AAECC (2006) 17: 195–203
DOI 10.1007/s00200-006-0001-1

Iris Anshel · Michael Anshel · Dorian Goldfeld

A linear time matrix key agreement protocol over small finite fields

Received: 5 September 2004 / Revised: 4 December 2004 / Published online: 16 March 2006

Abstract This note introduces Braid–Diffie–Hellman (BDH), a key agreement protocol employing matrices over small finite fields. The method employs colored Burau matrices and a refinement of a Diffie–Hellman type protocol. We discuss the security and examine performance parameters of BDH which permit linear time execution on platforms supporting basic algebraic primitives.

Keywords Colored Burau matrices · Key agreement protocol · Small finite fields · Linear time

1 Introduction

A central problem in contemporary cryptography is to develop a secure linear time key agreement protocol. For example, the classical Diffie–Hellman key agreement protocol requires us to compute $a^b \pmod p$ where a, b, p are each of bit-length n, and n is sufficiently large so that computations of discrete logs are infeasible. This computation requires about n squarings and $c \cdot n$ modular multiplies (where $c \sim 0.5$, on average), so the total number of arithmetic operations needed are about $\mathcal{O}\left(n^2 \log(n)\right)$ (see [7]). In this note we present Braid–Diffie–Hellman (BDH) a linear time, linear algebraic key agreement protocol employing matrices over small finite fields $\pmod p$. The method is motivated by ideas in [1,2], and is a refinement of the Diffie–Hellman type protocols in [4,9]. The BDH protocol employs

I. Anshel
Arithmetica Inc., 31 Peter Lynas Ct. Tenafly, NJ 07670, USA

M. Anshel
City College of New York, New York, NY 10031, USA

D. Goldfeld (✉)
Columbia University, New York, NY, USA
E-mail: goldfeld@columbia.edu

colored Burau matrices [6], which are studied in connection with the Artin braid group B_{N+1} [3]. We use the colored Burau matrices to specify the colored Burau group, a key space, and a one-way key extractor. This machinery allows the creation of two commuting one-way functions from the colored Burau group to the key space, and in turn the creation of a common shared secret. We examine performance parameters of BDH which permit linear–time execution on platforms supporting basic algebraic primitives. Attacks are briefly discussed. We shall show that BDH requires $\mathcal{O}(n)$ operations where n is the the size of the key and the key is a braid word written as a product of n Artin generators. The $\mathcal{O}$-constant depends on N and the prime p. Both of these are assumed to be small. For initial applications, we may take p to be an 8 bit prime and $N \leq 80$.

2 Background

A *finitely generated* group G is specified by a finite set of generators

$$g_1, g_2, \ldots, g_n$$

where every $g \in G$ is a product (also termed a word) in the generators and their inverses. Further, a group is termed *finitely presented* provided there are finitely many words

$$r_1, r_2, \ldots, r_m$$

(each of which is equal to the identity element e) called relators, so that any word w in the generators $g_1, g_2, \ldots, g_n$ that defines the identity in the group G can be expressed as a product of conjugates of the r_i's and their inverses. It is usual to suppress the trivial relators such as $g_i g_i^{-1} = g_i^{-1} g_i = e$. A *presentation* (see [8]) is written:

$$\langle g_1, g_2, \ldots, g_n \mid r_1, r_2, \ldots, r_m \rangle.$$

The Artin *Braid group* (see [3]), B_N, is generated by elements

$$x_1, x_2, \ldots, x_N$$

and has defining relators given by

$$\begin{aligned} x_i x_j x_i &= x_j x_i x_j, \quad \text{if } |i-j| = 1, \\ x_i x_j &= x_j x_i, \quad \text{if } |j-i| \geq 2. \end{aligned}$$

By adjoining the additional relators

$$x_i^2 = 1, \quad \text{for } i = 1, \ldots, N,$$

to the presentation of B_N we obtain a presentation for the *symmetric group*, S_N, and a natural homomorphism, f, from B_N onto S_N,

$$f : B_N \longrightarrow S_N.$$

The kernel of f is termed the pure braid group, and is denoted P_N. Note that every element in B_N can be expressed as a product of an element in P_N and a permutation in S_N.

In the course of our discussion we will utilize the concepts of *group action* and *semi-direct product* of two groups. A group K **acts** on a group H if there is a homomorphism from K to $Aut(H)$,

$$K \longrightarrow Aut(H), \quad k \mapsto \overline{k}, \quad \overline{k} : H \to H.$$

Given such an action, the semi–direct product of H and K, denoted $H \rtimes K$, is defined on the set of ordered pairs $\{(h,k)|h \in H, k \in K\}$ by

$$(h_1, k_1) * (h_2, k_2) = (h_1\ \overline{k_1}(h_2), k_1 k_2).$$

3 The key extractor algorithm

For $i = 1, \ldots N$, let $y_i = \big(C_i(t_i), \quad (i \ \ i+1)\big)$ where

$$(i \ \ i+1)$$

denotes the transposition (when $i = N$ the transposition is defined to be $(i \ \ 1)$), and

$$C_i(t) = \begin{pmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & t & -t & 1 & & \\ & & & & & & \\ & & & & & 1 & \end{pmatrix}.$$

For $i = 1$ and $i = N$ the matrix $C_i(t)$ is truncated appropriately to give two non-zero entries in the row i. Thus $C_i(t)$ is a matrix with ones on the diagonal, zeros elsewhere, except in the ith row where we have

$$0\,0 \cdots 0 \ \ t \ \ -t \ \ 1 \ \ 0 \cdots 0\,0$$

with $-t$ on the diagonal. We are thus in a position to define the *colored Burau group*, CB_{N+1}, to be the group generated by the set of ordered pairs of the form $\big(C_i(t_i), (i \ \ i+1)\big)$ where multiplication (denoted $*$) of two ordered pairs (M, σ) and (M', σ'), of the above form, in the group CB_{N+1} is given by

$$(M,\ \sigma)\ *\ (M',\ \sigma') = (M \cdot \sigma(M'),\ \sigma\,\sigma'),$$

where $\cdot$ denotes matrix multiplication, $\sigma\,\sigma'$ is the product of the permutations σ and σ' using the functional notation (i.e., evaluation is from right to left), and $\sigma(M')$ is simply the matrix obtained from M' by permuting the variables $t_1, \ldots, t_N$ appearing in the coefficients of M' by the permutation σ.

As an example, we compute in B_5:

$$\begin{aligned}
&\big(C_2(t_2),\ (2\ \ 3)\big) * \big(C_3(t_3),\ (3\ \ 4)\big)\\
&= \big(C_2(t_2)\cdot C_3(t_2),\ (2\ \ 3)(3\ \ 4)\big)\\
&= \left(\begin{pmatrix}1 & 0 & 0 & 0\\ t_2 & -t_2 & 1 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\cdot\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & t_2 & -t_2 & 1\\ 0 & 0 & 0 & 1\end{pmatrix},\quad (2\ \ 3)\ (3\ \ 4)\right)\\
&= \left(\begin{pmatrix}1 & 0 & 0 & 0\\ t_2 & 0 & -t_2 & 1\\ 0 & t_2 & -t_2 & 1\\ 0 & 0 & 0 & 1\end{pmatrix},\quad (2\ \ 3\ \ 4)\right).
\end{aligned}$$

One easily checks that the elements y_i (for $i = 1, \ldots, N$) satisfy the braid relations, and thus we may define a natural homomorphism, E, from the braid group B_{N+1} to the colored Burau group CB_{N+1},

$$E : B_{N+1} \longrightarrow CB_{N+1}, \quad E : x_i \mapsto y_i.$$

It follows that to every element w in the braid group we can associate the element $E(w)$ of the colored Burau group.

In general, a *keyspace* K of order k is a collection of bit strings, termed keys, each of which has length at most k. The elements of the keyspace can be used for cryptographic applications. A *key extractor* on a group G is a function that assigns to each element in the group a unique key in the keyspace.

Assuming now that $N \geq 3$ is fixed, let $p > N$ be a prime number, and let

$$\tau_1, \ldots, \tau_N$$

be a fixed set of distinct and invertible integers $\pmod p$ We define the keyspace to be

$$K_{N+1,p} = \{(M, \sigma) \mid M \text{ is an } N \times N \text{ matrix over } \mathbf{F_p}, \quad \sigma \in S_{N+1}\}.$$

Note that this keyspace is of order $N^2 \cdot \log_2(p) + \log_2\big((N+1)!\big) = O\big(N^2 \log_2(p)\big)$. Observe that there is a natural projection

$$P : CB_{N+1} \longrightarrow K_{N+1,p},$$

which depends on the choices $\tau_1, \ldots, \tau_N$ and p, where

$$P : (M, \sigma) \longrightarrow \Big(M(\tau_1, \ldots, \tau_N) \pmod p,\ \sigma\Big).$$

We now define a key extractor E_p from the Braid group B_{N+1} to $K_{N+1,\,p}$.

Definition *The key extractor $E_p : B_{N+1} \to K_{N,p}$ is defined to be the* composite of the functions E and P*, i.e., given $w \in B_{N+1}$,*

$$\begin{aligned}
E_p(w) &:= P \circ E(w) = P(E(w)) = P\big((M(t_1, \ldots, t_N),\ \sigma)\big)\\
&= \Big(M(\tau_1, \ldots, \tau_N) \pmod p,\ \sigma\Big),
\end{aligned}$$

where reduction $\pmod p$ *means reduction of every entry in the matrix.*

There is a very rapid and efficient linear time algorithm for computing the key extractor E_p. The input for the algorithm is an element of the braid group B_{N+1} of the form $g_1 g_2 \cdots g_\ell$, where each g_i is an Artin generator (i.e., one of $x_1, \ldots, x_N$) or its inverse (i.e., one of $x_1^{-1}, \ldots, x_N^{-1}$) and the output is a pair $(M,\ \sigma) \in K_{N+1,\ p}$.

Key extractor algorithm

Input: A braid word $w = g_1 g_2 \cdots g_\ell$ of length ℓ,
a prime p,
$\{\tau_1, \tau_2, \ldots, \tau_N\}$ distinct invertible integers $\pmod p$.
Initialization: $M =$ $N \times N$ Identity Matrix,
$k = 0$,
$\sigma =$ Identity Permutation.
STEP 1: IF $k = \ell$ then STOP
STEP 2: $k := k + 1$
STEP 3: IF $g_k = x_i$ then GO TO STEP 5
STEP 4: IF $g_k = x_i^{-1}$ then GO TO STEP 8
STEP 5: $M := M \cdot C_i(\tau_{\sigma(i)}) \pmod p$
STEP 6: $\sigma := \sigma \cdot (i\ \ i+1)$
STEP 7: GO TO STEP 1
STEP 8: $M := M \cdot C_i(\tau_{\sigma(i)})^{-1} \pmod p$
STEP 9: $\sigma := \sigma \cdot (i\ \ i+1)$
STEP 10: GO TO STEP 1
Output: $(M,\ \sigma)$.

We now discuss the precise running time of the Key extractor algorithm. We will show it to be linear in the Artin length ℓ of the braid word w. In order to explicitly compute all constants involved in the running time, we assume that the algorithm runs on a processor satisfying the following two hypotheses.

Assumption I We assume that the Key extractor algorithm runs on a processor where it takes:

- $t_{\mathbb{F}_p}$ seconds to add or multiply two elements in the finite field $\mathbb{F}_p$,
- t_{S_N} seconds to multiply two elements in the symmetric group S_N,
- t'_{S_N} seconds to apply a permutation $\sigma \in S_N$ to an integer $1 \le i \le N$,
- t_{B_N} seconds to determine if two Artin generators or their inverses are the same or not.

Assumption II We further assume that all the other operations in the Key Extractor Algorithm, i.e., incrementing an integer, comparing two integers, replacing one integer in a list with another, the GO TO operation, etc., have negligible running time compared with the 4 basic operations in Assumption I. More precisely, we assume that there exists a small $\epsilon > 0$ such that the total running time of the negligible steps 1, 2, 3, 4, 7 in the Key Extractor Algorithm run in time

$$\epsilon \cdot \left(t_{\mathbb{F}_p} + t_{S_N} + t'_{S_N} + t_{B_N}\right).$$

If we carefully analyze the Key Extractor Algorithm, one readily sees that the most time consuming steps are steps 5, 8 where one first applies the element $\sigma \in S_N$ to an integer i, replaces one element of a list by another twice (i.e., apply the permutation), and then multiplies a matrix with coefficients in $\mathbb{F}_p$ with a row vector

with only three entries. This involves three multiplications and two additions in $\mathbb{F}_p$. Since the matrix is an $N \times N$ matrix, the previous computation must be iterated N times. Further, these computations must be again iterated ℓ times for each Artin generator in the word w. This type of analysis immediately leads to the following proposition.

Proposition 1 *Let $w = g_1 g_2 \cdots g_\ell \in B_N$ have length ℓ in Artin generators (i.e., each g_k = an Artin generator or its inverse for $k = 1, 2, \ldots, N$). Under Assumptions I, II, the running time to compute $E_p(w)$ is at most*

$$\Big((5N + \epsilon)t_{\mathbb{F}_p} + (1 + \epsilon)t_{S_N} + (1 + \epsilon)t'_{S_N} + (N + \epsilon)t_{B_N}\Big) \cdot \ell$$

seconds.

Remark For small p and N the basic operations of addition and multiplication in $\mathbb{F}_p$ and comparing Artin generators can be done by table look up.

4 The left multiplication

We now introduce a left multiplication, denoted $\bullet$, which allows us to multiply an element $K_{N+1,p}$ by an element of CB_{N+1}. Recall that given elements $(M,\ \sigma)$ and $(M',\ \sigma')$ in CB_{N+1}, their product is given by

$$(M,\ \sigma)\ *\ (M',\ \sigma') = (M \cdot \sigma(M'),\ \sigma\,\sigma').$$

Now suppose we apply P to (M, σ), to yield the element $\Big(M(\tau_1, \ldots, \tau_N) \pmod p,\ \sigma\Big) \in K_{N+1,p}$. It is possible to *extend* the operation $*$ (multiplication in the colored Burau group) to an operation $\bullet$ defined in the following manner:

$$\begin{aligned}&\Big(M(\tau_1, \ldots, \tau_N) \pmod p,\ \sigma\Big) \bullet (M',\ \sigma') \\ &\quad := \Big(M \cdot \big(\sigma(M')(\tau_1, \ldots, \tau_N) \pmod p\big),\ \sigma\,\sigma'\Big). \qquad (\dagger)\end{aligned}$$

Notice that this extended definition cannot be reversed since we can only apply σ to a matrix whose entries are Laurent polynomials in the variables $t_1, \ldots, t_N$.

Remark The left multiplication $\bullet$ can be efficiently computed using the algorithm for E_p given in section 3.

Proposition 2 *The left multiplication $\bullet$ satisfies the following associative law. For all $(\bar{M},\ \sigma) \in K_{N+1,p}$, and $(M',\ \sigma')$, $(M'',\ \sigma'') \in CB_{N+1}$, we have*

$$\Big((\bar{M},\ \sigma) \bullet (M',\ \sigma')\Big) \bullet (M'',\ \sigma'') = (\bar{M},\ \sigma) \bullet \Big((M',\ \sigma') * (M'',\ \sigma'')\Big).$$

Proof Observe first that, given two Laurent polynomials in N variables, θ, ϕ, we have that $\theta \cdot \phi(\tau_1, \ldots, \tau_N) = \theta(\tau_1, \ldots, \tau_N) \cdot \phi(\tau_1, \ldots, \tau_N)$. As a result, given two matrices A, B whose entries are Laurent polynomials in N variables, then $A \cdot B(\tau_1, \ldots, \tau_N) = A(\tau_1, \ldots, \tau_N) \cdot B(\tau_1, \ldots, \tau_N)$.

We compute:

$$
\begin{aligned}
&\Big((\bar{M},\ \sigma) \bullet (M',\ \sigma')\Big) \bullet (M'',\ \sigma'') \\
&\quad = \Big(\bar{M} \cdot \sigma(M')(\tau_1, \ldots, \tau_N) \pmod p,\ \sigma\,\sigma'\Big) \bullet (M'',\ \sigma'') \\
&\quad = \Big(\bar{M} \cdot \sigma(M')(\tau_1, \ldots, \tau_N) \pmod p \\
&\qquad \times \sigma\,\sigma'(M'')(\tau_1, \ldots, \tau_N) \pmod p,\ \sigma\,\sigma'\,\sigma''\Big) \\
&\quad = \Big(\bar{M} \cdot \big(\sigma(M')(\tau_1, \ldots, \tau_N) \cdot \sigma\,\sigma'(M'')(\tau_1, \ldots, \tau_N)\big) \pmod p,\ \sigma\,\sigma'\,\sigma''\Big) \\
&\quad = \Big(\bar{M} \cdot \big(\sigma(M') \cdot \sigma\,\sigma'(M'')\big)(\tau_1, \ldots, \tau_N) \pmod p,\ \sigma\,\sigma'\,\sigma''\Big) \\
&\quad = \Big(\bar{M} \cdot \sigma\big((M') \cdot \sigma'(M'')\big)(\tau_1, \ldots, \tau_N) \pmod p,\ \sigma\,\sigma'\,\sigma''\Big) \\
&\quad = (\bar{M}, \sigma) \bullet \Big(M' \cdot \sigma'(M''),\ \sigma'\,\sigma''\Big) \\
&\quad = (\bar{M}, \sigma) \bullet \Big((M',\ \sigma') * (M'',\ \sigma'')\Big),
\end{aligned}
$$

which was to be shown. □

5 The linear time key agreement protocol BDH

We are now in a position to present our key agreement protocol. We refer to the subscribers of this protocol as Alice and Bob who will transmit information over a public channel and obtain a shared secret (neither of which can predict in advance) as a consequence. This shared secret cannot be determined in feasible time by an observer of the transmissions. All computations required by Alice and Bob to compute information to be transmitted and to determine the shared secret run in linear time in the bit length of Alice and Bob's secret keys. It is assumed that that we have two commuting subgroups U, V of the braid group B_{N+1}. The structure of these commuting subgroups needs to be hidden. For example, they could be conjugates (by a secret element) of two commuting subgroups.

Public information

- An integer $N \geq 6$, a prime $p > N$
- Invertible elements $\{\tau_1, \ldots, \tau_N\} \pmod p$
- Two commuting subgroups U, V of the braid group B_{N+1}.
- The group CB_{N+1}, the keyspace $K_{N+1,p}$, functions E and E_p and the operation
- defined in † in section 3.
- A fixed element in the key space $K_{N+1,p}$, denoted (M, σ).

Secret keys

- Alice chooses $\alpha \in U$, and Bob chooses $\beta \in V$. Then $\alpha\beta = \beta\alpha$.

Public keys

- Alice computes

$$(M, \sigma) \bullet E(\alpha) \in K_{N+1,p}$$

  by Proposition 2.
- Likewise Bob computes

$$(M, \sigma) \bullet E(\beta) \in K_{N+1,p}.$$

Shared secret

- Alice computes

$$\Big((M, \sigma) \bullet E(\beta)\Big) \bullet E(\alpha) = (M, \sigma) \bullet \Big(E(\beta) * E(\alpha)\Big)$$

  using †.
- Bob similarly computes

$$\Big((M, \sigma) \bullet E(\alpha)\Big) \bullet E(\beta) = (M, \sigma) \bullet \Big(E(\alpha) * E(\beta)\Big).$$

- Since $\alpha\beta = \beta\alpha$ and E is a homomorphism, $E(\alpha) * E(\beta) = E(\beta) * E(\alpha)$ and we have obtained a shared key:

$$\Big((M, \sigma) \bullet E(\beta)\Big) * E(\alpha) = \Big((M, \sigma) \bullet E(\alpha)\Big) * E(\beta).$$

Remark It is important that the public key $(M, \sigma) \in K_{N+1,p}$ and the generators of the hidden commuting subgroups U, V, be the images of braid words $w \in B_{N+1}$ (under the map E_p) where w is sufficiently long and involves all the Artin generators of B_{N+1}. These issues are discussed at length in [4] and scrutinized in [5] with regard to another type of Diffie–Hellman type protocol on braid groups, and we do not reproduce them here.

The security of the BDH key agreement protocol is tied to the difficulty of inverting the key extractor E_p which maps braid words to pairs $(M, \sigma) \in K_{N+1,p}$. We believe that this is a one–way function for the following reasons: (1) E_p is not a group homomorphism, and (2) because the representation of the braid group B_{N+1} into the colored Burau group induces a representation of the braid group with rank $> N \cdot (N + 1)!$. This rank is so large that it appears quite infeasible to attack the system by methods of linear algebra. We also feel that the combination of braid group multiplication and arithmetic reduction $(\mathrm{mod}\ p)$ provides a union of distinctly different mathematical operations which may make this protocol particularly robust. There will be a class of weak keys, however, arising from elements in $w \in B_{N+1}$ whose associated permutations are close to the identity. For these elements w, the application of E_p will not produce enough permutations of $\tau_1, \tau_2, \ldots, \tau_N$ to guarantee the security of the system. This needs to be studied further.

There are numerous applications of BDH. They include message authentication, digital signatures, and zero knowledge proof systems. We will discuss these in forthcoming work.

Acknowledgements The authors would like to thank Benji Fisher for many helpful discussions.

References

1. Anshel, I., Anshel, M., Goldfeld, D.: An algebraic method for public-key cryptography. Math Res Lett **6**, 1–5 (1999)
2. Anshel, I., Anshel, M., Fisher, B., Goldfeld, D.: New key agreement protocols in Braid group cryptography. In: Naccache, D. (ed.) Topics in Cryptography–CTRSA 2001, (Lecture Notes in Computer Science, vol 2020). Berlin Heidelberg New York: Springer (2001)
3. Birman, J.: Braids, links and mapping class groups, Annals of Mathematical Studies, Study 82. New Jersey: Princeton University Press (1974)
4. Ko, K.H., Lee, S.J., Chean, J.H., Han, J.W., Kang, J.S., Park, C.: New public-key cryptosystem using braid groups. In: Bellare, M. (ed.) Advances in Cryptology–Crypto 2000, (Lecture Notes in Computer Science, vol 1880). Berlin Heidelberg New York: Springer (2000)
5. Lee, E., Lee, S.J., Hahn, S.G.: Pseudorandomness from Braid groups. In: Advances in Cryptology–Crypto 2001 (Lecture Notes in Computer Science) Berlin Heidelberg New York: Springer (2001)
6. Morton, H.R.: The multivariable alexander polynomial for a closed Braid. Contemp Math AMS, **233**, 167–172 (1999)
7. Menzes, A.J., van Oorschot, P.C., Vanstone, S.A.: The handbook of applied cryptography. Boca Raton: CRC Press (1997)
8. Robinson, D.: A course in the theory of groups: graduate texts in mathematics 80. Berlin Heidelberg New York: Springer (1996)
9. Sidel'nikov, V.M., Cherepenev, M.A., Yashichenko, V.V.: Systems of open distribution of keys on the basis of noncommutative semigroups. Russ Acad Sci Dolk Math **48**(2), 384–386 (1994)