# ブレイド群を利用した公開鍵暗号系の実装と性能評価実験

**Experimental Performance Evaluations of Public-key Cryptosystem Using Braid Groups**

http://www.tani.cs.chs.nihon-u.ac.jp/g-2005/yuko/

**谷 研究室　田中 裕子**
**Yuko TANAKA**

**概要**

2000 年に，ブレイド (組み紐) 群を利用した公開鍵暗号系が提案された．本研究では，その実装と性能評価実験を行った．

## 1　はじめに

ネットワーク化が進んだ今日，情報の隠蔽を目的とした暗号技術は，数々のセキュリティ技術の中核を成すものの 1 つとして位置付けられている．暗号とは，当事者以外には意味がわからないように，鍵を使い決められた手順に従って変換した特殊な記号や文字，または方式のことをいう．暗号系には，受信側と送信側が共通の鍵を使う秘密鍵暗号系と異なる鍵を使う公開鍵暗号系があるが，今回扱うのは後者である．

現在最も広く使われている公開鍵暗号系は，素因数分解の難しさを利用した RSA 暗号系である．この暗号系では，2 つの大きな素数の積を鍵としている．安全性を高めるためにより長い鍵を使うことが求められているが，携帯電話や IC カードのように限られた環境で大きな整数の演算を行うことは困難であり，RSA 暗号系はそのような環境に適していない．そこで，楕円曲線暗号などの新しい公開鍵暗号系が研究されている．

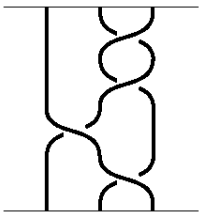
ブレイドとは組み紐のことである．ブレイド群に関する計算問題の計算量解析は応用も含めて活発に研究されており，2000 年に K.H.Ko, S.J.Lee, J.H.Cheon, J.W.Han, J.Kang, C.Park[1] によってブレイド群における共役問題の難しさを利用した公開鍵暗号系が提案された．

本研究では，ブレイド群を利用した公開鍵暗号系を実装し，性能評価実験を行う．2，3 章でブレイド群について解説し，4，5 章で，ブレイド群における困難な問題の紹介と，それを利用した公開鍵暗号系のしくみを解説する．6 章では，実装した公開鍵暗号系の性能評価実験の結果を示す．

## 2　ブレイド群に関する定義

$n-$ ブレイドとは平行な 2 つの面に接続している $n$ 本の紐であり，上の面から出ている紐をたどると常に下へ向かい必ず下の面につくものをいう．

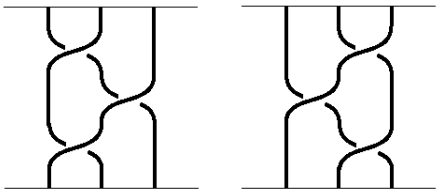
$n-$ ブレイド群のことを $B_n$ と表し，$n$ を braid index という．



3-ブレイドの例

### ブレイドの同値

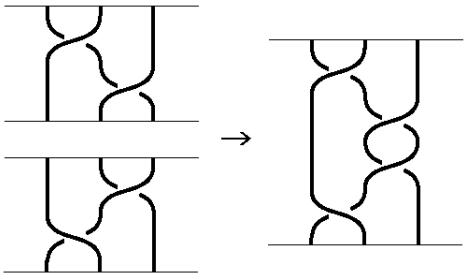
一方のブレイドを，紐が面に接続している点を固定したまま動かしてもう一方のブレイドと同じものにできるとき，2 つのブレイドは同値であるという．



同値なブレイドの例

### ブレイドの積

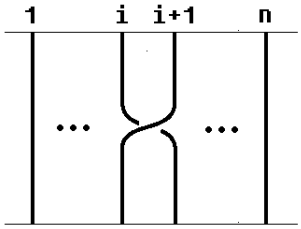
2 つのブレイド $a, b$ の積 $ab$ は，$a$ を $b$ の上に置くことで得られる．



2 つのブレイドの積

**生成元**

生成元 $\sigma_i$ とは，$i$ 番目と $i+1$ 番目の紐のみが交差しているブレイドで，$i$ 番目の紐が $i+1$ 番目の紐の下になるものをいう．生成元 ${\sigma_i}^{-1}$ とは，$\sigma_i$ の紐の重なり方を上下反対にしたものをいう．




生成元 $\sigma_i$

ブレイドは $\sigma_i, {\sigma_i}^{-1}$ の積の形で表される．このとき，ブレイドの同値の定義より次がいえる．

$$\sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j \qquad (if \quad |\, i-j \,| = 1)$$

$$\sigma_i\sigma_j = \sigma_j\sigma_i \qquad (if \quad |\, i-j \,| \geq 2)$$

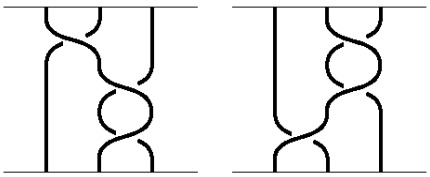
また，ブレイドを構成する生成元の個数をブレイドの長さという．

**単位元**

どの紐も交差していないブレイドのこと．$e$ で表す．

**逆元**

ブレイド $a$ の逆元 $a^{-1}$ は，$a$ を紐が接続している面で折り返したものである．もとのブレイドとその逆元との積は単位元になる．



もとのブレイドと逆元

**正ブレイド**

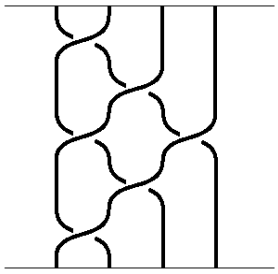
${\sigma_i}^{-1}$ を含まないもの．正ブレイドの集合を ${B_n}^+$ と表す．

**順列ブレイド**

正ブレイドの中で，それぞれの紐の交差が高々1 回であるもの．すべての順列ブレイドの集合を $\tilde{\Sigma}_n$ と表す．

**基本ブレイド**

順列ブレイドの中で，すべての紐がちょうど 1 回ずつ交差しているもの．$\Delta$ で表す．



基本ブレイド $\Delta_4$

基本ブレイドから交差を 1 つ以上取り除いたものが順列ブレイドになっている．

## 3 標準形

ブレイドは生成元 $\sigma_i, \sigma^{-1}$ の積で表されるが，それは一意に決まらない．そこで，left-canonical form という標準形を用いる．

**starting set と finishset**

正ブレイド $P$ に対して，starting set $S(P)$ と finishing set $F(P)$ が以下のように定義される．

$$S(P) = \{i \mid \text{ある } P' \in {B_n}^+ \text{に対して}, P = \sigma_i P'\}$$

$$F(P) = \{i \mid \text{ある } Q' \in {B_n}^+ \text{に対して}, P = Q'\sigma_i\}$$

**left-weighted**

任意の正ブレイド $P$ に対して，以下のような分割が一意に存在する．

$$P = A_1P_1, A_1 \in \tilde{\Sigma}_n, P_1 \in {B_n}^+, F(A_1) \supseteq S(P_1).$$

このような分割を left-weighted であるという．

left-canonical form

任意の $W \in B_n$ に対して，以下のような表現が一意に存在する．

$$W = \Delta^u A_1 A_2 \cdots A_p, u \in \mathbb{Z}, A_i \in \tilde{\Sigma}_n \backslash \{e, \Delta\}.$$

$1 \leq i \leq p-1$ に対して $A_iA_{i+1}$ は left-weighted になっている．
$p$ を canonical length といい，len$(W)$ と表す．

**left-canonical form 構成アルゴリズム**

基本ブレイドは以下のような性質を持つ．

**(a)** $1 \leq i \leq n-1$ に対して，$\Delta = \sigma_i A_i = B_i \sigma_i$ となるような $A_i, B_i \in \tilde{\Sigma}_n$ が存在する．

**(b)** $1 \leq i \leq n-1$ に対して，$\sigma_i\Delta = \Delta\sigma_{n-i}$ が成り立つ．

生成元の積の形で表されたブレイド $W$ が与えられたとき，(a) より，${\sigma_i}^{-1} = \Delta^{-1}B_i$ となり，これを使って ${\sigma_i}^{-1}$ を置き換えることができる．さらに，(b) を使って $\Delta$ を左に集めることができる．
以上より，

$$W = \Delta^u P, u \in \mathbb{Z}, P \in {B_n}^+$$

を得る．

任意の正ブレイド $P$ に対して，$P = A_1P_1, A_1 \in \tilde{\Sigma}_n, P_1 \in {B_n}^+$ のような left-weighted となる分割 が存在する．この分割を $P = A_1P_1, P_1 = A_2P_2 \cdots$ のように繰り返すことで，$P = A_1A_2 \cdots A_p$ を得る．

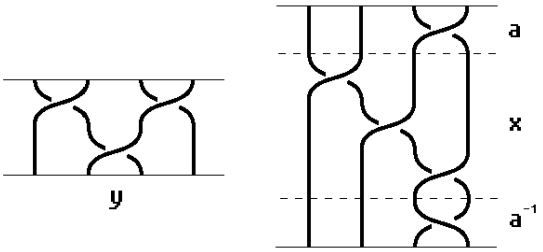
以上より，

$$W = \Delta^u A_1A_2 \cdots A_p, u \in \mathbb{Z}, A_i \in \tilde{\Sigma}_n \backslash \{e, \Delta\}.$$

を得る．

# 4 ブレイド群における計算問題

ブレイド $x$ と $y$ が共役であるとは，$y = axa^{-1}$ となるブレイド $a$ が存在するときをいう．




共役なブレイドの例

共役に関して，困難であると予想されている計算問題に以下のようなものがある．$B_m$ を $m < n$ に対して $\sigma_1, \cdots, \sigma_{m-1}$ で表される $B_n$ の部分群とする．

**共役決定問題**

**入力：** $(x, y) \in B_n \times B_n$
**出力：** $x$ と $y$ が共役であるか？

**共役探索問題**

**入力：** 互いに共役であるような $(x, y) \in B_n \times B_n$
**出力：** $y = axa^{-1}$ となるような $a \in B_n$

**一般共役探索問題**

**入力：** $b \in B_m, m \leq n$ に対して $y = bxb^{-1}$ となるような $(x, y) \in B_n \times B_n$
**出力：** $y = axa^{-1}$ となるような $a \in B_m$

**共役分解問題**

**入力：** $b \in B_m, m \leq n$ に対して $y = bxb^{-1}$ となるような $(x, y) \in B_n \times B_n$
**出力：** $y = a'xa''$ となるような $a', a'' \in B_m$

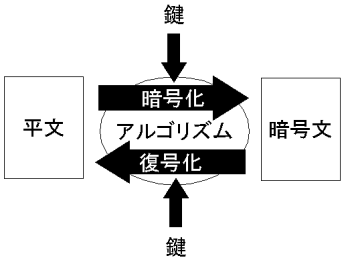
# 5 ブレイド群を利用した公開鍵暗号系

暗号とは，当事者以外には意味がわからないように，決められた手順に従って変換された特殊な記号や文字，またはその方式のことをいう．

送信者が送りたい元の文章を「平文」，暗号文に変換することを「暗号化」，受信者が暗号文を元の文章に戻すことを「復号化」といい，暗号化と復号化には「鍵」を用いる．

**秘密鍵暗号系**

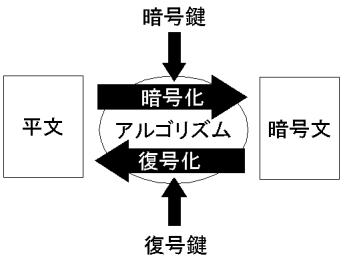
秘密鍵暗号系とは，暗号化と復号化に同じ鍵を用いる暗号方式である．



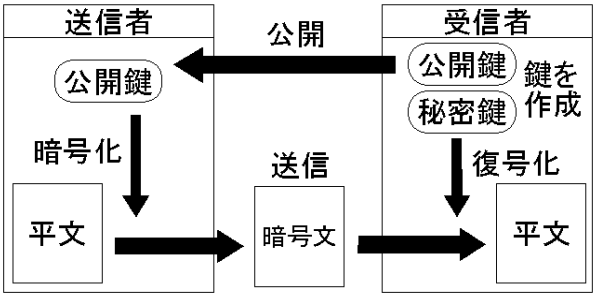

この方式では，事前に何らかの方法を用いて，鍵を第三者に知られることなく受信者側に渡す必要がある．また、多数の相手に対して同様の暗号化を行う場合、相手の数だけ鍵を用意する必要がある．その反面，暗号化と復号化に同じ鍵を使い，アルゴリズムも単純であることから，高速な処理が可能になっている．

**公開鍵暗号系**

公開鍵暗号系とは，暗号化と復号化に異なる鍵を用いる暗号方式である．




受信者は「秘密鍵」と「公開鍵」をペアで作成し,「公開鍵」を公開して「秘密鍵」は自分で保管しておく．送信者は受信者の公開鍵を使って暗号化を行い，暗号文を送信する．暗号文を受け取った受信者は，保管しておいた秘密鍵を使って復号化する．




この暗号系は,

**(a)** 公開鍵から秘密鍵を導き出すことはできない.

**(b)** 一方の鍵で暗号化したものは，そのペアとなるもう一方の鍵でしか復号化できない.

という条件のもとに成り立っている．(a) の条件には,「一方向関数」という，引数から値を求めることは易しいがその逆が難しいような関数が利用されている.

この方式では，多数の相手とのやり取りを行う場合でも，自分の公開鍵と秘密鍵のペアを 1 つだけ持っていればよく，鍵の管理が簡単になる．しかし、暗号化と復号化のアルゴリズムが複雑であるため，処理に時間がかかる.

## ブレイド群における一方向関数

$B_{l+r}$ の部分群 $LB_l$ と $RB_r$ 考える．$LB_l$ は $B_{l+r}$ の左側の $l$ 本の紐からなり，$RB_r$ は右側の $r$ 本の紐からなる．つまり，$LB_l$ は生成元 $\sigma_1 \cdots \sigma_{l-1}$，$RB_r$ は生成元 $\sigma_{l+1} \cdots \sigma_{l+r-1}$ から構成される．このとき，任意の $a \in LB_l$ と $b \in RB_r$ は可換である.

以下のような一方向関数が存在する.

$$f : LB_l \times B_{l+r} \to B_{l+r} \times B_{l+r}, f(a,x) = (axa^{-1}, x).$$

$(a,\ x)$ が与えられたとき $axa^{-1}$ を計算するのは易しいが，$(axa^{-1}, x)$ から $a$ を計算するのには指数時間を要する.

公開鍵暗号系の安全性は，以下の問題の難しさに基づく.

Base Problem

**入力：**
秘密である $a \in LB_l, b \in RB_r$ に対して，$y_1 = axa^{-1}, y_2 = bxb^{-1}$ となるような $B_{l+r}$ の元 $(x, y_1, y_2)$

**出力：**
$by_1b^{-1} (= ay_2a^{-1} = abxa^{-1}b^{-1})$

## ブレイド群を利用した公開鍵暗号系

以下，すべてのブレイドは left-canonical form で表されているものとする.

**鍵の生成**

**(a)** 十分に複雑なブレイド $x \in B_{l+r}$ を選ぶ.

**(b)** $a \in LB_l$ を選ぶ.

**(c)** $y = axa^{-1}$ を計算し，$(x, y)$ を公開鍵，$a$ を秘密鍵とする.

$x$ が十分に複雑であるとは，$x_1 \in LB_l, x_2 \in RB_r, LB_l$ と $RB_r$ が可換であるような $z \in B_{l+r}$ に対して，$x$ が $x_1x_2z$ に分解できないときをいう.

$x$ と $y = axa^{-1}$ は公開されるが，$f(a,x) = (axa^{-1}, x)$ は一方向関数であるため秘密鍵 $a$ が知られる可能性は低い.

**暗号化**

メッセージ $m \in \{0,1\}^k$ と公開鍵 $(x, y)$ が与えられる．また，$H : B_{l+r} \to \{0,1\}^k$ をブレイド群からメッセージ空間へのハッシュ関数とする．ハッシュ関数とは，与えられた原文を固定長のデータに要約するための関数である．不可逆な一方向関数を含むため，ハッシュ値から原文を再現することはできず，また同じハッシュ値を持つ異なるデータを作成することは極めて困難である．代表的なものには SHA-1 と MD5 があり，今回の実装では MD5[6] を用いた.

**(a)** $b \in RB_r$ をランダムに選ぶ.

**(b)** $c = bxb^{-1}$ と $d = H(byb^{-1}) \oplus m$ を計算し，$(c, d)$ を暗号文とする.

$c = bxb^{-1}$ と $d = H(byb^{-1}) \oplus m$ は暗号文として送信されるため，データが盗まれる危険性もある．しかし，$c = bxb^{-1}$ から $b$ を求めることは難しい（一方向関数）．また，$d = H(byb^{-1}) \oplus m$ より $m = H(byb^{-1}) \oplus d$ であるが，これは $byb^{-1}$ を知らなければ（すなわち $b$ を知らなければ）求めることはできない.

**復号化**

暗号文 $(c, d)$ と秘密鍵 $a$ から，$m = H(aca^{-1}) \oplus d$ を計算する.

$c = bxb^{-1}$，$y = axa^{-1}$，$a \in LB_l$ と $b \in RB_r$ が可換であることを使うと，

$$\begin{aligned} aca^{-1} &= abxb^{-1}a^{-1} \\ &= baxa^{-1}b^{-1} \\ &= byb^{-1} \end{aligned}$$

さらに $d = H(byb^{-1}) \oplus m$ を使うと，

$$\begin{aligned} & H(aca^{-1}) \oplus d \\ = \ & H(byb^{-1}) \oplus d \\ = \ & H(byb^{-1}) \oplus H(byb^{-1}) \oplus m \\ = \ & m \end{aligned}$$

となり，正しく復号化が行われている．

# 6 ブレイド群を利用した公開鍵暗号系の性能評価実験

5 章で解説した公開鍵暗号系を C++で実装し，性能評価実験を行った．

## 実行環境

CPU : Pentium 4(2.7GHz)
OS : Fedora Core 2
搭載メモリ : 1GB
コンパイラ : g++ 3.3.3

## 性能評価実験

### 先行実験

2000 年に Priit Karu, Jonne Loikkanen[2] によって行われた実験では，紐の本数 5 本〜100 本に対して，鍵の生成・暗号化・復号化それぞれにかかる時間を測定し結果を比較している．また，紐の本数が 48 本のとき RSA 暗号系における鍵長 1024bits のときと同程度の安全性があるとして，実行速度の比較を行っている．

### 実験方法

“Public-key Cryptosystem Using Braid Groups” という平文に対して，鍵の生成・暗号化・復号化にかかる合計時間を測定した．鍵をランダムに生成して 20 回実行し，その平均を求めた．

### 実験結果

《紐の本数とブレイドの長さによる比較》
紐の本数 4 本〜5 本に対して，ブレイドの長さを変えたときの処理速度を測定した結果を以下に示す．

| 長さ/本数 | 4 | 5 |
|---|---|---|
| 50 | 0.03 | 0.15 |
| 100 | 0.06 | 0.70 |
| 200 | 0.09 | 2.23 |
| 300 | 0.23 | 5.30 |
| 400 | 0.45 | 8.52 |
| 500 | 0.64 | 14.80 |

(単位：秒)

# 参考文献

[1]Ki Hyong Ko, Sang Jin Lee, Jung Hee Cheon, Jae Woo Han, Ju-sung Kang and Choonsik Park, *New publiv-key Cryptosystem Using Braid Groups*, Advances in Cryptography - CRYPTO 2000, Lecture Notes in Computer Science 1880, Springer-Verlag,166-183,2000.
[2]Priit Karu, Jonne Loikkanen, *Practical comparison of Fast Public-key Cryptosystems*, 2000.
[3]Dennis Hofheinz and Rainer Steinwandt, *A Practical Attack on Some Braid Group Based Cryptographic Primitives*, PKC,2003.
[4] 情報セキュリティの科学ーマジック・プロトコルへの招待ー　太田和夫・黒澤馨・渡辺治　講談社
[5]RSA ー暗号技術の基礎から C++による実装までー　橋本晋之介　ソフトバンクパブリッシング
[6]Ronald L.Rivest : HomePage
http://theory.lcs.mit.edu/ rivest/
[7] 情報処理推進機構 : CRYPTREC
http://www.ipa.go.jp/security/enc/CRYPTREC/