w_2 = b_1 w b_2$ to Alice.

Thus, in the first transmission, say, the adversary faces the problem of finding a_1, a_2 such that $w_1 = a_1 w a_2$, where $a_2 \in A$, but there is no explicit indication of where to choose a_1 from. Therefore, before arranging something like a length attack in this case, the adversary would have to compute the centralizer $C_G(B)$ first (because $a_1 \in C_G(B)$), which is usually a hard problem by itself.

2. The protocol

In this section we give a formal description of our protocol, but first we introduce one more piece of notation. As it is common in public key exchange based on abstract groups, when transmitting an element $g \in G$ of a group, one actually uses its *normal form* $N(g)$ which is a sequence of symbols uniquely defined for a given g. A specific way of constructing such a sequence depends, of course, on a particular platform group G which we discuss in subsequent sections of our paper.

Our protocol is the following sequence of steps.

Protocol:

(1) Alice chooses an element $a_1 \in G$ of length l, chooses a subgroup of $C_G(a_1)$, and publishes its generators $A = \{\alpha_1, \ldots, \alpha_k\}$ (see the following subsection 2.1 for specifications).

(2) Bob chooses an element $b_2 \in G$ of length l, chooses a subgroup of $C_G(b_2)$, and publishes its generators $B = \{\beta_1, \ldots, \beta_m\}$ (see the following subsection 2.1 for specifications).

(3) Alice chooses a random element a_2 from $\langle \beta_1, \ldots, \beta_m \rangle$ and sends the normal form $P_A = N(a_1 w a_2)$ to Bob.

(4) Bob chooses a random element b_1 from $\langle \alpha_1, \ldots, \alpha_k \rangle$ and sends the normal form $P_B = N(b_1 w b_2)$ to Alice.

(5) Alice computes $K_A = a_1 P_B a_2$.

(6) Bob computes $K_B = b_1 P_A b_2$.

Since $a_1 b_1 = b_1 a_1$ and $a_2 b_2 = b_2 a_2$, we have $K = K_A = K_B$, the shared secret key.

2.1. Suggested values of parameters. We suggest to use the following values of parameters in the above protocol: $G = B_n$, the group of braids on n strands (see our Section 4); $n = 64$; $l = 1024$. At Step (1) of the protocol Alice generates (a_1, A) and at Step (2) Bob generates (b_2, B), both using the algorithm from [**7**] for computing centralizers (actually, there is no need to compute the whole centralizer, just a couple of elements are sufficient).

3. Requirements on the platform group G

In this section we discuss possible attacks on the protocol described in the previous section, and also put together some requirements on the platform group G.

To break the protocol it is sufficient to find either Alice's or Bob's private key which may be accomplished as follows:

> **Attack on Alice's private key.** Find an element a_1' which commutes with every element of the subgroup $\langle A\rangle$ and an element $a_2' \in \langle B\rangle$, such that $P_A = N(a_1' w a_2')$. The pair (a_1', a_2') is equivalent to (a_1, a_2). (That means, $a_1' w a_2' = a_1 w a_2$, and therefore the pair (a_1', a_2') can be used by the adversary to get the shared secret key.)
> **Attack on Bob's private key.** Find an element $b_1' \in \langle A\rangle$ and an element b_2' which commutes with every element of the subgroup $\langle B\rangle$, such that $P_B = N(b_1' w b_2')$. The pair (b_1', b_2') is equivalent to (b_1, b_2).

Consider the attack on Alice's private key (the other one is similar). The most obvious way to carry out such an attack is the following:

(A1) Compute the centralizer $C_G(A)$.
(A2) Solve the search version of the membership problem in the double coset $C_G(A) \cdot w \cdot \langle B\rangle$

To make the protocol secure, we want both these problems to be computationally hard. For the problem (A2) to be hard, it is necessary for the centralizer $C_G(A)$ to be large. Otherwise, the adversary can use the "brute force" attack, i.e., enumerate all elements of $C_G(A)$ and find candidates for b_2' (assuming that the decisional membership problem in the subgroup B is efficiently solvable).

Thus the platform group G should satisfy at least the following properties in order for our key establishment protocol to be efficient and secure.

(P1) G should be a non-commutative group of exponential growth. The latter means that the number of elements of length n in G is exponential in n; this is needed to prevent attacks by complete exhaustion of the key space.
(P2) There should be an efficiently computable normal form for elements of G.
(P3) It should be computationally easy to perform group operations (multiplication and inversion) on normal forms.
(P4) It should be computationally easy to generate pairs $(a,\ \{a_1, \ldots, a_k\})$ such that $aa_i = a_i a$ for each $i = 1, \ldots, k$. (Clearly, in this case the subgroup generated by $a_1, \ldots, a_k$ centralizes a).
(P5) For a generic set $\{g_1, \ldots, g_k\}$ of elements of G it should be difficult to compute

$$C(g_1, \ldots, g_n) = C(g_1) \cap \ldots \cap C(g_k).$$

(P6) Even if $H = C(g_1, \ldots, g_n)$ is computed, it should be hard to find $x \in H$ and $y \in H_1$ (where H_1 is some fixed subgroup given by a generating set) such that $xwy = w'$, i.e., to solve the membership search problem for a double coset.

4. Braid groups

In this section we consider a particular class of groups, namely braid groups, which were a popular choice for the platform of various cryptographic protocols in the last 6-7 years, starting with the seminal paper [**1**].

Let B_n be the group of braids on n strands and $X_n = \{x_1, \ldots, x_{n-1}\}$ the set of standard generators. Thus,

$$B_n = \langle x_1, \ldots, x_{n-1};\ x_i x_{i+1} x_i = x_{i+1} x_i x_{i+1},\ x_i x_j = x_j x_i \text{ for } |i-j| > 1 \rangle.$$

For more information on braid groups, we refer to the monographs [**2**], [**6**]; here we address the properties (P1)-(P6) from the previous section.

(P1) Braid groups B_n are non-commutative groups of exponential growth if $n \geq 3$.
(P2) There are several known normal forms for elements of B_n, including Garside normal form (see [**2**]) and Birman-Ko-Lee normal form [**3**]. Both of these forms are efficiently computable (in quadratic time with respect to the length of a given element).
(P3) There are quadratic time algorithms to multiply or invert normal forms of elements of B_n.
(P4) It is not so easy to compute the whole centralizer of an element g of G (cf. [**11**]). The number of steps required to compute $C_G(g)$ is proportional to $|SSS(g)|$, the size of the "super summit set" of g, which is typically huge. Nevertheless, there are approaches to finding "large parts" of $C_G(g)$, e.g. one can generate a sufficiently large part of $SSS(g)$ and pick several elements from there, see [**11**] for more details.
(P5) For a generic subgroup A it is hard to compute $C_G(A)$. The complexity of such computation is proportional to $|SS(A)|$, the size of the summit set of A (see [**7**]), which is typically huge.
(P6) There is no known solution to the membership search problem for double cosets $H \cdot w \cdot H'$ in braid groups. This problem, in theory, appears to be much more complicated (for generic subgroups H and H') than the conjugacy search problem.

5. Generating commuting elements in braid groups

In this section, we explain how one can efficiently generate elements commuting with a given element of a braid group, which is important for our protocol (steps (1), (2)).

In a group B_n, for $i < j$ define a braid word $\delta_{i,j} = x_i \ldots x_j$. Denote $\delta_{1,n-1}$ by just δ. Clearly for any braid word $w = x_{i_1}^{\varepsilon_1} \ldots x_{i_k}^{\varepsilon_k}$ such that $i_j < n-1-m$ for each $j = 1, \ldots, k$, one has

$$w^{\delta^m} = x_{i_1+m}^{\varepsilon_1} \ldots x_{i_k+m}^{\varepsilon_k} \text{ in } B_n.$$

Now given $w \in B_n$, we are going to generate a set C of elements commuting with w, as follows.

A. **Initial setup:**
- Take $n = 32$.
- Generate w_1 to be a random freely reduced braid word of length 40 in the generators $\{x_1, x_2, x_3\}$.
- For $j = 2, \ldots, 8$ generate random freely reduced braid words c_i of length 20 in the generators $\{x_1, x_2, x_3\}$ and put $w_j = c_j^{-1} w_1 c_j$. Let c_1 be ε, the empty word.
- Let $w = w_1 w_2^{\delta^4} w_3^{\delta^8} \ldots w_8^{\delta^{28}}$.

- Compute generators of the centralizer $C(w_1)$ (using technique from [**11**]). Clearly, $C(w_i) = C(w_1)^{c_i}$ for $i = 2, \dots, 8$. Let
$$C = C(w_1) \cup C(w_1)^{\delta^4 c_2} \cup \ldots \cup C(w_1)^{\delta^{28} c_8}.$$
- Add to C braid words $flip(1,2), \dots, flip(7,8)$, where
$$flip(i, i+1) = (\delta_{4i,4i-3}\delta_{4i+1,4i-2}\delta_{4i+2,4i-1}\delta_{4i+3,4i})^2.$$
These commute with w.

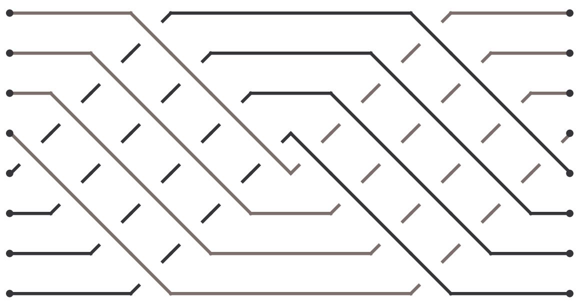

FIGURE 1. Flip.

- Add to C braid words $swap(1,2), \dots, swap(7,8)$, where $swap(i, i+1)$ is the following braid:
$$(c_i^{-1}c_{i+1})^{\delta^{(4i-4)}}(c_{i+1}^{-1}c_i)^{\delta^{(4i)}}\delta_{4i,4i-3}\delta_{4i+1,4i-2}\delta_{4i+2,4i-1}\delta_{4i+3,4i}.$$
These, too, commute with w.

B. **For more diffusion:** Generate a random braid word x of length 150 in the generators of B_n and conjugate w as well as elements of C by x.

The word w generated this way has length bounded by $8 \cdot 40 + 2 \cdot 150 = 620$. Also note that $|flip(i, i+1)| = 32$ and $|swap(i, i+1)| \le 20 + 20 + 20 + 20 + 16 = 96$. We can therefore select sufficiently many elements of length ≤ 100 from the set C and publish them (in Garside normal form).

6. Semantic security

In this section, we discuss *semantic security* of a cryptosystem that would be based on a shared key obtained in our protocol. Semantic security is the standard notion of security for encryption protocols, see [**10**].

Security of the protocol described in our Section 2 is based on the assumption that the following problem is computationally hard:

Given the public information w, P_A, and P_B it is hard to compute the shared key K.

This assumption is the *computational assumption of the protocol.* The stronger *decisional* version of this assumption would be:

Given w, P_A, and P_B, it is hard to distinguish the shared key K from a random element of the form awb.

We should point out that without this decisional assumption, it may still be possible to design a semantically secure encryption protocol in the "random oracle model" the same way it was done in [**15**, Section 3.3], namely, by employing a hash function $H : B_n \to \{0,1\}^k$ from the braid group to the message space. Still, it would be quite interesting to find out whether or not the shared key K obtained in our key establishment protocol can be directly used for semantically secure encryption.

The decisional assumption above appears to be wrong for most choices of w, P_A and P_B because of the following consideration. Since $P_A = a_1 w a_2$, we have $a_1 = P_A a_2^{-1} w^{-1}$. Therefore, $K = a_1 b_1 w b_2 a_2 = P_A a_2^{-1}(w^{-1}P_B)a_2$. Hence, K is a product of a public element P_A and a public element $w^{-1}P_B$ conjugated by an element from a subgroup $\{\beta_1, \ldots, \beta_k\}$.

It seems plausible that, for some choices of the keys, elements of this type can be distinguished from random elements of the form awb along the same lines it was done in [**9**] (in a different, but similar context). Indeed, if $w^{-1}P_B$ is not a *pure braid*, then it projects to a non-trivial permutation, call it ρ_B, under the natural homomorphism π from the braid group B_n onto the symmetric group S_n. Then the conjugate permutation $\pi(a_2)^{-1}\rho_B\pi(a_2)$ has the same cyclic structure as ρ_B does, and this gives away some information about the permutation $\pi(K) = \pi(P_A)\pi(a_2)^{-1}\rho_B\pi(a_2)$; for example, from knowing $\pi(P_A)$ and the cyclic structure of $\pi(a_2)^{-1}\rho_B\pi(a_2)$, one can get information about possible order of the permutation $\pi(K)$.

If both P_A and $w^{-1}P_B$ are pure braids, then it is possible to use other homomorphisms (e.g. pulling out a strand) to obtain some partial information; see [**9**] for details. If $w^{-1}P_B$ is a pure braid but P_A is not, then, again, the homomorphism π reveals partial information about the shared key K.

Acknowledgement. The second author is grateful to Alexei G. Myasnikov and Robert Gilman for their kind support and encouragement.

References

[1] I. Anshel, M. Anshel, D. Goldfeld, *An algebraic method for public-key cryptography*, Math. Res. Lett. **6** (1999), 287–291.

[2] J. S. Birman, *Braids, links and mapping class groups*, Ann. Math. Studies **82**, Princeton Univ. Press, 1974.

[3] J. S. Birman, K. H. Ko, S. J. Lee, *A new approach to the word and conjugacy problems in the braid groups*, Adv. Math. **139** (1998), 322–353.

[4] F. Celler, C. Leedham-Green, S. H. Murray, A. Niemeyer, E. A. O'Brien, *Generating random elements of a finite group*, Comm. Algebra **23** (1995), 4931–4948.

[5] J. C. Cha, K. H. Ko, S. J. Lee, J. W. Han, J. H. Cheon, *An Efficient Implementation of Braid Groups*, ASIACRYPT 2001, Lecture Notes in Comput. Sci. **2248** (2001), 144–156.

[6] D. B. A. Epstein, J. W. Cannon, D. F. Holt, S. V. F. Levy, M. S. Paterson, W. P. Thurston, *Word processing in groups.* Jones and Bartlett Publishers, Boston, MA, 1992.

[7] N. Franco, J. Gonzalez-Meneses, *Computation of Centralizers in Braid groups and Garside groups*, Rev. Mat. Iberoamericana **19** (2) (2003), 367–384.

[8] D. Garber, S. Kaplan, M. Teicher, B. Tsaban, U. Vishne, *Probabilistic solutions of equations in the braid group*, Advances in Applied Mathematics **35** (2005), 323–334.

[9] R. Gennaro and D. Micciancio, *Cryptanalysis of a pseudorandom generator based on braid groups*, in EUROCRYPT 2002, Lecture Notes in Comput. Sci. **2332** (2002), 1–13.

[10] S. Goldwasser and S. Micali, *Probabilistic encryption*, Journal of Computer and System Sciences **28** (1984), 270–299.

[11] J. Gonzalez-Meneses and B. Wiest, *On the structure of the centraliser of a braid*, Ann. Sci. École Norm. Sup. **37** (5) (2004), 729–757.

[12] D. Hofheinz and R. Steinwandt, *A practical attack on some braid group based cryptographic primitives*, in Public Key Cryptography, 6th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2003 Proceedings, Y.G. Desmedt, ed., Lecture Notes in Computer Science **2567**, pp. 187–198, Springer, 2002.

[13] J. Hughes and A. Tannenbaum, *Length-based attacks for certain group based encryption rewriting systems*, Workshop SECI02 Securitè de la Communication sur Intenet, September 2002, Tunis, Tunisia.
http://www.storagetek.com/hughes/

[14] D. Grigoriev, I. Ponomarenko, *Homomorphic public-key cryptosystems and encrypting boolean circuits*, preprint.
http://eprint.iacr.org/2003/025

[15] K. H. Ko, S. J. Lee, J. H. Cheon, J. W. Han, J. Kang, C. Park, *New public-key cryptosystem using braid groups*, Advances in cryptology—CRYPTO 2000 (Santa Barbara, CA), 166–183, Lecture Notes in Comput. Sci. **1880**, Springer, Berlin, 2000.

[16] A. G. Myasnikov, V. Shpilrain, and A. Ushakov, *A practical attack on some braid group based cryptographic protocols*, in CRYPTO 2005, Lecture Notes Comp. Sci. **3621** (2005), 86–96.

[17] V. Shpilrain and A. Ushakov, *Thompson's group and public key cryptography*, Lecture Notes Comp. Sc. **3531** (2005), 151–164.

[18] V. Shpilrain and G. Zapata, *Combinatorial group theory and public key cryptography*, Applicable Algebra in Engineering, Communication and Computing, to appear.
http://eprint.iacr.org/2004/242

[19] V. Shpilrain and G. Zapata, *Using the subgroup membership search problem in public key cryptography*, this volume.

Department of Mathematics, The City College of New York, New York, NY 10031
E-mail address: `shpil@groups.sci.ccny.cuny.edu`

Department of Mathematics, CUNY Graduate Center, New York, NY 10016
E-mail address: `aushakov@mail.ru`