Cryptanalysis of a Pseudorandom Generator Based on Braid Groups

Rosario Gennaro[1] and Daniele Micciancio[2,⋆]

[1] IBM T.J.Watson Research Center, New York, USA,
rosario@watson.ibm.com
[2] University of California, San Diego. La Jolla, California, USA,
daniele@cs.ucsd.edu

Abstract. We show that the decisional version of the Ko-Lee assumption for braid groups put forward by Lee, Lee and Hahn at Crypto 2001 is false, by giving an efficient algorithm that solves (with high probability) the corresponding decisional problem. Our attack immediately applies to the pseudo-random generator and synthesizer proposed by the same authors based on the decisional Ko-Lee assumption, and shows that neither of them is cryptographically secure.

1 Introduction

The search for computationally hard problems to be used as a basis of secure encryption functions is a central problem in cryptography. Recently, braid groups have attracted the attention of many cryptographers as a potential source of computational hardness and many cryptographic protocols have been suggested based on braid groups [1, 10, 11]. Computational assumptions and cryptographic protocols based on braid groups often resemble similar constructions based on number theoretic groups.

In this paper we point out some fundamental differences between braid groups and number theoretic ones, and show that protocols based on braid groups that are naively designed by analogy with number theoretic groups, can be easily broken. In particular, we show that the decisional version of the Ko-Lee problem put forward by Lee, Lee and Hahn in [11], can be efficiently solved and cannot be used as a basis for the design of secure cryptographic functions. Our attack is extremely efficient: for the values of the security parameters suggested in [11] it only requires a handful of arithmetic operations. Moreover, the attack is asymptotically fast (i.e., polynomial in the input size) and cannot be avoided simply by increasing the value of the security parameter.

Our attack immediately invalidates the security proof of the pseudo-random generator suggested in [11], based on the conjectured (and now disproved) hardness of the decisional Ko-Lee problem. In fact, the scope of our attack extends beyond simply invalidating the computational assumption. Essentially the same techniques used to show that the computational assumption is false, can be used

⋆ Supported in part by NSF Career Award CCR-0093029

to break the "pseudo-random" braid generator proposed in [11] and efficiently distinguish the braids produced by the generator from truly random braids. Our attack applies to the pseudo-random braid generator, as well as the pseudo-random braid synthesizer proposed in [11].

We point out that our attack does not seem to apply to the *computational* version of the Ko-Lee problem, as used in [10] to design a practical encryption scheme in the random oracle model [2]. Without "random oracles", our attack could have been used to extract partial information about the message in the encryption scheme proposed in [10], thereby breaking semantic security. (I.e., the standard notion of security for encryption schemes, see [9].) The use of "random oracles" in [10] protects their encryption scheme from the kind of attacks described in this note, and seems to require a solution to the *computational* version of the Ko-Lee assumption in order to successfully attack the system. A similar "fix" (i.e., applying a hash function modeled as a random oracle to the output of the generator) would clearly apply to the generator of [11] as well, but it would also make the problem studied in [11] completely trivial: in the random oracle model, an extremely efficient pseudo-random generator can be immediately built applying the random oracle directly to the seed, without any computational assumption about braids whatsoever.

It is important to observe that in order to make the result of a Diffie-Hellman or Ko-Lee key exchange look pseudo-random, it is not enough to apply a universal hash function [6]. Indeed universal hash functions produce an almost-random output when starting from an input that has enough entropy to begin with, and this seems to require the *decisional* assumption which we prove to be false for the case of braid groups. In other words, to prove semantic security based only on the *computational* Diffie-Hellman or Ko-Lee assumption the full power of the random oracle model seems to be required.

Our attack does not imply that braid groups cannot be used for cryptographic purposes, and we believe that the use of braid groups as an alternative to number theory in cryptographic applications is an attractive and promising research area. However, extreme care must be used to avoid pitfalls as those exploited in our attack, and cryptographic protocols based on the hardness of computational problems on braid groups must be carefully validated by accurate proofs of security.

Most of the proposed cryptographic schemes based on braid groups, rely on the hardness of the conjugacy problem. Besides [10, 11] other schemes include [1] for example. We point out that our attack to the Decisional Ko-Lee assumption, also reveals a more general fact that applies to all the schemes above. The one-way function constructed from the conjugacy problem reveals partial information about its input (it is indeed this partial information that allows us to build our attack). This leakage of information can be avoided by a careful choice of the design parameters (i.e., by working in an appropriate subgroup), which should be incorporated in all schemes based on the braid conjugacy problem. This is however not enough to save the Decisional Ko-Lee Assumption which we show to be insecure for essentially *any* choice of the design parameters.

As a final remark, we point out that the proof of security of the "hard-core" predicate for the conjugacy one-way function described in [11] contains a fundamental flaw (see Sect. 5). Although we do not know of any cryptanalytic attack at the time of this writing, the security of that predicate by no means can be considered proven based on the conjectured intractability of standard problems on braid groups. Until a satisfactory proof of security is found, the only way to get hard-core predicates for the braid conjugacy function is to invoke general results like the Goldreich-Levin predicate [8].

2 Preliminaries

The n-braid group B_n is an infinite non-commutative group defined by the following group presentation:

$$B_n = \left\langle \sigma_1, \ldots, \sigma_{n-1} : \begin{array}{ll} \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j & \text{if } |i-j| = 1, \text{ and} \\ \sigma_i\sigma_j = \sigma_j\sigma_i & \text{if } |i-j| \geq 2 \end{array} \right\rangle$$

The integer n is called the *braid index*. Elements of B_n are called n-braids. We can give braids the following geometric interpretation. Think of n strands hanging contiguously from the ceiling. Each generator σ_i represents the process of swapping the i^{th} strand with the next one (with the i^{th} strand going under the $(i+1)^{th}$ one).

The multiplication of two braids a, b is then defined as their concatenation (i.e. geometrically putting a on top of b). The identity e is the braid of n straight strands. The inverse of a is the reflection of a across an horizontal line at the bottom.

There is an efficient algorithm to put braids in a normal form. We are not going to use this fact, except for allowing us to define uniquely the *length* of a braid a, which we denote by $|a|$.

Permutations vs Braids. It is interesting to note that there is a natural projection from braids to permutations. Geometrically, given a braid a we can define a permutation π_a induced by a as follows. For the i^{th} strand, consider the position j in which this strand ends at the bottom of a, and define $\pi_a(i) = j$. We will make extensive use of this projection from braids to permutations.

Conjugacy. Given a braid a, we say that another braid y is a *conjugate* of a if there exists a braid x such that $y = xax^{-1}$. It is assumed that solving the conjugacy problem (i.e., retrieving x from a and y) is computationally hard in B_n.

2.1 Key Exchange Based on Braids

At Crypto 2000, [10] suggested a new key exchange protocol based on the conjugacy problem on braid groups. We briefly recall their ideas here.

Notation: For the rest of the paper, we assume that n is even, and let $\ell = \frac{n}{2}$. The key exchange protocol in [10] concerns the braid groups B_n with even n, and the two subgroups $B_L, B_R \subset B_n$ defined as follows:

- B_L is the subgroup generated by $\sigma_1, \ldots, \sigma_{\ell-1}$, i.e., the subgroup of braids that only act on the left strings $1, \ldots, \ell$.
- B_R is the subgroup generated by $\sigma_{\ell+1}, \ldots, \sigma_{n-1}$, i.e., the subgroup of braids that only act on the right strings $\ell + 1, \ldots, n$.

The relevant property is that elements of B_L and B_R commute, i.e., for any $x_l \in B_L$ and $x_r \in B_R$ we have $x_l x_r = x_r x_l$.

This property was used in [10] to construct the following key exchange protocol. Let a be a public braid, $a \in_R B_n$, with $|a| = k$ (where k is a security parameter). Alice has a public key y, where $y = x_l a x_l^{-1}$ for $x_l \in_R B_L$, $|x_l| = k$. Similarly Bob has a public key z, where $z = x_r a x_r^{-1}$ for $x_r \in_R B_R$, $|x_r| = k$. The shared key is $s = x_l x_r a x_r^{-1} x_l^{-1}$. Indeed, since x_l and x_r commute, s can be computed given one of the public keys and the other secret key:

$$s = x_l x_r a x_r^{-1} x_l^{-1} = x_r x_l a x_l^{-1} x_r^{-1} = x_r y x_r^{-1} = x_l z x_l^{-1}$$

Notice that it is necessary to assume that conjugacy is hard in B_n for this to be a secure protocol (otherwise an attacker could compute x or w on its own). But the security of the protocol actually relies on a stronger assumption: i.e. that s must be hard to compute given y, z. We call this the *Computational Ko-Lee Assumption.*

Moreover this assumption alone is not sufficient to achieve provable semantic security for the resulting cryptosystem, since if s is used as a shared key it should be random or pseudo-random. In [10] this problem is resolved by setting $key = H(s)$ where H is a suitable hash function and security can be proven in the random oracle model.

There is an analogy with the Diffie-Hellman key exchange, where Alice has a public key $y = a^{x_l}$ and Bob has public key $z = a^{x_r}$, and they share the key $key = H(s)$ where $s = a^{x_l x_r}$. In order to prove security (in the random oracle model) the (Computational) Diffie-Hellman assumption is required, not just the hardness of computing discrete logarithms. Alternatively, one can assume that the Diffie-Hellman problem is hard even in its decisional version: given a, y, z, w it is hard to tell (with probability substantially better than 1/2 if $w = s$ or w is a randomly chosen element in the group generated by a. This Decisional Diffie-Hellman (DDH) assumption has been widely used in cryptography and it is now a relatively established assumption. Under the DDH assumption, no random oracle is needed because s is already a pseudo-random group element. (See [3] for further discussion of the DDH problem.)

3 Main Result

In this section we prove the main result of the paper, namely that the decisional version of the Ko-Lee assumption is false. This assumption was suggested in [11] as the basis for some constructions of pseudo-random generators and synthesizers based on braid groups.

The assumption can be considered the equivalent of the Decisional Diffie-Hellman Assumption for the key exchange scheme based on braid groups proposed in [10].

Informally the **Decisional Ko-Lee assumption** says the following.

Given the following public information: a, y, z where
- $a \in_R B_n$, with $|a| = k$,
- $y = x_l a x_l^{-1}$ for $x_l \in_R B_L$, $|x_l| = k$
- $z = x_r a x_r^{-1}$ for $x_r \in_R B_R$, $|x_r| = k$

it is hard to distinguish the "shared key" $s = x_l x_r a x_r^{-1} x_l^{-1}$ from a random conjugate of a of the form waw^{-1}.

The Ko-Lee Decisional Assumption, goes a step further with respect to its computational counterpart. It claims that, not only s is hard to compute, but it's even hard to *distinguish* from a random conjugate of a. In other words, under this assumption the hash function H would not be necessary to prove security since s could be used directly as a random shared key.

More formally the decisional version of the Ko-Lee Assumption can be stated as follows.

Assumption 1. *For every probabilistic polynomial-time Turing machine $\mathcal{D}$, for every polynomial P, for all sufficiently large k*

$$\begin{aligned} &|\Pr[\mathcal{D}(a, x_l a x_l^{-1}, x_r a x_r^{-1}, x_l x_r a x_r^{-1} x_l^{-1}) = 1] \\ &\quad - \Pr[\mathcal{D}(a, x_l a x_l^{-1}, x_r a x_r^{-1}, waw^{-1}) = 1]| \le \frac{1}{P(k)} \end{aligned}$$

where the probability is taken over the coin tosses of $\mathcal{D}$ and the following random choices: $a, w \in_R B_n$, $x_l \in_R B_L$, $x_r \in_R B_R$, all braids of length k.

We show that Assumption 1 is false, by exhibiting partial information about s that can be computed from the public information a, y, z. In fact we describe a sequence of attacks. Each attack exploits some specific partial information, and can be avoided by suitably restricting the way a and w are chosen. But then, another attack applies. The sequence of attacks leads to a complete break of the system, showing that for any reasonable choice of probability distribution on the key space a, x_l, x_r, w there is an efficient algorithm that distinguishes $(x_l x_r)a(x_l x_r)^{-1}$ from a random conjugate waw^{-1}.

3.1 The Permutation Attack

The main idea behind the first attack is to focus on the permutation induced by each braid, over the set of strings $\{1, 2, \ldots, n\}$. We are going to use the fact that the braid x_l only acts on the strings on the left, while the braid x_r acts only on the strings on the right.

Our attack shows that the permutation induced by s must satisfy some constraints, which are very unlikely to be satisfied by the permutation induced by a random conjugate. Our attack is actually stronger than needed since it works *for almost any* braid a and not just for a randomly selected one. We start with a basic fact about the permutation induced by a conjugate of a braid.

Fact 2. *Let $a, x, y \in B_n$ be such that $y = xax^{-1}$. Then π_a and π_y have the same cycle structure. Moreover if A is a cycle in π_a, then the corresponding cycle in π_y is $\pi_x^{-1}(A)$.*

The above fact gives some information about the permutation π_x which is hidden in the conjugate y. We are going to use this information to distinguish s from a random conjugate. The only case in which the above fact does not reveal anything about π_x is when π_a is the identity permutation. We say that a is a *pure* braid if π_a is the identity permutation. We are going to show that if a is *not* a pure braid than Assumption 1 is false.

Remark: Before we show that Assumption 1 is false, let us point out that Fact 2 holds regardless of the way we choose the string x used for the conjugation. Thus, we are basically pointing out that the conjugacy problem, although supposed to be hard to solve, does reveal some partial information about x, unless the braid a is chosen to be pure. Since the conjugacy problem is used in all the braid group cryptosystems, this basic fact applies to all of them. To avoid this leakage of partial information, it seems that the all the schemes above should select the braid a as a pure braid.

We now resume the proof that Assumption 1 is false when a is not a pure braid. We distinguish two cases. In the first case, the permutation π_a maps some elements of $\{\ell+1, \ell+2, \ldots, n\}$ to the set $\{1, 2, \ldots, \ell\}$, i.e., from the right half to the left half. (Notice that this is equivalent to the symmetric condition, i.e., the permutation maps some string from the left half to the right half.) The second case covers all the other non-trivial permutations, i.e., the ones that have cycles of at least size 2 and they are all contained in one of the two halves.

Case 1. Let i be an integer in $\{\ell+1, \ell+2, \ldots, n\}$ such that $j = \pi_a(i) \in \{1, 2, \ldots, \ell\}$.

We now find the element $j' = \pi_y(i) = \pi_{x_l}^{-1}(\pi_a(\pi_{x_l}(i)))$. Since π_{x_l} only acts on elements on the left, we have that $\pi_{x_l}(i) = i$. Thus $j' = \pi_{x_l}^{-1}(\pi_a(i)) = \pi_{x_l}^{-1}(j)$. In other words we found the mapping of the point j under $\pi_{x_l}^{-1}$.

A similar reasoning tells us that if we take i' such that $\pi_z(i') = j$ we have that $\pi_{x_r}(i') = i$. Indeed by taking the inverse of π_z we get $i' = \pi_{x_r^{-1}}(\pi_{a^{-1}}(\pi_{x_r}(j)))$. Now recall that $j = \pi_a(i)$ is on the left and x_r acts only on the right elements, thus we simplify to $\pi_{x_r}(i') = i$ as desired.

At this point we can check if s was generated from the public keys since if so π_s must map i' into j'. When $s = waw^{-1}$, this is true if and only if π_a maps $\pi_w(i')$ to $\pi_w(j')$. For a randomly generated conjugate, the pair $(\pi_w(i'), \pi_w(j'))$ is distributed (almost) uniformly, so the probability that π_a maps $\pi_w(i')$ to $\pi_w(j')$ is roughly proportional to $1/n$.

Case 2. This case is actually easier than the previous one. We are assuming that π_a does not map any element "across the border" from the two halves. In other words all the cycles of π_a are fully contained in either half and π_a can be written as the product $\pi_a = \pi_{a_l}\pi_{a_r}$ where π_{a_l} acts only on the left elements and

π_{a_r} only on the right ones. But then the cycles in π_s will be the union of the cycles on the left half of π_y and the cycles on the right half of π_z. A random conjugate will not have this property with probability close to 1, unless a is a pure braid and π_a is the identity permutation.

We have seen that when the braid a is randomly chosen (or, even more generally, whenever a does not belong to the subgroup of pure braids) it is easy to distinguish triples $x_l a x_l^{-1}, x_r a x_r^{-1}, x_l x_r a (x_l x_r)^{-1}$ (x_l and x_r chosen at random from B_L and B_R) from $x_l a x_l^{-1}, x_r a x_r^{-1}, w a w^{-1}$ (where w is just any random braid).

An easy "fix" that comes to mind is to redefine the decisional Ko-Lee assumption, setting w to the product of two random braids $w = w_l w_r$, uniformly chosen from B_L and B_R. In other words, instead of claiming that $s = x_l x_r a (x_l x_r)^{-1}$ is indistinguishable from a random conjugate of the form waw^{-1} with $w \in_R B_n$, one could claim that s is indistinguishable from a conjugate of the form $w_l w_r a w_r^{-1} w_l^{-1}$ where $w_l \in_R B_L$ and $w_r \in_R B_R$. But our distinguisher works with this modified definition as well, as long as a is not a pure braid. Details follows. (1) If permutation π_a maps some element of $\{\ell+1, \ldots, n\}$ to $\{1, \ldots, \ell\}$ we can find i' and j' such that $\pi_s(i') = j'$, exactly the same way we did is Case 1 above. If w is chosen as the product $w_l w_r$ of a left and right half braids, then $\pi_w(i') = \pi_{w_r}(i')$ is distributed uniformly at random in $\{\ell+1, \ldots, n\}$ and $\pi_w(j') = \pi_{w_l}(i')$ is distributed uniformly at random in $\{1, \ldots, \ell\}$. So, the probability that π_a maps $\pi_w(i')$ to $\pi_w(j')$ is at most $1/(n/2) = 2/n$. (2) If permutation π_a is the product $\pi_{a_l}\pi_{a_r}$ of a left and right permutation, then π_s is also the product of a left and right permutation, and we can completely recover π_s as the product of the left half of π_y and the right half of π_z. Also in this case, if $s = waw^{-1}$ is a random conjugate with $w = w_l w_r$, the probability of getting the right permutation π_s is at most $2/n$, unless a is a pure braid and π_a is the identity permutation.

This shows that restricting w to the product $w_l w_r$ of a left and half braid does not make the problem substantially harder. Still we believe that the decisional Ko-Lee problem is more naturally defined with w chosen at random within the subgroup $B_R B_L$, and in the rest of this section we concentrate on this alternative definition.

3.2 The Half-Braid Attack

We now consider the case where a is a pure braid, and show that unless the choice of a is restricted to even a smaller subgroup, it is possible to successfully attack the decisional problem. For a pure braid a, define the left and right projections $\tau_l(a)$ and $\tau_r(a)$ as the braids (over $n/2$ strings) obtained removing the first or last half of the strings. Now, we are given a, $x_l a x_l^{-1}$, $x_r a x_r^{-1}$ and $s = waw^{-1}$ where either $w = x_l x_r$ or $w = w_l w_r$ for independently and randomly chosen w_l, w_r. It is easy to see that in the first case $\tau_l(x_l a x_l^{-1}) = \tau_l(s)$ and $\tau_l(x_r a x_r^{-1}) = \tau_r(s)$, while in the second case equality does not hold (with high probability) unless $\tau_l(a) = \tau_r(a) = e$ are both the identity braid over $n/2$ strings.

At this point, the only case for which our attacks do not work is when a is chosen as a pure braid with $\tau_l(a) = \tau_r(a) = e$, and w is chosen as the product $w = w_l w_r$ of a left and right braid. This seems an interesting subgroup of braids and an interesting special case of the conjugacy problem. If the conjugacy problem were hard for this special case even in the decisional version, then one could build a pseudo-random generator out of it, fixing the problem of [11]. Unfortunately we will see in the next subsection that even under this restrictions the decisional problem is easy.

3.3 The Single String Attack

Assume we start from a braid a such that $\tau_l(a) = \tau_r(a)$ is the identity. For every $i = n/2+1, \ldots, n$, consider the braid $\tau_l^i(a)$ obtained from a removing all right strings except the ith. These are very simple braids: braids obtained running a single string around $n/2$ parallel strings. We claim that unless braids $\tau_l^i(a)$ are the same for all $i = n/2+1, \ldots, n$, we can break the decisional Ko-Lee problem.

Assume they are not all the same and divide the indices $i = n/2, \ldots, n$ according to their equivalence classes, with $i \equiv j$ if and only if $\tau_l^i(a) = \tau_l^j(a)$. Notice that this equivalence relation is efficiently computable because the word problem on braids can be solved in polynomial time. Moreover, since here we are considering a very special class of braids, equivalence can be decided very efficiently.

Notice that the equivalence relation induced by $\tau_l^i(a) = \tau_l^j(a)$ is the same as the one induced by $\tau_l^i(x_l a x_l^{-1}) = \tau_l^j(x_l a x_l^{-1})$. Similarly, $\tau_l^i(x_r a x_r^{-1}) = \tau_l^j(x_r a x_r^{-1})$ induces the same equivalence classes as $\tau_l^i(x_l x_r a (x_l x_r)^{-1}) = \tau_l^j((x_l x_r) a (x_l x_r)^{-1})$. On the other hand, when we compute the conjugate under a right braid x_r the equivalence classes are mapped by the permutation associated to x_r. So, we can decide whether $(w_l w_r) a (w_l w_r)^{-1} = (x_l x_r) a (x_l x_r)^{-1}$ or not by comparing the equivalence relation induced by $\tau_l^i(w_l w_r a (w_l w_r)^{-1}) = \tau_l^j((w_l w_r) a (w_l w_r)^{-1})$ with that of $\tau_l^i(x_r a (x_r)^{-1}) = \tau_l^j((x_r) a (x_r)^{-1})$.

In order to avoid this attach braid a should be chosen in such a way that not only $\tau_r(a) = \tau_l(a) = e$, but also for all $i, j \in \{n/2, \ldots, n\}$, $\tau_l^i(a) = \tau_l^j(a)$.

Using τ_r instead of τ_l we get a symmetric attack that shows that we need an analogous condition $\tau_r^i(a) = \tau_r^j(a)$ for all $i, j = 1, \ldots, n/2$. Now the question is: assume that the braids are chosen in such a way that all above conditions are satisfied; what are the remaining braids? and, is the conjugacy problem for these braids still hard?

3.4 The Trivial Attack

Consider pure braids a satisfying all the conditions stated at the end of the previous subsection:

- $\tau_r(a) = e$
- $\tau_l(a) = e$
- $\tau_l^i(a) = \tau_l^j(a)$ for all $i, j \in \{n/2+1, \ldots, n\}$,

– $\tau_r^i(a) = \tau_r^j(a)$ for all $i, j \in \{1, \ldots, n/2\}$.

It is easy to see that there are only very few braids that satisfy these conditions: the group generated by the braid $a = (\Delta_{n/2}\Delta_n^{-1}\Delta_{n/2})^2$, where Δ_k is the fundamental braid over (the first) k strings. Essentially this is a braid obtained swapping the left and right strings without permuting string in each half, and passing all the right strings over the left strings. The whole operation is performed twice so that the permutation of a is the identity, i.e., a is a pure braid.

But the conjugacy class under left or right braids of a (or any of its powers) is trivial, i.e., $x_l a x_l^{-1} = a$ and $x_r a x_r^{-1} = a$. So, the instances of the decisional Ko-Lee problem have always the form (a, a, a, a) and the answer is always yes.

4 The Attack Extends to the Pseudo-Random Constructions

In [11] the Ko-Lee Decisional Assumption is used to construct a pseudo-random generator and a pseudo-random synthesizer. By showing that the underlying assumption is false we have removed the proof of security for the above constructions. However that does not immediately imply an *attack* on the generators.

In this section we show that it is possible to use the above attack to distinguish the output of the generators from random. This is because the constructions are a straightforward application of the conjugacy function. Here we only show the attack on the pseudo-random generator.

The construction is a follows: there are the following public parameters:

– $a \in_R B_n$, with $|a| = k$
– $a_1, \ldots, a_m$, with $a_i = x_i a x_i^{-1}$ for $x_i \in_R B_L$ with $|x_i| = k$.

On input a random seed $w \in_R B_R$ the generator outputs the conjugate of w with respect to all the a_i's. That is, the output is m braids $s_1, \ldots, s_n$ where $s_i = w a_i w^{-1}$. Clearly each component of the output can be distinguished from random using the attack described above.

It should be noted that this attack applies to the "pseudo-random *braid* generator", i.e., a function that on input a short seed, produces a sequence of seemingly random *braids*. In [11] it is suggested that if one wants a "pseudo-random *bit* generator", then universal hashing and the left-over hash lemma can be used to transform a pseudo-random sequence of braids into a pseudo-random sequence of bits. However, the converse is not necessarily true: if the original braid sequence is not pseudo-random, there is no guarantee that applying universal hashing results in a bit-sequence which is computationally indistinguishable from random.

5 On the Bit Security of the Conjugacy Problem

We now offer some remarks on an independent result still contained in [11]. Besides the pseudo-random constructions based on the Ko-Lee Decisional Assumption, [11] claims to construct a hard-core bit for the one-way function induced by the conjugacy problem.

Let $a \in B_n$ and define the following function over B_n, $f(x) = xax^{-1}$. By restricting the input size and assuming the conjugacy problem is hard, it is reasonable to assume that f is a one-way function.

The next question is: does f have any natural hard-core predicate (by natural, we mean something specific to the description of f and not a generic hard-core predicate like the Goldreich-Levin [8] which holds for any one-way function).

We recall that a hard-core predicate π for f is defined as follows. We say that the predicate π over B_n is hard-core if for any efficient (i.e., PPT) Turing machine $\mathcal{A}$ we have that

$$\mathrm{Prob}_{x,a \in B_n}[\mathcal{A}(a, xax^{-1}) = \pi(x)] = \frac{1}{2} + \epsilon \tag{1}$$

where ϵ is a negligible quantity. We stress that the probability of success is taken over *BOTH* the internal coin tosses of $\mathcal{A}$ and the choice of x. In other words if we find an $\mathcal{A}$ that predicts $\pi(x)$ for just 51% of the inputs x, then π is NOT an hard-core bit for f.

Usually proofs for hard-core bits go by contradiction. We assume that there exists an oracle $\mathcal{A}$ that contradicts Equation 1. We then show how to use $\mathcal{A}$ to invert f over a random point. The proof is usually complicated by the fact that $\mathcal{A}$'s responses on any input (a, xax^{-1}) may be wrong and cannot be accepted at face value. Usually the proof contains some random self-reducibility argument (although that's not necessarily the only way to prove hard-core bits). That is instead of querying $\mathcal{A}$ only on (a, xax^{-1}) we also query it on several randomized versions of it and then use some trick to extract the right value of $\pi(x)$ from all these responses (which in the majority are correct).

In [11] such an hard-core predicate and a proof of security is supposedly presented. For the discussion that follows, it is not really important to understand what the hard-core predicate is. For completeness, we briefly recall the definition anyway. Every braid can be uniquely expressed in canonical form as the product $\Delta_n^u \chi_1 \cdots \chi_p$ where Δ_n is the fundamental braid defined in Sect. 3, and χ_p are permutation braids, i.e., braids where each string can be described as a straight line from the initial point to the end point, and at all the crossings the left string goes under the right string. This normal form (called the left canonical form) can be computed in polynomial time. The candidate hard core predicated π proposed in [11] takes a braid x as input, computes the left canonical form $x = \Delta_n^u \chi_1 \cdots \chi_p$ and outputs the parity of u.

The proof of security in [11] misses the whole step of randomizing over the inputs to the predicting oracle. The reduction from predicting π to inverting f assumes that the oracle $\mathcal{A}$ answers correctly *FOR ALL* x with sufficiently high probability (bounded away from a 1/2), but this probability is taken ONLY over the coin tosses of $\mathcal{A}$. If x happen to be one of the bad inputs for which $\mathcal{A}$ gives the wrong answer (possibly for any value of $\mathcal{A}$ internal coin tosses), there is no guarantee that majority voting on $\mathcal{A}$ answers relative to the same input x and independent coin tosses produces the right answer.

Such set of *bad* inputs might constitute 49% of the possible braids, and still π not be a hard-core predicate because (when x is chosen at random) $\mathcal{A}$ has a 1% advantage in predicting $\pi(x)$ over a random guess. (For cryptographic security, even 1% is not a negligible advantage. Formally speaking, the 49% and 1% above should be interpreted as $(50-\epsilon)\%$ and $\epsilon\%$ where $\epsilon(k)$ is a function of the security parameter such that $\epsilon(k) < 1/k^c$ for all $c > 0$ and for all sufficiently large k.)

We were not able to repair the proof of security for the hard-core predicate presented in [11]. Moreover we were not able to construct any other natural hard-core predicate. Thus we conclude that the construction of a hard-core predicate for the conjugacy problem is still an (interesting) open problem.

6 Conclusion

We showed that the decisional version of the Ko-Lee Assumption is false in a very strong sense: for essentially all reasonable input probability distributions, there is a very efficient distinguisher that invalidate the assumption. The attack extends to the pseudo-random generator and synthesizer proposed in [11] which are consequently shown to be insecure.

With regard to the conjugacy problem over braids, which is at the basis of all the cryptographic schemes based on braid groups, we show two facts. (1) The conjugacy problem reveals some partial information about the permutation induced by the input braid, unless the central braid is chosen to be pure; we suggest to incorporate this choice in the design of cryptographic schemes based on the conjugacy problem. (2) The proof of a hard-core bit for the conjugacy problem in [11] is flawed and we were not able to repair it.

In spite of our attacks, braid groups remain an attractive and promising tool for the construction of cryptographic algorithms. However our findings highlight the need for extreme care in analyzing these new protocols and warn about drawing unmotivated conclusions from parallel ones in number-theoretic constructions.

Which leads us, in turn, to many interesting open problems. For example, although we proved that the result of a Ko-Lee exchange is not totally random, it might still be possible to prove that it contains a lot of "computational entropy". If true, this would allow the construction of an efficient secure encryption scheme *without* resorting to the random oracle model, and possibly provide a fix for the pseudorandom constructions presented in [11]. Also, is there a hard-core bit for the conjugacy problem? Another interesting problem is related to the construction of digital signatures. In [13] it is shown that certain groups of points on an supersingular elliptic curves have the property that the Decisional Diffie-Hellman problem is easy, while the Computational Diffie-Hellman problem is presumably hard. This properties can be used to design a digital signature scheme as shown in [5, 12, 4]. The idea is the following. The secret and public keys of the system are a and $y = g^a$, and the signature of message m is computed as $s = m^a$. It is easy to see that the signature is valid if and only if (g, y, m, s) is a valid DDH sequence. An interesting open question is whether the conjectured hardness of the conjugacy problem for braid groups, together with the techniques to attack the decisional problem presented in this paper, can be used to design a provably secure digital signature scheme based on braid groups.

7 Acknowledgements

We would like to thank N. Howgrave-Graham for useful discussions and for drawing our attention to the subgroup of pure braids. A preliminary version of this attack was presented at the rump session of Crypto 2001 [7]. We would like to thank E. Lee and S.G. Hahn for useful discussions following the rump session presentation, and the anonymous referees for their comments to the preliminary version of this paper.

References

1. I. Anshel, M. Anshel and D. Goldfeld. An Algebraic Method for Public-Key Cryptography. Mathematical Research Letters, 6 (1999), pp. 287–291.
2. M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols", *1st ACM Conference on Computer and Communications Security,* 1993, 62-73.
3. D. Boneh. The Decision Diffie-Hellman Problem. Third Algorithmic Number Theory Symposium. LNCS 1423, pp.48–63, Springer 1998.
4. D. Boneh, H. Shacham, and B. Lynn. Short signatures from the Weil pairing. Asiacrypt '2001. LNCS 2248, pp. 514–532, Springer-Verlag 2001.
5. S. Brands. An efficient off-line electronic cash system based on the representation problem. Technical Report CS–R9323, CWI (Centre for Mathematics and Computer Science), Amsterdam, 1993.
6. J.L. Carter and M.N. Wegman, Universal classes of hash functions, Journal of Computer and System Sciences 18:143-154, 1979.
7. R. Gennaro, D. Micciancio. Cryptanalysis of a Pseudorandom Generator based on Braid Groups. CRYPTO'2001 rump session, August 2001.
8. O. Goldreich, L. Levin. Hard-core Predicates for any One-way Function. 21st STOC, pp.25-32, 1989.
9. S. Goldwasser, S. Micali. Probabilistic Encryption. Journal of Computer and System Sciences 28:270–299, April 1984.
10. K.H. Ko, S.J. Lee, J.H. Cheon, J.W. Han, J. Kang, C. Park. New Public-Key Cryptosystem Using Braid Groups. CRYPTO'2000, LNCS 1880, pp.166–183, Springer 2000.
11. E. Lee, S.J. Lee, S.G. Hahn. Pseudorandomness from Braid Groups. CRYPTO'2001, Springer 2001.
12. T. Okamoto, D. Pointcheval The Gap problem: a new class of problems for the security of cryptographic primitives Public Key Cryptography, PKC 2001, LNCS 1992, Springer-Verlag 2001.
13. E. R. Verheul Evidence that XTR Is More Secure than Supersingular Elliptic Curve Cryptosystems Eurocrypt'2001. LNCS 2045, p. 195-210