Attack On the Markov Problem

James L. Adams
jamesladams@email.com

March 23, 2014

Abstract

In 2000 Ko gave potential hard problem is proposed called the Markov problem. We give an algorithm, for certain parameters, for solution of the Markov problem. The Markov problem is related to the knot recognition problem. Hence we also a new algorithm the knot recognition problem . This knot recognition algorithm may be used for previously proposed cryptosystem that uses knots.

1 Attack On the Markov Problem

1.1 The Markov Problem

The goal of this paper, is to show that the suggested hard problem [2] is solvable, for certain parameters, and we describe probabilistic practical implementation of this algorithm that may give a practical solution to it. There is no known, polynomial complexity, algorithm to solve,for all parameters, the Markov problem. As an aside to solving this problem we get a new knot recognition, algorithm. Every knot can be represented as the closure of a braid. Two different braids can represent the same knot. We know a knot is ambient isotopic to a closed braid by Alexanders Theorem. A knot is transformed into an equivalent knot by a chain of Reidemeister moves. A knot can be constructed from a braid. Markov in the 1930's discovered the closure of any two braids is equivalent if they undergo a chain of his Markov moves. A full proof of his idea came later. In 1961 [6] the first algorithm given to compare knots was given by Haken. A cryptosystem based on knots was proposed in [5]. Our knot algorithm, given in the section below, may be used with protocol based on knots, for example, possibly with scheme in [5],or a variant of [5].

The Markov problem as stated in [2] is the following (and using the notation of [2]):

Instance: $y \in B_n$ such that y is conjugate to a braid of the form $w\sigma_{n-1}^{\pm 1}$ for $w \in B_{n-1}$

Objective: Find $(z, w) \in (B_n, B_{n-1})$ such that we have the equation one.

$$zyz^{-1} = w\sigma_{n-1}^{e} \text{ where } e = 1 \text{ or } e = -1 \tag{1}$$

As is mentioned in [2] the Markov problem is closely related to the study of knots and links via braids and hence should be hard. In [3] it was described how the DP can be transformed (in a polynomial number of steps) into a set of conjugacy search equations, for any group. So the Markov problem is different from the conjugacy search problem since w is unknown and y is conjugate to a braid mostly in B_{n-1}. In 1969 Garside gave the first algorithm [1] to solve the conjugacy search problem.

Since the algorithm in [3] only requires the presentation for one of A or B, both presentations are known in the double coset problem, in the problem. So this means if it was applied to a version of a DP problem, where one of the presentation of the sets are not known, it can reveal information "for a hidden subgroup" in the problem (if this subgroup may need to be found); but since the definition of th DP [3] states "find elements $f \in A, g \in B$ " we may say that this implies a subset (or subgroup) membership to be used on the solutions found. Following the notation in [3].

Public Information: G is a semigroup. A, B are subsets of $G.x, y \in G$ with

$$y = axb.$$

Secret Information: a is a element in A, b is an element in B.

Objective: find elements $f \in A, g \in B$ such that $fxg = y$.

The solution to the decomposition problem (DP) implies a solution to the decision version of the DP problem. We may write $*$ for the binary operation of G. We refer to an instance of DP as the function $DP((G', A', B', x'), (a', b'))$ (G' means the group for G in the problem an so on) sometimes we write this as DP when it is obvious what is meant in the context. The double coset problem can be solved by considering decomposition problem (DP)-with the parameters A, B are subgroups of the group G. Since once we have solved for f and g, (if they exist), we test for if $f \in A$, and $g \in B$ by using some method to test for subgroup membership, we can decide that if its true or false that y is equal to fxg in the problem, and so, for example, decide is a pair of elements lie within the same double coset or not. We refer to an algorithm that solves $DP((G, A, B, x), (a, b))$ as

$$DP((G, A, B, x), (a, b), (F, G))$$

means that $DP((G, A, B, x), (a, b), (F, G))$ has solutions $f \in F, g \in G$ in the decomposition instance

$$DP((G, A, B, x), (a, b))$$

Suppose we may know the generating set for x we refer to this as $X = \{x_1, x_2, ..., x_{m_a}\}$. In this case we write $DP((G', A', B', X', x'), (a', b'))$ or similarly again refer to it as DP.

1.2 A Partial Solution for the The Markov Problem

The algorithm may imply a redefinition conguacy extractor function of [3] sketched as follows. When constructing CE functions consider functions of the form $L(l)B(b)P^{-1}(l)$ where L, B, P are some functions over G, may have $P = L$, are functions over G and $l, b \in G$.

Algorithm to Solve Decomposition Problem

Based on the algorithm in [3] for the DP , when we say based we mean that if we did not use the ideas in the algorithm in [3] the algorithms presented here would not be possible,algorithm there, we apply the ideas to the Markov problem, and we give an algorithm here for solving the DP type problems (such as the Markov, and double coset), we give a suggestions for the algorithm that may speed up the algorithm. So this means if we have , for example, information about commutativity condition of the set A and B , as the DP has in [2], we could put down a system of equations to represent this. One reason the multiple simultaneous conjugacy search problem, and hence this DP algorithm, is of interest because the paper [4] and the follow up paper [3] show that the multiple conjugacy search problem is the basis of security for a lot of non-commutative group based protocols. So it an interesting question to research how hard the multiple simultaneous conjugacy search problem is to solve. For example it was shown in [3] that the Dehornoy's authentication scheme based on shifted conjugation, and braid-Diffie Hellman protocol in [2], and its generalisation, are based on the multiple conjugacy search problem. We use this extra information, along with a system of multiple conjugacy equation which we can always derive with the parameters we use. We follow the notation in [3]. The minimal input information we need for this algorithm to work includes a representation A', B' for one of the subgroup A or B.

We consider

$$DP((G', A', B', X', x'), (a', b')) = DP((G, X, Z, A, a), (x, z))$$

refers to input to the algorithm that solves the DP , given (the DP equation)

$$u = xaz.$$

This attack reveals recovers z. So we use the version that tries to recover z. To apply it to the Markov problem we would modify the algorithm in [3] means we may know one or both of the representations X or Z.

$$\begin{aligned}
X &= \{x_1, x_2, ..., x_{m_q}\} \\
& \quad m_q \text{ is an integer for the number of generators of } X. \\
Z &= \{z_1, z_2, ..., z_{m_w}\} \\
A &= \{a_1, a_2, ..., a_{m_a}\} \\
G &= \{g_1, g_2, ..., g_{m_g}\} \\
C(Z) &= \{f_1, f_2, ..., f_{m_z}\} \\
C(X) &= \{h_1, h_2, ..., h_{m_x}\}
\end{aligned}$$

1. Given $xazS_Iz^{-1}a^{-1}x^{-1}$ the attacker picks elements S_I according to some criteria relating to commutativity [3].

So for this condition we choose, to make it simply for more efficient implementation in multiplications, to select S_I so that $S_I \in C(z) \cap C(a)$ so $xazS_Iz^{-1}a^{-1}x^{-1} = xS_Ix^{-1}$ and this choice reduces it to conjugacy search problem in the relation $S_I \tilde{} xS_Ix^{-1}$, this equation represents the centraliser of $(C(x))$. So this means $S_I \in C(x)$. Since we can express $x^{-1}ax = a \leftrightarrow x = a^{-1}S_Ia \rightarrow a^{-1}x_{K_1}aa^{-1}x_{K_2}a...a^{-1}x_{K_L}a$ for some integer sequence K_i then (if we knew the generators of $C(z) \cap C(a)$), this gives the choices for e_i for the values in the system of equations below, or more generally any set that commutes with a, such as $\{f_1^2, f_2^3, ..., f_{m_z}^{m_z}\}$, for the choices (if we assume X commutes with Z).

Or consider we are given in the problem that $S_I \in C(z)$ this we can take e_i (below) to be the generating set of $C(Z)$. So the generators that compose the word S_I will commute with the generators of $C(Z)$.

In general we don't know if x' must commute with the generating set for z.In most random instances of the Markov problem we would not expect it to. Because in the problem $w \in B_{n-1}$. For general values of n strings and general subgroups, testing for membership of an arbitrary element braid subgroup, ignoring brute for membership test, is not usually possible easily. However in the Markov problem, where we know the structure of the subgroup, its easy to see that we have the expression for w', we reduce this expression to its shortest form in Artin generators, e.g. using the heuristic algorithm in [7], then by inspection of the generators to see it it contains the generator unique to B_n, we can decide if $w \in B_{n-1}$.

As an aside,note if we were using the algorithm to solve the double coset type problem, or related problems, in a generic G, we may choose to test the relations, of if the solutions are members of the subgroup X , and if the other solutions are members of the subgroup Z. The Markov problem requires the solution to be from a certain subgroup. So in this aside, if $G = B_n$ and we have subgroups $X = B_l$ and $Z = B_r$, where

$$\begin{aligned} B_l &= \{\sigma_1, \sigma_2, ..., \sigma_{\lfloor (n-1)/2 \rfloor -2}\}, B_r = \{\sigma_{\lfloor (n-1)/2 \rfloor}, \sigma_2, ..., \sigma_{n-1}\} \\ & \text{or} \\ B_l &= \{\sigma_1, \sigma_2, ..., \sigma_{d_1}\}, B_r = \{\sigma_{d_2}, \sigma_2, ..., \sigma_{n-1}\} \\ \text{with } 0 &< d_1 \leq \lfloor (n-1)/2 \rfloor - 2 \text{ and } \lfloor (n-1)/2 \rfloor < d_2 \leq n-1 \end{aligned}$$

, then as the remark above mentions, given the shortest expression of a word, the membership problem can be solved for some $n > 3$, by computing the shortest expression in Artin generators then inspecting the elements, this works because the subgroups have disjoint strings,so this algorithm can be used to solve the double coset problem in this case. Back to the Markov problem.

In the Markov problem if we assume we know the representation of the subgroup that correspond to Z , then in the braid group, there are ways to calculate commuting subgroups for a given subgroup,using a centraliser algorithm, to compute $C(Z)$. So this can be computed if we do not know some or all of

$C(Z)$ directly..Since we require that the solution commutes with elements of Z (which we do know the representation to in general), in the DP, then we compute the generators of $C(Z)$ to create the elements S_I. Then we have the system of equation 2 .However if we are looking at certain subgroups of B_n ,such as in the Markov, problem we would then of course know the generators x', where x' is a solution that can be used in place of x or is x.

2. Then for $1 \leq I \leq M$ for a sequence of integers $1, 2, ..., K$ where K is known there are choices for e_i such that $e_i = a^{-1}e_i a$.

$$e_i = x'^{-1}e_i x' \text{ for } 1 \leq i \leq K \ . \ : \ f_i = x'^{-1}f_i x' \text{ for } 1 \leq i \leq m_y \tag{2}$$

The above equation would be as below if we had the extra information, as the DP problem, as it does in the cryptosystem [2], that X commutes with Z. So for [2] we would have two sets of equations like this.

$$e_i = x'^{-1}e_i x' \text{ for } 1 \leq i \leq K \ . \ : \ h_i = x'^{-1}z_i x' \text{ for } 1 \leq i \leq m_z$$

.(trivially e_i can be the identity).

So if we have the generators f_i and z_i and they are expressed as "short words" then it maybe faster to compute the above equation, as in the braid group multiplication and the word algorithm depend on the length of the braid, Now instead we if we generic S_I. we have the following

$$CE(S_I, u) = uS_I u^{-1} = xazS_I z^{-1}a^{-1}x^{-1} = x'aS_I a^{-1}x'^{-1}\text{for } 1 \leq i \leq K \tag{3}$$

.

$$\begin{aligned} CE(f_I, u) &= u(f_I)u^{-1} = xaz(f_I)z^{-1}a^{-1}x^{-1} = xa(f_i)_I a^{-1}x^{-1}, & (4) \\ 1 &\leq I \leq m_z \\ CE(S_I, u) &= u(S_I)u^{-1} = xaz(S_I)z^{-1}a^{-1}x^{-1} = xa(S_i)_I a^{-1}x^{-1}, & (5) \\ m_z &< I \leq L \end{aligned}$$

.

3. So the DP is deterministically reduced to solving the system of conjugacy equations using a combination of some or all of the sets of conjugacy equations 2,3,4,5 solving for the sets of values for the sets F and G. for the solution z we can get it by $z' = ((xaz^{-1})^{-1}x'az)$. There maybe other ways to simplify identities 2,3,4,5 and so make a more efficient implementation , e.g. maybe sufficient to use 5 in some cases. Hence we construct the sets F and G when we try to solve for value for x or another value that can be used in place of x that also satisfies the DP.

Or instead of computing z in above way we can derive a system of conjugacy equations for z ,like we did for x,then test if the DP relation $f'xg' =^? u$ (for all $f' \in F, g' \in G$) using a word algorithm. So solving for (x', z') gives a "dual" version of the algorithm, because we are in a way, using algorithm 8.2 of [3] twice, which may result in a more efficient implementation.

Also some of the factors,(e.g. powers of the fundamental braid), for the recovered z' and x' may cancel out it in the CE functions, it can be shown, these factors can be recovered in the braid group in finite time.

The sketch of the proof that the DP can reduced a system of conjugacy equation, in polynomial time, is as follows. To solve for a or b in polynomial time, note this requires the conjugacy search problem to be solved in polynomial (space and time) complexity, on for any G (where G may be a group), is to show that the number of group operations in the transformation is polynomially bounded, and than K can be computed in polynomial time. Then applying a deterministic algorithm to solve the system of conjugacy equations, then noting the groups operation such as inverses can be done in polynomial time If one of the operations is not in polynomial time then this still of course shows a deterministic reduction from the DP to solving conjugacy equation in finite ,non-polynomial, time (when we mean non-polynomial it means it depends on a type of algorithm used for some G). This implies generically of course that the DP, and related hard problems, are solvable in any group when we have an algorithm to solve the conjugacy search problem in G, the more general problem of recovering one representation for a coset and solving the DP.

In braid groups (and related groups) the group inverse operation can be done polynomial complexity. So polynomial time reduction in braid groups. So this detailed sketch proves in the worst case the DP can be reduced to a system of conjugacy equations in finite time. By applying the DP algorithm repeatedly we can solve decomposition problems in more than two unknowns. Now we will use this algorithm to derive a new solution to know problem. The author of [3] seems to prefer when the generating set of X, as defined in the previous section, is known, but it should be possible to modify the algorithm in [3], for a similar algorithm to above depending on properties of X; and it does appear that the algorithm of [3] will work when the generating set of X is not known-if that is what the author there did mean.

1.3 An Attack The Markov Problem

We now sketch an algorithm for the Markov problem (there is some reuse of notation in this paper, e.g. since we copy the problem as stated in [2] following the notation there in this section). There is enough description of the algorithm to do an implementation or a variant implementation of them. We may consider using the above algorithm or a variant mentioned at step 3 that build systems of conjugacy equation for both x and z. Again from now we use the notation in [2].

1. For the Markov equation given in [2] we have $y \in Y$,$z \in Z$,$w \in W$.

$$zyz^{-1} = w\sigma_{n-1}^{\pm 1} \rightarrow zyz^{-1}w^{-1} = \sigma_{n-1}^{1} \text{ or } zyz^{-1}w^{-1} = \sigma_{n-1}^{-1}$$

so the Markov problem can be considered as a DP problem and applying the algorithm above will give a new way to solve it. This can be thought of as a triple decomposition type problem [4] .In [4] a solution to solve the triple decomposition problem was given. Hence we can use the ideas that solve the TDP to solve the Markov problem; we outline a few algorithms here with enough detail so to do an implementation. Suppose we know one or more of the three generating sets W, Z, Y. Minimally the subgroup Z needs to be known as a generating set for the Markov attack.

Note we are not solving the Markov problem completely because we are assuming the element z comes from a certain subgroup; as we also do in our knot recognition algorithms are partial recognition algorithms.

$$\begin{aligned} W &= \{w_1, w_2, ..., w_{m_w}\} \\ Z &= \{z_1, z_2, ..., z_{m_z}\} \\ Y &= \{a_1, a_2, ..., a_{m_y}\} \\ & G = B_n \end{aligned}$$

Suppose to begin with we only know the generator set for Z. So

$$DP((B_n, Z, W, y), (z, wyz^{-1}w^{-1}), (F, G))$$

so the algorithm return the answer $(z', z'^{-1}w^{-1\prime})$ for elements of $z' \in F$ and $z'^{-1}w^{-1\prime} \in G$. But we can rewrite solution as

$$z'^{-1}w'^{-1} = \alpha, w'z' = \alpha^{-1}$$

2. Solve the multiple conjugacy search problem again now select different r_i from $C(z)$ we have (using our sketched definition of conjugacy extractor function)

$$\alpha^{-1}r_i\alpha = w'^{-1}r_iw'$$

Because one of the values z' will be equal to the actual value of z used in the DP , this means we would recover the actual value z,w (along with all the values that can be used in place of it) at some. at some point. So at this point we have a set of possible solutions (size of set are bounded in $O(n!)$ follows from Garside algorithm in [1]) for z and a set of possible solutions for w.

.At this point we need a method to test for if the solution is a member of some subgroup. Then rewriting w in its minimal length expression, e.g. using heuristic algorithm in [7], and inspecting the generators that compose w, decide $w \in B_{n-1}$,we put these set of values into the Markov equation 1 and use a word algorithm, to find the solution to Markov's equation. Hence we have two sets of elements,of some braids , which do satisfy the Markov equation. So now we have described how to solved Markov's problem as described [2]. The Markov's problem above is closely related to the equality of knots. There is a similar proof starting with

$$\begin{aligned} zyz^{-1}w^{-1} &= \sigma_{n-1}^{\pm 1} \rightarrow r_izyz^{-1}w^{-1} = r_i\sigma_{n-1}^{\pm 1} \rightarrow \\ (zyz^{-1}w^{-1})^{-1}(r_izyz^{-1}w^{-1}) &= (\sigma_{n-1}^{\pm 1})^{-1}r_i\sigma_{n-1}^{\pm 1} \end{aligned}$$

Algorithm: Knot equality Test

Markov's Theorem

$y \in B_n$ and $z \in B_m$ respectively. Then the links (closures of the braids y and z) Are ambient isotopic if and only if '

z can be obtained from y by a series of

1. Given a braid group., equivalences of it

2. given braid group, conjugation in it.

3. Markov Moves: replacing $y \in B_n$ by $y\sigma_n^{\pm 1} \in B_{n+1}$ or the inverse of this operation, replacing $y\sigma_n^{\pm 1} \in B_{n+1}$by $y\sigma_n^{\pm 1} \in B_n$ if y if has no occurrences of n.

In general this we see theorem be expressed as, for given braids y and z, with $y \neq z$

$$y_m = (\beta_m^{-1}\beta_{m-1}^{-1}...\beta_{n-2}^{-1}...\beta_n^{-1})z(\beta_n\sigma_n^{\pm 1}\beta_{n+1}\sigma_{n+1}^{\pm 1}\beta_{n+2}\sigma_{n+2}^{\pm 1}...\beta_{m-2}\sigma_{m-1}^{\pm 1}\beta_{m-1}\sigma_{m-1}^{\pm 1}\beta_m\sigma_m^{\pm 1}) \tag{8}$$

we are not considering replacing

$$y\sigma_n^{\pm 1} \in B_{n+1}\text{by } y\sigma_n^{\pm 1} \in B_n$$

but the idea is similar if we did, prove that M is expressed as above. So to prove that two knots are equivalent is reduced to the problem of, given z and y_m, prove y_m has the above decomposition. We write y_n for the nth of Markov moves on z. $y = y_m$. Select z as identity element it test y to be the trivial knot. Use another algorithm to convert the knots into the braid form for z and y. Or if we can prove they do not have the above decomposition then they are not the same knot.

Suppose we have a function $M(d)$ that takes a braid d and returns p where $p \in C(d)$ in finite time. In other words M will recover central factors that cancel out in the CE functions. So the following algorithm can be used, note that its not a complete knot equality test because Z is a subgroup of B_n and not $Z = B_n$.

Write $\beta_n \in T_n \leq B_n$, where is it assumed that that we know a presentation for T_n, so this means each β_n is selected from a subgroup T_n on n strings. In Markov's theorem [locate ref] we do not know T_n as subgroups. We can express 6 as

$$\begin{aligned}
y_n &= IzH = \\
y_m &= (\beta_m^{-1}\beta_{m-1}^{-1}...\beta_{n-2}^{-1}...\beta_n^{-1})z(\beta_n\sigma_n^{\pm 1}\beta_{n+1}\sigma_{n+1}^{\pm 1}\beta_{n+2}\sigma_{n+2}^{\pm 1}...\beta_{m-2}\sigma_{m-1}^{\pm 1}\beta_{m-1}\sigma_{m-1}^{\pm 1}\beta_m\sigma_m^{\pm 1}) \\
\text{Where } I &= \beta_m^{-1}\beta_{m-1}^{-1}...\beta_{n-2}^{-1}...\beta_n^{-1} \\
H &= (\beta_n\sigma_n^{\pm 1}\beta_{n+1}\sigma_{n+1}^{\pm 1}\beta_{n+2}\sigma_{n+2}^{\pm 1}...\beta_{m-2}\sigma_{m-1}^{\pm 1}\beta_{m-1}\sigma_{m-1}^{\pm 1}\beta_m\sigma_m^{\pm 1})
\end{aligned}$$

So this is a DP which we can use the DP algorithm to solve for. Now to find I we can select and $r_i \in C(\beta_n) \cap C(\beta_{n+1})... \cap C(\beta_m)$.and $r_i \notin C(z), i \geq 1$.

Compute

$$\begin{aligned}
CE(r_i, y_m^{-1}) &= y^{-1} r_i y = \qquad (9)\\
&= ((\beta_m^{-1}\beta_{m-1}^{-1}...\beta_{n-2}^{-1}...\beta_n^{-1})z *\\
&\quad (\beta_n\sigma_n^{\pm1}\beta_{n+1}\sigma_{n+1}^{\pm1}\beta_{n+2}\sigma_{n+2}^{\pm1}...\beta_{m-2}\sigma_{m-1}^{\pm1}\beta_{m-1}\sigma_{m-1}^{\pm1}\beta_m\sigma_m^{\pm1}))^{-1}\\
&\quad *r_i *\\
&\quad (\beta_m^{-1}\beta_{m-1}^{-1}...\beta_{n-2}^{-1}...\beta_n^{-1})z *\\
&\quad (\beta_n\sigma_n^{\pm1}\beta_{n+1}\sigma_{n+1}^{\pm1}\beta_{n+2}\sigma_{n+2}^{\pm1}...\beta_{m-2}\sigma_{m-1}^{\pm1}\beta_{m-1}\sigma_{m-1}^{\pm1}\beta_m\sigma_m^{\pm1})\\
&= (z(\beta_n\sigma_n^{\pm1}\beta_{n+1}\sigma_{n+1}^{\pm1}\beta_{n+2}\sigma_{n+2}^{\pm1}...\beta_{m-2}\sigma_{m-1}^{\pm1}\beta_{m-1}\sigma_{m-1}^{\pm1}\beta_m\sigma_m^{\pm1}))^{-1}\\
&\quad *r_i z *\\
&\quad (\beta_n\sigma_n^{\pm1}\beta_{n+1}\sigma_{n+1}^{\pm1}\beta_{n+2}\sigma_{n+2}^{\pm1}...\beta_{m-2}\sigma_{m-1}^{\pm1}\beta_{m-1}\sigma_{m-1}^{\pm1}\beta_m\sigma_m^{\pm1})\\
&= H^{-1}z^{-1}r_i z H
\end{aligned}$$

G is the set of solution in the DP composed of the elements w'.

So we can solve, for multiple solutions w' ,for H, the terms that contains the stabilisers, and hence $y_m w'^{-1} = Iz$. At this point we have three choices one is to use length function, to test if knots are equal, 1,2,3 as follows.

1) Assume $l(p)$,$p \in B_n$ is a length function- which means it gives a real value relating to length, in Artin generators of some braid p. When computing the solution we may need to recover the cancelled factors using the function M. If

$$l(y_n w'^{-1}) < l(y_n), l(y_n w'^{-1}) \approx l(y_n) - l(w'^{-1})$$

and

$$l(y_n w'^{-1} z^{-1}) \approx l(y_n w'^{-1}) - l(z), l(y_n w'^{-1} z^{-1}) < l(y_n w'^{-1}).$$

2) For each w' select $s_i \in C(z)$

$$\begin{aligned}
CE(s_i, (Iz)^{-1}) &= (Iz)s_i(Iz)^{-1} =\\
&= I s_i I^{-1}
\end{aligned}$$

F is the set of solution in the DP composed of the elements v'.

So we can solve, for multiple solutions v' ,for I. This means we can find v', w', if they exist in F and G, through F, G until we find

$$v'^{-1} y_n w'^{-1} = z$$

hence the knots are equal if the above is shown to be true with a word algorithm. If the search cannot find such solution then the knots are unequal.

3) Test the relation

$$\begin{aligned}
CE(r_i, y_n^{-1}) &\sim z^{-1} r_i z =\\
H^{-1}z^{-1}r_i z H &\sim z^{-1} r_i z
\end{aligned}$$

using an algorithm that solves the conjugacy decision problem. For example this can be done with a knot invariant such as Jones polynomial. So it may be useful to detect collision in knot invariant polynomials. If the conjugacy decision problem decides the relation as true, for $i \geq 1$, then it is concluded that the knots are equal; of course id the solution to conjugacy decision problem was probabilistic then so is this answer This equation of course represents a knot, if r_i is a knot.

As mentioned above, this method does not recognise knots generally because it requires knowledge of $T_n \leq B_n$ for different n. So if this is not known , it may be known by computing

$$\begin{aligned} C(y) &= C(\beta_m^{-1}\beta_{m-1}^{-1}...\beta_{n-2}^{-1}...\beta_n^{-1})z(\beta_n\sigma_n^{\pm 1}\beta_{n+1}\sigma_{n+1}^{\pm 1}\beta_{n+2}\sigma_{n+2}^{\pm 1}...\beta_{m-2}\sigma_{m-1}^{\pm 1}\beta_{m-1}\sigma_{m-1}^{\pm 1}\beta_m\sigma_m^{\pm 1}) \\ &\subseteq C(\beta_m) \cap C(\beta_{m-1}) \cap ...C(\beta_{n-2}) \cap C(\beta_n). \end{aligned}$$

If T_n are generated by known types of knots, such as prime knots, then T_n is known. If β_i are selected from disjoin subsets then its easy to find elements that commute with them; so in this case it may be that the knot can be recognised in polynomial time.

Consider 9 again, there is a similar knot algorithms that can be derived solving for I to begin, so there is an iterative variant version that involves computing $CE(w_i, y_n) = y_n r_i y_n^{-1}$ $r_i \in C(\beta_i) \cap C(\sigma_i^{\pm 1})$, recovering the factors β_i and $\sigma_i^{\pm 1}$, this can be done heuristically using length functions to "peel off" the factors β_i, or deterministically by solving the multiple conjugacy search problem (and keep all solutions for β_i), until the factor z is reached in the last iteration. If not the variant algorithm will converge, in finite time,to a different set of values, from solving the conjugacy search problems, branching out from each solution each step, which is an exponential number of time, to solutions that do not contain z. So in this case the knot would be unequal. So in this variant algorithm the actual values of β_i used in the Markov conjugation move. The maximum number of iterations required can be estimated from the input length,L , in Artin generators. We may have to append a stabiliser term to standardise inputs to this variant algorithm; and this can be decided by computing a CE function. The worst case,complexity of the sketched variant algorithm is roughly $O(n!^{O(Ln!)})$ We note that we can derive variants of the above and the algorithm to solve Markov's problem in lots of ways.

1.4 Conclusion

This paper is a work in progress but we have presented some new results. Further work would be to show the implementation results, we have done,of our new attack. We gave an attack for the Markov problem. The Markov problem in [2] was proposed as a hard problem. As an aside we showed how to implement different algorithms to solve the knot recognition problem with certain parameters. So we gave a partial knot recognition algorithm which may be used with knot based protocols. If a protocol was designed using the Markov algorithm then it should resist the attack presented here.

References

[1] F. A. Garside. The braid group and other groups, Quart. J. Math.Oxford 78, 1969

[2] K.K. Ko, S.J. Lee, J.H. Cheon, J.W. Han, J.S. Kang, C. Park. New public-key cryptosystem using braid groups. CRYPTO 2000,LNCS, pages. 166-183, 1880,2000, .

[3] M. M.Chowdhury, On the Security of the Cha Ko Lee Han Cheon Braid Group Public-key Cryptosystem,available at www.ArXiv.com , 2007

[4] M. Chowdhury, On the security of new key exchange protocols based on the triple decomposition problem,available at www.ArXiv.com, 2006.

[5] M. Zucker, Studies in Cryptological Combinatorics, Ph.D. thesis, The City University of New York, 2005

[6] W. Haken, Theorie der Normalflachen: Ein Isotopiekriterium fur den Kreisknoten, Acta Math., 105, pages, 245-375, 96

[7] P D Bangert, M A Berger and R Prandi, In search of minimal random braid configurations,Journal of Physics A: Mathematical and General Volume 35 Number 1