On the AAGL Protocol

M. M. Chowdhury

1. Abstract

Recently the AAGL (Anshel-Anshel-Goldfeld-Lemieux) has been proposed which can be used for RFID tags. We give algorithms for the problem (we call the MSCSPv) on which the security of the AAGL protocol is based upon. Hence we give various attacks for general parameters on the recent AAGL protocol proposed. One of our attack is a deterministic algorithm which has space complexity and time complexity both at least exponential in the worst case. In a better case using a probabilistic algorithm the time complexity can be $O(|XSS(u'_i)|^{\lambda_5 n^{1+\epsilon}})$ and the space complexity can be $O(|XSS(u'_i)|^{\lambda_6})$, where the element $u'_i$ is part of a public key, $n$ is the index of braid group, $XSS$ is a summit type set and $\epsilon$ is a constant in a limit. The above shows the AAGL protocol is potentially not significantly more secure as using key agreement protocols based on the conjugacy problem such as the AAG (Anshel-Anshel-Goldfeld) protocol because both protocols can be broken with complexity which do not significantly differ. We think our attacks can be improved.

2. Introduction

Recently the AAGL (Anshel-Anshel-Goldfeld-Lemieux) key agreement protocol using braid groups has been proposed [1] an application of the AAGL protocol is for RFID tags [1]. There is an instantiation of the AAGL protocol in [1] where the AAGL protocol uses braid groups, in all of this paper we refer to the AAGL protocol when it uses braid groups. In this note we give an attack which can show the security of the protocol is based on the multiple simultaneous conjugacy search problem (see definition 1 below). We think our attack can be improved. Note once $z$ is recovered with our attack then agreed upon key can be computed with the linear algebraic attack given in [1]. All our algorithms can work in groups that are not the braid group.

2.1 Hard Problems in Non-abelian Groups.

Definition- The MSCSP (multiple simultaneous conjugacy search problem) [4] is find elements $g \in G$ such that $y_i = gx_i g^{-1}$, given the publicly known information: $G$ is a group, $x_i, y_i \in G$ with $x_i, y_i = ax_i a^{-1}$, $1 \leq i \leq u$, with the secret element $a \in G$.

Notation- We refer to an example of the MSCSP as $((x_1, x_2, ..., x_u), (y_1, y_2, ..., y_u))$ with solution $(g, g^{-1})$.

Definition- Consider the following variant of the MSCSP. If $(x_1, x_2, ..., x_u)$ is unknown in the MSCSP $((x_1, x_2, ..., x_u), (y_1, y_2, ..., y_u))$ and we are then to find the elements $g$. We refer to the above variant of the MSCSP as the MSCSPv (MSCSP-variant).

Notation- We refer an example of the MSCSPv as $((x_1, x_2, ..., x_u), (y_1, y_2, ..., y_u))$ with solution $(g, g^{-1})$. 


Definition-The CSP [4] can be defined as the MSCSP with $u = 1$.

Notation-We refer to an example of the CSP as $(x, y)$ with solution $(g, g^{-1})$.

Notation-In this paper $XSS$ refers a set that potentially contains one or more solutions for the MSCSP so $XSS$ can refer to a summit type set such as $SSS$.

The security of the AAGL protocol is based on the MSCSPv-this is shown below. Our attack is an algorithm to solve the MSCSPv. The main purpose of this paper is using an algorithm of deterministic factorial time and space complexity or a probabilistic algorithm with time complexity $O(E(u'_i)n^2)$ and space complexity $O(E(u'_i))$ ($O(E(u'_i))$ is of at least exponential complexity in the braid index $n$ and its word length $W$ of some braid, so $E(u'_i)$ grows at most, like a power of the factorial of $n, O(n!^W)$, in this paper we refer to $O(n!^W)$ as the abbreviation factorial complexity), it is then to shown the security of the AAGL protocol is equivalent to solving the MSCSP (and hence the CSP) instead of the MSCSPv.

Our result is better than all previous results in the connection: it works for general parameters, that there is only a brute force algorithm (which has factorial complexity) to solve the MSCSPv and our algorithms are better than the brute force algorithm. The above factorial complexity algorithm of ours may use any factorial time algorithm for the CSP, note the best algorithm to solve the CSP in general has in the worst case factorial running time. Hence this means the AAGL protocol is no more secure than using the AAG protocol [2] in the connections:

- We show they are both protocols are based on the MSCSP (so the AAGL protocol is not strictly based on the MSCSPv as implied in [1]).
- They can both be broken using related deterministic algorithms of factorial complexity that solve the MSCSP.
- There are related probabilistic algorithms (including our probabilistic algorithm 4) that may break both protocols depending on the parameters used.

3. AAGL Key Agreement Protocol

In the recent [1] AAGL propose a key agreement protocol it differs mainly from the seminal AAG (Anshel-Anshel-Goldfeld) algebraic protocol given in [2] because it is based on the MSCSPv, the AAG protocol is based on a system of conjugacy equations (the MSCSP) [2]. We do not reproduce all details of the AAGL protocol which can be found in [1] but restrict to the details we require. Let $B_n = \{b_1, b_2, ..., b_{n-1}\}$ be the Artin representation of the braid group on $n$ strings. In [1] an example of the protocol is given using braid groups the security is based on the TTP algorithm in [1] given below. $e$ is the identity element in the braid group.

**Algorithm 1- TTP Algorithm of [1].**

1. Choose two secret subset $BL = \{b_{l_1}, ..., b_{l_a}\}, BR = \{b_{r_1}, ..., b_{r_\beta}\}$ of the set of generators of $B_n$ where $|l_i - r_j| \geq 2$ for all $1 \leq i \leq l_a$ and $1 \leq j \leq r_\beta$.
2. chooses a secret element $z \in B_n$.
3. Choose words $\{w_1, ..., w_\gamma\}$ of bounded length from $BL$. 

4. Choose words \( \{v_1, ..., v_\gamma\} \) of bounded length from \( BR \).

5. For \( 1 \leq i \leq \gamma \)
   a. calculate the left normal form \( zw_i z^{-1} \) and reduce the result modulo the square of the fundamental braid.
   b. set \( w'_i \) equal to the sequence of integers that correspond to the element calculated in a.
   c. calculate the left normal form \( zv_i z^{-1} \) and reduce the result modulo the square of the fundamental braid.
   d. set \( v'_i \) equal to the sequence of integers that correspond to the element calculated in c.

6. Publish the two sets \( \{w'_1, ..., w'_\gamma\} \) and \( \{v'_1, ..., v'_\gamma\} \).

The security of the TTP algorithm is based on the MSCSPv with the elements \((x_1, x_2, ..., x_u) = (w_1, ..., w_\gamma, v_1, ..., v_\gamma)\), \((y_1, y_2, ..., y_u) = (w'_1, ..., w'_\gamma, v'_1, ..., v'_\gamma)\) and \( u = 2\gamma \). Assume the attacker knows this instance of the MSCSPv in Artin representation.

3.1 Security of AAGL Protocol is Based on the Multiple Simultaneous Conjugacy Search Problem

Notation-\( u'_i \in \{w'_1, ..., w'_\gamma\} \cup \{v'_1, ..., v'_\gamma\}, u_i \in \{w_1, ..., w_\gamma\} \cup \{v_1, ..., v_\gamma\}, u'_i \sim u_i \), for some \( 1 \leq i \leq 2\gamma \).

Notation-S_{u'_i} is a set that contains elements of the form \( zkz^{-1} \) when the set is used in a deterministic algorithm. \( S_{u'_i} \) contains elements of the form \( zkz^{-1} \) with some probability when the set is used in a probabilistic algorithm. Where \( zkz^{-1} \) are elements in the centraliser of \( u'_i \).

Recall the centraliser of an element is the set of all elements that commute with it, for the infinite braid group the centraliser of an element will contain an infinite number of elements hence we approximate the centraliser with a finite set. Let \( \lambda \) be a braid invariant, it is possible (but unlikely) for two different braids have the same value for \( \lambda \) see [6]. Note there are practical algorithms to compute the braid invariant \( \lambda \) because the CDP (conjugacy decision problem) is feasible in braid groups. Good bounds for summit type sets are not known but the SSS certainly has the upper bound \( n!q \), where \( q = \max_i (\min \sup(u'_i) + \max \inf(u'_i)) \). For example see [6], so all known algorithms for computing SSS are in the worst case have factorial complexity but it is conjectured that the size SSS is exponential in \( n \). Because we use the upper bound \( n!q \) our algorithm is of factorial complexity. It is known the quantity \( \min \sup(u'_i) + \max \inf(u'_i) \) can be computed in polynomial time and space so \( q \) can be computed in polynomial time and space in \( n \) and the length of \( u'_i \) (obviously this means \( q \) can be computed in factorial space and time complexity). Refinements of Garside’s algorithm for the CSP/CDP [8] (Garside’s is the first algorithmic solution of CDP/CSP) to solve the CSP such as the solution given in [7] can be used to solve deterministically the CDP in worst case factorial time and space complexity. Elements of the form \( zkz^{-1} \) (where \( k \) commutes with \( u_i \)) may be found by computing the centraliser of \( u'_i \) (which any attacker
can compute) because \(zk^{-1}zuiz^{-1} = zuiz^{-1}zk^{-1} \Rightarrow zk^{-1}u_i' = u_i'zk^{-1} \) so \(zk^{-1}\) is in \(S_{u_i'}\).

### 3.1.1 Attack Based on MSCSP

The only known attack, given in [1], without side information on the AAGL is a brute force attack the above linear algebraic attack is given in section 6 of [1]. We give a deterministic algorithm based on computing centralizers to solve the MSCSPv and our algorithm maybe uses algorithms that compute super summit sets. Our algorithm has factorial complexity some reasons are because all known algorithms to compute the centralizer of an element are factorial complexity (in the worst case)/the best known algorithm to solve the CSP in general has factorial running time, which means computing centralisers and solving the CSP can take around the same time. Hence our algorithm is the best known way to attack the AAGL if suitable parameters can be found and potentially its efficiency improved.

**Notation-C** is an algorithm that computes \(S_{u_i'}\) in factorial space and time complexity in a worst case.

**Algorithm 2 General deterministic algorithm for MSCSPv.**

1. Compute \(S_{u_i'}\) for \(u_i' = zuiz^{-1}\), \(S_{u_i'}\) contains some or all elements of the form \(F = zk^{-1}\) it follows choices for \(k\) includes all elements which commute with \(u_i\) etc., \(S_{u_i'} \subseteq S_F\).
2. For an element \(u_i'\) in \(S_{u_i'}\) find \(k\) then solve the CSP with \((k, zk^{-1})\) for \((z, z^{-1})\). We find \(k\) as follows.
   2i. Select a function \(f_P\) which parametrizes in \(P\) a finite approximation to the centralizer \(u_i'\).
   2ii. Select a function which parametrizes in \(L_P\) a finite approximation to words in \(G\). We define by the set \(U_{L_P}\) as containing all words defined by \(L_P\).
3. Set \(L_P = L_0\). \(P = P_0\). \(L_P\) may depend on \(P\). Compute if necessary \(S_{u_i'} = f_P\).
   3ii. Update value \(P\) as, \(P \in p_v\). Initialise \(I = I_0\).
   3iii. Select \(S_{u_i', L_P} \subseteq S_{u_i'}\) compute if necessary \(S_{u_i'} = f_P\). \(S_{u_i', L_P}\) may depend on \(L_P\). Using a chosen algorithm, find a (CSP) pair \((b, a)\) such that

\[
b \sim a, \ a \in S_{u_i', L_P}, \ b \in U'_{L_P}
\]

where \(U'_{L_P} \subseteq U_{L_P}\). The pair \((b, a)\) is stored.
4. If the values of \(P\) have been exhausted from the set \(p_v\), then goto step 4.
3v. Update value of \(I\) as \(I \in i_v\), if the values of \(I\) have been exhausted from the set \(i_v\), then goto step 3ii.
3vi. Update \(L_P\) as \(L_P = L_{P,I}\). If the values of \(L_P\) have not been exhausted goto step 3iii.
4. Solve the MSCSP for all the pairs \((b, a)\). Terminate algorithm.
We now give an example of algorithm 2 which is also a general algorithm (note $P$ is redundant in this example for the reason given in the proof below) of the above algorithm where $G$ may be $B_n$ or any Garside group.

Algorithm 3 An example of algorithm 2.

1. Compute $S_{u'_i}$ for $u'_i = zu'_iz^{-1}$, $S_{u'_i}$ contains some or all elements of the form $F = z^{k}z^{-1}$ hence for choices of $k$ includes all elements in $BR$ or $BL$ depending if $u'_i = v'_i$ or $u'_i = u'_i$.

   A second possible choice is to compute $S_{u'_i}$ represented by a generating set of the centraliser of $u'_i$ such as using the algorithm in [9].

2. Find $k$ then solve the CSP with $(k, z^{k}z^{-1})$ for $(z, z^{-1})$. We find $k$ as follows.

   2i. Select a function which parametrizes in $P$ a finite approximation to the centralizer $zu_i z^{-1}$. We choose to the function $F_{u'_i, P}(P_0, \alpha)$ which computes the set which contains all braids $F \in SF$ in the centralizer of $u'_i$ such that $\Delta^P \leq F \leq \Delta^{P+1}$, for $\forall P$, $P_0 \leq P < \alpha$, which is $\Delta^{P_0} \leq F \leq \Delta^{\alpha}$. $SF$ here contains at least one element in the centraliser of $u'_i$ if using $C$ as described in the proof below.

   2ii. We define the set (we construct) $U_{L_P}$ as $U_{L_P} \subseteq B_n^+ \setminus 1$ to contain all distinct words in Artin of length $L_P$ with the length is in the number of Artin generators. Or a second possible choice for $U_{L_P}$ may contain some or all of the union of the centralisers of short words in Artin generators, so for example, to compute $k$ is to (where $k$ may be long) choose it from the generating set of the centraliser of the single Artin generators $\sigma_i$ say using the algorithm in [9], the above approach can be used when $k$ not one Artin generator.

   3. Set $L_P = 1$. $P_0 = -2g$, $P = P_0$. Let $S_{u'_i} = F_{u'_i, P}(P_0, \alpha)$.

   3i. $P = P + 1$. $I = 1$.

   3ii. We test the relation using an algorithm for the CDP (an alternative step instead of this step is described in the proof below)

   $$\lambda(a) = \lambda(b), \ a \in S_{u'_i, L_P}, \ b \in U'_{L_P}$$

   where $S_{u'_i, L_P} \subseteq S_{u'_i}$ and $U'_{L_P} \subseteq U_{L_P}$. The pair $(b, a)$ are found with a linear search. If the above relation is true then let $k = b$. The pair $(b, a)$ is stored.

   3iv. If $P > P_0 + 1$ then goto step 4.

   3v. $I = I + 1$. $L_P.I = I$.

   3vi. $L_P = L_P.I$. If $L_P > f(u'_i)$ then goto step 3ii. Where $f(u'_i)$ may depend on $u'_i$.

4. Solve the MSCSP for all the pairs $(b, a)$ using a deterministic algorithm. Terminate algorithm.

As would be expected, for poorly chosen parameters our algorithm may not be more efficient than a brute force attack. A potential variant is to check if a short word in length of Artin generators is not conjugated by $z$ an attacker can compute then length for a given length function the average (or an upper bound) length of $u'_i$ and if a word $u'_i$ is significantly larger than the average
length (or an upper bound) of \( u'_i \) it is considered not a potential value for \( a \), so here \( S'_v \) is the subset \( S'_v \) in the bounds for \( F \) such that \( \Delta^{P_0} \lesssim F \lesssim \Delta^{\delta} \) (so here \( P \) may have a larger range), for some integer \( \delta \), and not restrict \( b \) to positive words.

The parameters \( P, L_P \) control the lengths of \( a \) and \( b \). We now show for suitable parameters our attack will terminate with a solution for the MSCSPv used in the AAGL protocol.

**Proposition 1**

Solving the MSCSPv as used in the AAGL protocol is equivalent to solving MSCSP (which can be shown in deterministic factorial time) using algorithm 3 twice and possibly using algorithm \( C \) with the parameters \( P_0 = -2g_z, \alpha = 2g_z + L_m, f(u'_i) = O\left(\frac{n}{\log(\ell)}\right) \) where \( L_m = \max_v P_P \). As is shown in the proof \( L_m = 1 \) is sufficient. This requires in the worst case, space complexity

\[ O(c_1|SSS(u'_i)| + c_2|SSS(b)| + (n - 1)^{O\left(\frac{1}{\log(\ell)}\right)}) \]

and time complexity

\[ O(|SSS(b)||SSS(u'_i)|n^{-1})^{O\left(\frac{1}{\log(\ell)}\right)}) \]

where the element \( u'_i \) is part of the TTP’s public key and \( b \in S'_v \cup S'_w \).

**Proof**

We use algorithm that computes summit type sets we call XSS we analyze the cases for XSS, SSS and C both in the proof below.

- Case using C and SSS.

For C we use an existing algorithm for the centraliser or the CSP. \( \inf_{ss}(u'_i) \) means elements of the SS which have maximum infimum of the conjugacy class of \( u'_i \). For example we use the algorithm to compute the SS [8] to solve the CSP \( (u'_i, u'_i) \) then it follows for all \( P \) used or the choice for computing the SS can be computed in worst case factorial complexity, so \( P \) is redundant in this case, but here \( \inf(zk^{-1}) = \inf_{ss}(u'_i) \) must be true for the algorithm to find a solution. Another choice for \( S'_v \) involves computing all braids \( F \),

\[ \Delta^{P_0} \lesssim F \lesssim \Delta^{\alpha} \tag{1} \]

then it would follows from the analysis and the bound on the number of braids in canonical factors and the braid index [3] our algorithm would be of factorial complexity but this choice of parameters for the algorithm results in complexity similar to a brute force algorithm. The above method using 1 can be potentially improved for example if \( k \) (described below) is short length, then depending on the method of rewriting elements, \( F \) is expected of short length hence the value of \( \alpha \) can be lowered.

In the following analysis we assume the algorithm C is used to compute \( F \in S'_v \), with \( F \) in the bounds given by 1, for all \( P \) used or the choice for computing
above or we assume our analysis below we use the SSS based algorithm of [7] which has worst case factorial time and factorial space complexity. If $C$ computes $S_{u'}$ contains at least one element of the form $zkz^{-1}$ is used then our algorithm always terminates with a solution using the bounds for $P_{0, α}$ derived below.

We prove below that computing $U_{L′}$ has worst case factorial time and factorial space complexity. Note if it is true at least for a class of braids for a value of $XSS$ that $|XSS|$ is exponential in $n$ in the proof below then the algorithm works in worst case exponential time (hence the term $O(\frac{n}{\log(n)})$ appears below).

From [1] any attacker can compute the smallest $g_z$ from

$$g_z \geq m$$

where $g_z$ is the length of $z$ in it Artin generators and we assume the smallest $g_z$ is used. If the above assumption turns out to be false then the attacker may estimate $g_z$ from the elements $u_i'$, $g_z$ can be feasibly computed otherwise the public keys are too long in Artin generators to use.

We use the easy theorem 1.5 given in [7] which is if $B$ is any braid word represented in $N$ negative Artin generators $P$ positive Artin generators and then $\Delta^N \ll B \ll \Delta^P$. Consider then for any $b \in U_{L′}$, and $z$ then $\Delta^{-g_z} \leq z \leq \Delta^{g_z}$, $e \ll b \ll \Delta^{L′}$, hence

$$\Delta^{-2g_z} \ll zbz^{-1} \ll \Delta^{2g_z+L′}$$

hence if we let $P_0 = -2g_z$, $α = 2g_z + L′ < 2g_z + L_m$ from the above bound on $zbz^{-1}$ then it must be true that the centraliser of $u_i'$ contains elements of the form $zkz^{-1}$ where $k$ is an element in $BL$ or $BR$.

Note $|S_F|$ by assumption is of factorial complexity in the parameters $2g_z + L′$, $n$ and contains $zkz^{-1}$ for values of $k$.

This is true because we know from the condition $|l_i - r_j| \geq 2$ it follows that $BL$ and $BR$ do not have any generator in common.

Now the attacker runs the algorithm 3 twice in parallel so that $u_i' = v_i'$ or $u_i' = w_j'$ in each the runs but the attacker may compute the common computations (such as computing $U_{L′}$) once in each of the runs. Hence for $L = 1$ one of the choices of $b$ (selected by the attacker) is one of the Artin generators of $BR$ must be a correct choice, generally this means for $k$ there are $l_n + r_β$ easy to guess when it is of feasibly computable length from the parameters suggested in [1]. Such easy choices for $k$ exist because the TTP algorithm specifies the subgroup in terms of single Artin generators. The attacker using linear search algorithm, through $S_{u_i'} \cup S_{w_j'}$ for $zkz^{-1}$ and using the CDP with $b$ a word in a single Artin generator of up length $n$ in step 3iii finds the CSP pair $(b, zbz^{-1})$ to solve which must exist by construction. An option at step 3iii is to solve the CDP in deterministic factorial time using the algorithm in [7].

We estimate $|U_{L′}|$ with the upper bound $S_{n,L′} = (n-1)^{L′}$ hence $S_{n,α}(\frac{n}{\log(n)})$ is at worst an exponential function. Hence computing all distinct words (which are not optimally bounded above exponentially by $S_{n,L′}$) of length $\theta = O(\frac{n}{\log(n)})$
from $U'_{L'}$ will mean the attacker is guaranteed to find all the words (there is an exponential amount of these and if $\theta = O(\log(n))$ they are a polynomial long) in $BL, BR$ of length $\theta$ in $S_{\omega'} \cup S_{\omega'}$. Hence the attacker can take $1 \leq L_p \leq O(\frac{n}{\log(n)})$, as this keeps the complexity exponential so the attacker selects $L_p$ up to $O(\frac{n}{\log(n)})$ (actually the attacker can select any $\theta$ such that the complexity of the algorithm is factorial in the worst case). To get more conjugacy equations the attacker can try for $b$ all words of length $O(\frac{n}{\log(n)})$ but as expected the longer the word length of $b$ is the less chance (as described below) that a CSP pair will be found but for short word length of $b$ there is a non-negligible probability that the attacker can guess a correct $b$. We show below that $1 \leq L \leq O(\log(n)$ can be chosen.

In the following $\theta = O(\frac{n}{\log(n)})$. Let $c_1, c_2, ..., \lambda_1, \lambda_2, ... \in \mathbb{R}$, we assume we may approximate $S_{\omega'}$ and centraliser computations by $O(|XSS(u'_i)|)$ this assumption is based on the fact that in some cases, e.g. $XSS = SSS$, it is known algorithms to computing the centraliser of $u'_i$ are proportional in space and time complexity to $|XSS(u'_i)|$. $|S_{\omega'}|$ and hence any centraliser is at most of space factorial complexity using (2.1) above. $SSS(u'_i)$ is the super summit set of the element $u'_i$ the size of these sets are not fully known it is known that $|SSS|$ to be at least exponential in $n$, for a fixed $n$ is proportional to $(n!)^q$, we write $SSS(u'_i)$ for the maximum size of $SSS(u'_i)$ where $q = \max_{u'_i} \min \sup(u'_i) + \max \inf(u'_i)$. Observe $O((n!)^q)$ is of smaller order than $O(e^{qn \log(n)})$ we use a similar notation for $XSS(u'_i)$.

If we use an algorithm that stores at least all the elements in $S_{\omega'} \cup S_{\omega'}$ and stores all elements in $U'_{L'}$, uses an deterministic algorithm to solve both the MSCSP (the number of equations $\nu$ in the MSCSP may be constant) and CDP that uses exponential space, then the space complexity of the algorithm is factorial in the worst case it is

$$O(c_1 E(u'_i) + c_2 E(b) + c_3 (n-1)^\theta)$$
$$= O(c_4 |SSS(u'_i)| + c_5 |SSS(b)| + (n-1)^O(\frac{n \log(n)}{\log(\log(n))}))$$
$$\approx O(c_6 e^{q \log(n)} + c_7 e^{q_2 \log(n)} + c_8 \omega n)$$

where the constant $\omega$ depends on the function used for $O(\frac{n}{\log(n)})$.

Note the space and time complexity of solving the MSCSP is proportional to $\nu|SSS(b)|$ if the MSCSP solved by intersections of elements of summit sets.

We write $O(|SSS(b)|) \approx O(E(b)) = O(e^{q_2 n \log(n)})$ as the maximum size of $SSS(b)$ which is determined by $q_2 = \max_{b \in U'_{L'}} \min \sup(b) + \max \inf(b)$, so $q_2$ is the canonical length of $b$.

We use this result at step 3iii in the CDP. Computing $S_{\omega'} \cup S_{\omega'}$, the linear searches through $S_{\omega'} \cup S_{\omega'}$ for each element of $U'_{L'}$ and solving the CDP for every potential pair at 3iii, and then solving the resulting MSCSP means in the
worst case the time complexity is factorial and it is

\[
O(c_1 E(u'_1^i) + c_2 E(u'_1^i) E(b)(n - 1)^d) \\
\approx O(E(b) E(u'_1^i)(n - 1)^{O(\log(n))}) \\
= O(E(b) E(u'_1^i) \omega^n) = O(|SSS(b)| SSS(u'_1^i) |\omega^n) \\
\approx O((\omega e^{(q+q_2) \log(n)})^n)
\]

It is understood the constants \(c_1, c_2, \ldots\) are different from the constants \(c_1, c_2, \ldots\) used in the notation of the space complexity and other complexity computations below. Now consider a variant of the above attack, which is not use an algorithm for the CDP in step 3iii but instead solves the CSP with the guess for \(b\) with every possible element in \(S'_i, L_P\) and recovers \(z\) and hence the shared secret using the algorithm in [1], the attacker may test if \(z\) is the correct solution: for example, \(z\) is used in an impersonation attack or if (using an algorithm for the CDP) \(z^{-1}u'_1^i z \sim u'_1^i\). For the variant attack above the worst case space and time complexity are the same.

- XSS Case.

In the more general case of XSS we can use any algorithm we refer to as \(S\) and outputs for a conjugate pair \((b, a)\) or potential pair \((b, a)\) using the sets \(S'_i, S'_j, U_L\), let \(S'_i, S'_j, S_{a b}^i\) be the time complexity and space complexity respectively to compute \(S'_i, S'_j\), we refer to the time complexity and space complexity to solve the CDP as \(CDP_i\), \(CDP_j\), we refer to an algorithm for \(CDP\) with input \(b, a\) as \(CDP(b, a)\) similar notation for \(MSCSP\). Note it is assumed the CDP and MSCSP can be solved in worst case time complexity and space complexity proportional to \(|XSS|\), this is true for example when \(XSS = SSS\) and the assumption is based on this example. By a similar argument to the SSS a time complexity bound in the worst case is,

\[
O(S_i) + S_j + \max_{P \in P_{a b} b \in U_L} CDP_i(S(S_i \cup S_j), b) + MSCSP_i
\]

A bound for the space complexity in worst case is

\[
O(S_i, S_j, U_L) + MSCSP + \max_{P \in P_{a b} a \in S_i \cup S_j} \max_{b \in U_L} CDP_i(b, a)
\]

\[
\approx O(c_1|XSS(u'_i)|^{\lambda_1} + c_2|XSS(b)|)
\]

Now consider a variant of the above attack, which is not use an algorithm for the CDP in step 3iii but instead solves the CSP with the guess for \(b\) with every possible element in \(S'_i, L_P\) and recovers \(z\) and hence the shared secret using the algorithm in [1], the attacker may test if \(z\) is the correct solution: for example, \(z\) is used in an impersonation attack or if \(z^{-1}u'_1^i z \sim u'_1^i\).

Note it may be that \(b \in S'_i, L_P\) (which can be verified using a polynomial time word algorithm in \(B\)), in this case \(z\) must be in the centraliser of \(b\), call
the set of all such stored $b, B_z$ and so $z$ can be found by testing every element (for the choice of $z$) of the centraliser of a subset of $B_z$ for the correct element.

If $S_{u'}$ is computed using the second choice in step 1 and the corresponding value of $k$ is found using the second choice in step 2i then because it is known the centraliser of every element in $B_n$ can be generated by $O(n^2)$ generators hence the MSCSPv can be solved feasibly depending on $|SSS|$ [9].

Algorithm 4- Probabilistic algorithm for MSCSPv.

1. Compute $S_{u'_i}$ a suitably small $S_{u'_is}$ of $u'_i = zu_i z^{-1}$ that may contain elements of the form $F = zk z^{-1}$ hence for choices of $k$ includes all elements in $BR$ or $BL$ depending if $u'_i = v'_i$ or $u'_i = w'_i$. One possible simple choice at this step is to compute $S_{u'_i}$ as randomly chosen elements of the centraliser of $u'_i$.

2. Find $k$ then solve the CSP with $(k, zk z^{-1})$ for $(z, z^{-1})$. We find $k$ as follows.

2i. Select a function which parametrizes in $P$ a suitably small finite approximation to the centralizer $zu_i z^{-1}$. We choose the function $F_{u'_i, P}(P_0, \alpha)$ which computes the set which contains a subset of the braids $F \in S_F$ and if possible maybe using heuristic method(s) gives $F$ where $k$ is short (in a given length function) with high probability.

2ii. We define the set we can feasibly compute $U_{L_P}$ as $U_{L_P} \subseteq B_n$ of short words in length $L_P$ for some length function.

3. Set $L_P = 1, P_0 = -2g_z, P = P_0$. Let $S_{u'_i} = F_{u'_i, P}(P_0, \alpha).

3iii. We test the relation using an efficient algorithm to solve the CDP such as the one in [6]

$$\lambda(a) = \lambda(b), \ a \in S_{u'_i, L_P}, \ b \in U_{L_P}'$$

where $S_{u'_i, L_P} \subseteq S_{u'_i}$ and $U_{L_P}' \subseteq U_{L_P}$. If the above relation is true then let $k = b$. The pair $(b, a)$ is stored. When enough pairs have been computed goto step 4.

3iv. $I = I + 1$.

3v. $L_{P, I} = I$.

3vi. $L_P = L_{P, I}$. If $P > P_0$ then terminate then goto 4. If $L_P > f(u'_i)$ then goto step 3ii. Where $f(u'_i)$ may depend on $u'_i$ we can take it to be on feasibly computable words up to $O(\frac{n}{\log(n)})$.

4. Solve the MSCSP for all the pairs $(b, a)$ using an algorithm that works with high probability. Terminate algorithm.

Proposition 2

Solving the MSCSPv can be done with the probabilistic algorithm 4, in approximately, time $O(|XSS(u'_i)|^{\lambda_4} \omega^n)$ and space $O(c_3 |XSS(u'_i)|^{\lambda_4} + c_1 \omega^n)$, and with additional reasonable assumptions this can be improved to time $O(|XSS(u'_i)|^{\lambda_5} n^{1+\epsilon})$ and space $O(|XSS(u'_i)|^{\lambda_6})$ using algorithms to solve the MSCSP, CDP efficiently, where the element $u'_i$ is part of the TTP’s public key, $XSS$ is a summit type set and the constant $\omega$ depends on $n$. 

10
Proof

The following easy computations involved in computing the complexity in the better case: a randomly chosen generator has probability $p_\alpha = \frac{l_\alpha}{n-2}$ and $p_\beta = \frac{r_\beta}{n-2}$ of being in $BL$ and $BR$ respectively, then an attacker can selects a random word $b$ from $U'_{L,P}$ using in length of $\theta$ Artin generators then it has $p_{\alpha,\beta,\theta} = \frac{1}{p_\alpha} + \frac{1}{p_\beta}$ probability of being in $BL$ or $BR$. From the algorithm used to compute $S_{u'_i}$ (which computes words less than a certain bound) we do not have to pick $\theta$ too large there exist some $k$ of short length in Artin generators. Choosing $\theta \leq n$ as this keeps the algorithm in factorial complexity but this is not a good choice, from the above discussion the attacker can take $\theta = O(n \log(n))$. Observe the attacker must on average compute

$$[WP_{\alpha,\beta,\theta}] < \frac{(n-2)^\theta}{\min(l_\alpha, r_\beta)^\theta}$$

before expecting $W$ words to be found in $S_{u'_i} \cup S_{w'_j}$. The attacker may estimate $p_\alpha, p_\beta$ if the attacker assumes $l_\alpha \approx r_\beta$ and assumes all (then $p_{\alpha,\beta,1} = 1$ for the selection of $b$ used in both runs) or nearly all possible Artin generators are used in $BL, BR$ so $p_\alpha \approx p_\beta$. Hence (independent of large enough $n$), the attacker needs to compute approximately as few as $2^\theta$ distinct words for the parameters suggested in [1] to ensure on average a reduction to the MSCSP with at least $2$ equations. We would need to select only approximately $4$ distinct random words of length $3$ from $U'_{L,P}$ before the attacker expects to get one conjugacy equation or the CSP, the example above use little memory and potentially little computing power.

In this better case we use an efficient algorithm for the CDP such as the one given in [6], use a linear search, and use an algorithm for the MSCSP that works with high probability. We assume in all of this second proof, the length of $b$ is less than $u_i$, this means in general $O(|SSS(u_i)|)$ is greater than $O(|SSS(b)|)$.

We assume in this proof there is an algorithm that can compute $S_{u'_i}$ proportional in space and time complexity to $|XSS|$, this assumption is based on the fact that such an algorithm exists when $XSS = SSS$ see [9], and that this algorithm has worst time space and time exponential complexity. From the description above (from $[WP_{\alpha,\beta,\theta}] < O(n^{O(\log(n))})$), the time complexity in this better case is $O(|XSS(u'_k)|^{\lambda_\theta} \omega^n)$ as shown below. We assume we use an algorithm for the CDP which has time and space proportional to $|XSS|$ such as Garside’s algorithm [8]. Here $CDP_t$ is the average time taken to solve the CDP over all pairs $(b, a)$. Then by a similar argument to above the time complexity is

$$O(S_{t,v'_i} + S_{t,w'_j} + CDP_t(|S_{u'_i}|) + |S_{w'_j}|)\omega^n + MSCSP_t$$

$$\approx O(E(u'_i)O(n^{O(\log(n))}))$$

$$= O(|XSS(u'_k)|^{\lambda_\theta} \omega^n)$$
also the space complexity is

\[ O(S_{s,v_i} + S_{s,w_j} + c_1\omega^n + \max_{\forall P \in \mathcal{P}, a \in S_{s,v_i} \cup S_{s,w_j} \ b \in U_L} \max_{b \in U} \max \ CDP_s(b, a) + MSCP_s) \]

\[ = O(c_2 E(u'_i) + O(n^{O(\log(n))})) \]

\[ \approx O(c_3 |XSS(u'_i)|^{\lambda_4} + c_4\omega^n) \]

where the constant \( \omega \) depends on the function used for \( O(n^{\log(n)}) \).

• Better Complexity Bounds

The above analysis for the better case is may not be optimal, for example if we make some assumptions then we get a tighter bound on the complexities as follows. For this case the average complexities for algorithm for the CDP, CSP and MSCSP are considered. From the from 2 and if we assume \( l_\alpha, r_\beta \) are linear in \( n \) and \( \theta = O(\log(n)) \), we assume we have an efficient algorithm for the CDP which has average linear complexity possibly the one given in [5], this assumption is based on the result that empirically for randomly chosen long random braids which have simple elements randomly chosen the \([USS]\) is on average likely to be linear in the word length and independent of the braid index \( n \) e.g. see [5], and use an algorithm for the MSCSP that works with high probability. Hence the CDP/CSP in this average case can be solved in linear space and time complexity. The time complexity in this better case can be with high probability be (recall computing \( S_{u'_i} \) that in proportional in space an time complexity to \( |XSS| \))

\[ O(c_1 E(u'_i)^{\beta n^{O(\log(n))} O(n)}) \approx O(E(u'_i)n^{1+\epsilon}) \]

\[ = O(|XSS(u'_i)|^{\lambda_5 n^{1+\epsilon}}) \]

for some \( \beta \in \mathbb{R} \) which depends on 2. The space complexity in this better case is

\[ O(c_1 E(u'_i) + B^{O(\log(n))} + S_{s,v_i} + S_{s,w_j}) \]

\[ \approx O(c_1 E(u'_i) + c_2 n^{\epsilon} ) \approx O(|XSS(u'_i)|^{\lambda_6}) \]

using straightforward algebra it can be shown \( \epsilon \) can be close to a constant as \( n \) is larger and depends on \( \beta \) and the constants in \( O(\log(n)) \), if \( \beta \) is bounded then \( \epsilon \) is bounded.

Note the space can be up to exponential size (so giving a better space bound here) the only requirement is the set \( S_{u'_i} \) must be of at least size greater than one as it must contain at least one element with some feasible computable \( k \).

The above shows using the AAGL protocol can potentially be as secure than using CSP based protocols such as the AAG protocol [2] as both can be broken with attacks of the same or similar complexity depending on the values \( \lambda_5, \lambda_6 \) and \( \epsilon \), by similar we mean our instantiations of our attack can differ by a factor of a polynomial in \( n \) from attacks such as on the AAG protocol, for example the time complexities of attack differ by a factor of \( n^{1+\epsilon} \). If the attacker decides to compute \( S_{u'_i} \) as randomly elements chosen elements of the centraliser.
of $u'_i$ then the success of this attack in this case depends on the probability of $S_{u'_i}^L$ containing elements of the form $zkz^{-1}$. Informally, the attack consists of computing subset of centralisers and extracting suitable elements from the centralisers: we refer to our attack as a two central element attack.

Now consider a variant of the above attack, which is not use an algorithm for the CDP in step 3iii but instead solves the CSP with the guess for $b$ with every possible element in $S_{u'_i}^{L,P}$ and recovers $z$ and hence the shared secret using the algorithm in [1], the attacker may test if $z$ is the correct solution: for example, $z$ is used in an impersonation attack or if $z^{-1}u'_iz \sim u'_i$.

Note it may be that $b \in S_{u'_i}^{L,P}$ (which may be verified using a polynomial time word algorithm in $B_n$), in this case $z$ must be in the centraliser of $b$, call the set of all such stored $b$, $b_z$ and so $z$ can be found by testing every element of the centraliser of a subset of $b_z$ for the correct element.

Observe if the attacker assumes his guess of the generators of $BL, BR$ are correct (or manages know these subgroups in a different way) the attacker can compute randomly chosen words computable in polynomial time in $BL, BR$ and in up to factorial time (in approximately the time taken to solve the CDP) find a system of conjugacy equation / reduce the security of the AAGL protocol to the MSCSP so this is another reason why the users should keep the subgroups $BL, BR$ secret. For general $BR$ and $BL$ the algorithm has to be modified to use the publicly known information about their structures. The complexity of the algorithm is mainly determined by computing $S_{u'_i}^{L,P}, S_{u'_j}^{L,P}$ which may contain portions of the centralisers, so we can estimate this to be approximately the same time and space complexity of computing the $SSS$ of an element so in general it is exponential, also the size of the sets $S_{u'_i}^{L,P}$ and $U_{L,P}^{'}$ affect the complexity of the example algorithm in this connection $O_1$:

$$O_1 = \sum_{P, \forall \ell \in \ell_0, \forall \ell' \in \ell_0} |S_{u'_i}^{L,P,1}| + |U_{L,P,1}'| + |p_v||i_v|$$

$$= \sum_{P=\ell_0}^{=2g_m} \sum_{I=1}^{f(u'_i)} |S_{u'_i}^{L,P,1}| + |U_{L,P,1}'| + f(u'_i)$$

will make the example algorithm can be used as a parameter to measure the efficiency of algorithm 3, minimising $O_1$ will make the algorithm more efficient. Generally in our probabilistic algorithm we could use an heuristic optimization algorithm instead of a linear search if we do this then we suggest trying the differential evolution algorithm because it is known to be fairly fast and reasonably robust [12]. The components of the vectors used in the differential evolution algorithm depending on $L$ and $L_P$ the differential evolution means in general the components of the trial vector will not increase linearly so this means in steps 3i, 3iv will not be increased linearly as is done in the probabilistic algorithm.

Algorithm-5 To Recover $BL$ and $BR$.

With a little more work we give an attack that recovers the secret subgroups $BL$ and $BR$. Any attacker can compute for $i, j$ for sufficiently many $i$ and $j$.
using the attack above (to recover $z$) $v_i, w_j$ the attacker checks for the generator $b_r$, $1 \leq r \leq n$ if

$$w_ib = b_ww_i \text{ for all } i \quad (3)$$
$$v_jb = b_nv_j \text{ for all } j. \quad (4)$$

If 3 is true then $b_r$ is a generator of $BR$ similarly if 4 is true then $b_r$ is a generator of $BL$.

Algorithm 6-To modify our attack to solve the general MSCSP$_v$.

1. For the MSCSP$_v$ $((v_1, v_2, \ldots, v_u), (v'_1, v'_2, \ldots, v'_u))$ compute a suitable finite approximation $Z$ of the centralisers of the set of elements $(v'_1, v'_2, \ldots, v'_u)$.
2. Find elements in $Z$ such that the elements are conjugated by $g$ (we refer to such elements as the system of conjugacy equations $((w_1, w_2, \ldots, w_u), (w'_1, w'_2, \ldots, w'_u))$ such that the sets $(v_1, v_2, \ldots, v_u)$, $(w_1, w_2, \ldots, w_u)$ are commuting.
3. Solve the MSCSP$_v$ using a version of algorithm 2 with the pair of MSCSP$_v$ $((v_1, v'_1), (v_2, v'_2), \ldots, (v_u, v'_u))$, $((w_1, w'_1), (w_2, w'_2), \ldots, (w_u, w'_u))$.

Here $v_i$, $w_i$ are chosen from the subgroups $B_L$ and $B_R$ respectively. Note if the structure of $B_L$ and $B_R$ is known then this may be used in our deterministic algorithm. Combinations of centraliser elements and their inverses of the conjugated generators may be computed to attempt to construct shorter words $k$. In an example of the above algorithm it may feasible to compute one or more of the elements $v_1, v_2, \ldots, v_u$ possibly using the relation $\lambda(v'_i) = \lambda(\tilde{v}_i)$ where $\tilde{v}_i$ is the guess for $v_i$, and hence reducing to the MSCSP there.

3.2 Defending Against Attacks

The attacks may be avoided if

1) Ensure if possible that elements of the centralisers of $u'_i$ are hard with the CSP $(k, zkz^{-1})$.
2) Ensure if possible that elements of the centralisers of the form $zkz^{-1}$ of $u'_i$, that the element $k$ cannot be feasibly computed.
3) To maximize the value $O_1$, with the constraint of making MSCSP$_v$ as difficult as possible for the attacker.
4) The TTP algorithm may be modified with different choices for $BL, BR$ so that larger generators are used with the constraint of the computing platform.
5) The security of the AAGL protocol is based on the complexity of algorithm $C$ not being efficient and $C$ may be based on the following problem which is problem 1 given in [10],

$$\text{given } g_1, \ldots, g_k \in G \text{ compute } C(g_1, \ldots, g_k)$$

where $(g_1, \ldots, g_k) = (u'_1, \ldots, u'_u)$ and $C(g_1, \ldots, g_k) = C(g_1) \cap C(g_2) \cap \ldots C(g_k)$.

4. Potential Length Based Algorithm for MSCSP$_v$

We show that modified a known basic length attack for example see [11] can be used to for the general MSCSP$_v$ and then any algorithm can be used to solve
the MSCSP such as a known length attack such as [11]. We refer to our length based MSCSP algorithm the length-MSCSPv algorithm. Suppose we are given an example of the MSCSPv.

Compute the centralizer of $zw_i$ or a portion of this set we call $S_{w_i}$ then $S_{w_i}$ contains all elements of the form $F = zkz^{-1}$ it follows choices for $k$ includes all elements in $BL$ for a suitable approximation of the centraliser. Hence the generators we peel from $F$ in our length attack are the generators of the centralisers of $w_i$. For example in $B_4$ one of the generators of the centraliser of the element $\sigma_1^3 \sigma_2 \sigma_3$ is $\sigma_1^3 \sigma_2 \sigma_1 \sigma_2^{-2} \sigma_1^{-3}$ and the above generator is of length 10.

Algorithm 7-Length-MSCSPv.

Run step 1 and 2 of algorithm 6 for step 3 use the algorithm below instead of a version of algorithm 2.

Compute $r_i' = z r_i z^{-1} z t_i z^{-1} z r_i^{-1} z^{-1} = z r_i t_i r_i^{-1} z^{-1}$

where the words $r_i$ and $t_i$ are a word in the generators $w_i$. Note an element of the form $z r_i z^{-1}$ may be used instead for $r_i'$.

1. Select a length function $l$. Construct $r_i'$ as a word in the generators $w_i'$ for some $1 \leq i \leq n_w$. $A$ is set to the identity element. Set the iteration $n$ to zero.

2. Select suitable elements $s_n \in C_r$. If $r(r_1',\ldots,r_{n_w}',s_n) \preceq r(r_1',\ldots,r_{n_w}',e)$

where $\preceq$ is a linear ordering (or an objective function) on a vector of real numbers [11] and each element of the tuple $r(r_1',\ldots,r_{n_w}',s_n)$ except the last is given by the corresponding number $l(s_n^{-1} r_i')$.

3. Update the word $A$ as

$$A_{n+1} = A_n s_n.$$

The algorithm stops at this part when depending on $r(r_1',\ldots,r_{n_w}',s_n)$ and the stopping criteria (there can be more than one stopping criteria) then goes to step 6. The algorithm stops with some probability $\rho$ with $A = z \tilde{r}_i = \tilde{r}_i$ and this braid is stored, where $\tilde{r}_i = r_i C_i$ for some $C_i \in G$, in other words $A$ is the product of $z$ and a partial factor of $r_i$ we call $\tilde{r}_i$.

4. Update the element $r_i'$ as

$$r_i' = s_n^{-1} r_i' s_n$$

5. $n = n + 1$. Goto 2.

6. Repeat steps 1 to 5 $a_w$ times (obviously with a different choice(s) elements $r_i' 1 \leq i \leq a_w$ and maybe a different choice for the integer $a_w$).

7. Steps 1 to 6 are repeated again $a_v$ times but with $v_i'$ in place of $w_i'$ and $w_i'$ in place of $v_i'$ using a system of $n_v$ equations.
8. We now have stored two sets we refer to as \( BV \) and \( BW \).

\[
\{ z\overline{w}_1, z\overline{w}_2, ..., z\overline{w}_{a_w} \} = \{ \hat{w}_1, ..., \hat{w}_{a_w} \} = BW
\]

\[
\{ z\overline{v}_1, z\overline{v}_2, ..., z\overline{v}_{a_v} \} = \{ \hat{v}_1, ..., \hat{v}_{a_v} \} = BV
\]

9. If follows from the MSCSPv example that \( \overline{w}_I \overline{v}_J = \overline{v}_I \overline{w}_J \) for any \( I \) or \( J \). The attacker picks \( I \) and \( J \) and computes

\[
M_1 = \overline{v}_I^{-1} \overline{w}_J = \overline{v}_J^{-1} z^{-1} z \overline{w}_J \overline{v}_I^{-1} \quad \text{and} \quad Y = \hat{w}_J \hat{v}_I^{-1} = z \overline{w}_J \hat{v}_I^{-1} z^{-1}
\]

hence the attacker can solves the CSP \((M_1, Y)\) for \((z, z^{-1})\). Similarly

\[
M_2 = \overline{v}_I \overline{w}_J^{-1} = \overline{v}_J z^{-1} z \overline{w}_J^{-1} = \overline{w}_J^{-1} \overline{v}_I
\]

the CSP an solves the CSP \((M_2, Y^{-1})\) for \((z, z^{-1})\). Repeating the above for similar computations for different \( I, J \) builds up a system of conjugacy equations hence this reduces the MSCSPv to the MSCSP.

The algorithm is a modification of known length attack because we use the generators of the whole conjugated word \( zr_i z^{-1} \) and not just as usual the generators of \( z \), conjugated element with a partial factor of \( r_i \) is recovered and intermediate partial factors involving the secret are recovered and used not as usual the secret element.

A simple stopping criteria is for some \( C \)

\[
r(t_1, ..., t_{a_w}, e) < C < r(r'_1, ..., r'_{a_w}, e)
\]

stop when

\[
r(r'_i, s_n) < C
\]

and \( r(t_1, ..., t_{a_w}, e) \) is to be estimated by the attacker using the value of \( L \) given in [1].

At step 3 we could solve the equation \( z\tilde{r}_i \) for \( z \) which may be easier than solving the MSCSP instead and so we do not need to run all the steps, to be precise the algorithm is.

Algorithm 8 Length-MSCSPv.

Run step 1 and 2 of algorithm 6 for step 3 use the algorithm below instead of a version of algorithm 2.

Compute

\[
r'_i = zr_i z^{-1} t_i z^{-1} z r_i^{-1} z^{-1} = zr_i t_i r_i^{-1} z^{-1}
\]

where the words \( r_i \) and \( t_i \) are a word in the generators \( w_i \). Note an element of the form \( zr_i z^{-1} \) may be used instead for \( r'_i \).

1. Select a length function \( l \). Construct \( r'_i \) as a word in the generators \( w'_i \) for some \( 1 \leq i \leq n_w \). \( A \) is set to the identity element. Set the iteration \( n \) to zero. Computes a subset \( C_r \) of the generator set of the intersection of the centralisers generator sets, \( C_r \subseteq (C(v'_1) \cap ... \cap C(v'_{a_w})) \), \( a_c = 1 \) is sufficient, we may also try and compute the length of the generators of \( C_r \) with suitably long generators.
2. Select suitable elements $s_n \in C_r$. If

$$r(r'_1, ..., r'_{a_w}, s_n) \preceq r(r'_1, ..., r'_{a_w}, e)$$

where $\preceq$ is a linear ordering (or an objective function) on a vector of real numbers [11] and each element of the tuple $r(r'_1, ..., r'_{a_w}, s_n)$ except the last is given by the corresponding number $l(s_n^{-1}r'_i s_n)$.

3. Update the word $A$ as

$$A_{n+1} = A_n s_n.$$  
The algorithm stops at this part when depending on $r(r'_1, ..., r'_{a_w}, s_n)$ and the stopping criteria (there can be more than one stopping criteria) then goes to step 6. The algorithm stops with some probability $\rho_2$ with $A = \hat{z} r_i = \hat{r}_i$ and this braid is stored, where $\hat{r}_i = r_i C_i$ for some $C_i \in G$, in other words $A$ is the product of $z$ and a partial factor of $r_i$ we call $\hat{r}_i$.

4. Update the element $r'_i$ as

$$r'_i = s_n^{-1} r'_i s_n.$$

5. $n = n + 1$. Goto 2.

6. Repeat steps 1 to 5 $a_w$ times (obviously with a different choice(s) elements $r'_i$ $1 \leq i \leq a_w$ and maybe a different choice for the integer $a_w$).

7. Steps 1 to 6 may be repeated again $a_v$ times but with $v'_i$ in place of $w'_i$ and $w'_i$ in place of $v'_i$ using a system of $n_v$ equations.

8. We now have stored one set or two sets we refer to as $BV$ and $BW$.

$$\{ z w_1, z w_2, ..., z w_{a_w} \} = \{ \hat{w}_1, ..., \hat{w}_{a_w} \} = BW$$

$$\{ z \hat{w}_1, z \hat{w}_2, ..., z \hat{w}_{a_w} \} = \{ \hat{v}_1, ..., \hat{v}_{a_w} \} = BV$$

Using another algorithm we use the elements in $BW$ or $BV$ and solve for $z$ one of the simplest choices at this step is given an element of $BW$ or $BV$ find $\hat{w}_i$ or $\hat{v}_i$ by brute force and hence compute $z$ by using a right multiplication.

5. Attack Using Conjugacy Extractor Functions

5.1 First Attack using CE Functions

In the TTP algorithm above given in [1] step 2 is “chooses a secret element $z \in B_n$” a user could implement this step 2 as $z$ is chosen from a publicly known subgroup of $B_n$ we show that this implementation means a CE (conjugacy extraction) function [13] can be given. It is not given in [1] to not pick $z$ from a publicly known subgroup. The attack is as follows.

1. Let $z \in R$ where $R = \{ \alpha_1, ..., \alpha_k \}$ is a publicly known subgroup of $B_n$. In this step it is required the attacker just needs to find one element that commutes with $z$ and not with all possible choices of $u_i$ (using a chosen algorithm by the attacker) to show the AAGL protocol is based on the MSCSP one way to find such elements is as follows. The attacker picks a subgroup of $R$ given by the generators $g_1, ..., g_k$. Then the attacker computes all of or a large part of

$$S = C(\alpha_1, ..., \alpha_k) = C(\alpha_1) \cap ... C(\alpha_k).$$
2. Then

\[ CE(S_I, u'_I) = u'_I S_I u'^{-1}_I = zu_I z^{-1} S_I zu_I^{-1} z^{-1} = zu_I S_I u_I^{-1} z^{-1}, \]

will be true if \( S_I \) does not commute with \( u_I \). \( S_I \in S \), \( 1 \leq I \leq M \). The the protocol can be based on the MSCSP with

\[ ((S_1, S_2, ..., S_M), (CE(S_1, u'_1), CE(S_2, u'_2), ..., CE(S_M, u'_M))) \]

with solution \((o, o^{-1}), o = zu_i\).

and \( z \) can be found by computing \((o^{-1} u'_I)^{-1} = z\).

As a variant of the above algorithm an attacker may try to compute an element \( S'_I \in C(u'_I) \) then it may be possible to use \( S'_I \) instead of \( S_I \) in the attack above where \( u'_I \neq u_I \), so in this variant knowledge of \( z \) being chosen from a subgroup is not required.

5.2 Second Attack using CE Functions

This attack reveals partial information about the secret \( z \).

1. The attacker picks elements \( V_I \) according to some criteria for example elements \( V_I \) may be picked randomly or \( V_I \) may be composed of a few Artin generators as these may commute to some degree with \( z \).

2. Then for \( 1 \leq I \leq M \) for a sequence of integers \( T_I \)

\[ CE_I(S_I, u'_I) = u'_I S_I u'^{-1}_I = zu_{T_I} z^{-1} S_I zu_{T_I}^{-1} z^{-1} = zu_{T_I} \bar{z}^{-1} S_I \bar{zu}_{T_I}^{-1} \bar{z}^{-1} \]

where \( \bar{z} \) is a partial factor of \( z \) with probability \( \rho_3 \) for some \( I \) this means \( z = zu_{T_I} \bar{z}^{-1} \).

3. We solve for each \( I \) the CSP \((S_I, zu_{T_I} \bar{z}^{-1})\) and hence compute \( z_{T_I} = ((zu_{T_I} \bar{z}^{-1})^{-1} z_{u_{T_I}} z^{-1})^{-1} \)

4. We now and find \( z \) using the information \((S_I, z_{u_{T_I}} \bar{z}^{-1}, z_{T_I})\) and the other information used in the protocol. One of the simplest choices to implement this step is trying to find \( \bar{z}_{T_I} \) for each \( I \) by brute force.

A variant of the above attack is after \( z_{T_I} \) is recovered is to repeat at the attack (at least once) by iterating with \( z_{T_I}^{-1} u'_I, z_{T_I} \) instead of \( u'_I, \) (and obviously all other values may be different) so in this way we may be able to find a bigger factor of \( z \). It may be true some probability \( \rho_4 \) that \( \bar{z} \) contains a partial factor of \( u_{T_I} \) which means the CSP is solved to give \( \bar{z}_{T_I}, \bar{z}_{T_I}, \bar{z}_{T_I} \) where \( \bar{z}_{T_I} \) is some partial factor of \( u_{T_I} \). Then the simplest choice at this step to recover \( z \) is to find \( \bar{z}_{T_I} \) by brute force and use \( \bar{z}_{T_I} \) to recover \( z \). Note this attack is easily modified to solve the decomposition problem which means using a product of three elements instead of \( u_{T_I} \).

Another conjugacy extractor (see [13]) (i.e. this will show the AAGL protocol is based on the MSCSP again). The user may try computations of the form (following the notation of [1]) \( A_{public} o A_{public}^{-1} A_{public}^{-1} \alpha B_{public} \beta B_{public}^{-1} \delta B_{public}^{-1} \)

where \( A, B, C, D \) are chosen from \( \alpha \in B, \beta \in N_B, \gamma \in A, \delta \in N_A \). The information recovered from the MSCSP above may be used in attack such as for example the following attack. Then once an element of the form may be found
z \ast (x_{a_{1}}(t), s_{a_{1}}) \ast \ldots \ast (x_{a_{i}}(t), s_{a_{i}}) \ast z^{-1} \text{ then an element such as } (n_a, id) \text{ may be found and so the shared secret can be computed. We will give further details of this attack. To resist the above attack the public elements should be chosen so that they do not have an inverse.}

6. An algorithm for the MSCSP

Consider the MSCSP \(((x_1, x_2, \ldots, x_u), (y_1, y_2, \ldots, y_u))\) with solution \((g, g^{-1})\). Suppose \(x_i \in A, g \in B, \text{ with } A = \langle a_1, a_2, \ldots, a_M \rangle, B = \langle b_1, b_2, \ldots, b_N \rangle\) Compute a large part of all of the centraliser

\[ D = C(b_1, \ldots, b_N) = C(b_1) \cap \ldots C(b_N). \]

The we can compute the \(CE\) functions

\[ CE_k(d_k, y_i) = y_i dy_i^{-1} = gx_i d_k x_i^{-1} g^{-1}. \]

This means we have transformed the MSCSP into another MSCSP. We can use this transformed MSCSP to attack the protocol in [2], e.g. we may use the transformed MSCSP as part of another algorithm that solves the MSCSP such as a length attack, such a length attack is as follows.

1. Select a length function \(l\). \(A\) is set to the identity element. Set the iteration \(n\) to zero. Computes a large part or all of the centraliser \(D\).

2. While (Criteria=True)
   
   \{ Select elements \(s_n \in B\).
   
   Compute
   
   \[ CE_k = CE_k(d_k, y_i) = y_i dy_i^{-1} = gx_i d_k x_i^{-1} g^{-1}. \]

   for some \(1 \leq k \leq I\) for some \(I, d_k \in D\).

   \[ r(CE_1, \ldots, CE_I, s_n) \preceq r(CE_1, \ldots, CE_I, e) \]

   where \(\preceq\) is a linear ordering (or an objective function) on a vector of real numbers [11] and each element of the tuple \((CE_1, \ldots, CE_I, s_n)\) except the last is given by the corresponding number \(l(s_n^{-1}CE_k s_n)\).

   \} End While.

3. Update the word \(A\) as

   \[ A_{n+1} = A_n s_n. \]

   The algorithm stops at this part when depending on \(r(r_1', \ldots, r_{n_w}', s_n)\) and the stopping criteria (there can be more than one stopping criteria) then goes to step 6. The algorithm stops with some probability \(\rho_3\).

4. Update the element \(CE_k\) using

   \[ s_n^{-1}CE_k s_n \]

5. \(n = n + 1\). Goto 2.
6. Output $A$.

One choice at step 2 is to select $s_n$ as choices from all the generators of $B$, then choice for the Criteria at step 2 is to increase $I$ using a chosen algorithm until it is decided peeling occurs for one of the $N$ choices (we try all $N$ choices) from $B$ for $s_n$: if peeling is still undecided then the algorithm can pick a generator randomly or stops. We may include in step 2 the equations $y_i$ to peel from in the above.

7. Conclusion

The above attacks needs to be investigated further, because large parts of the centraliser for an element can be computed (but in general it is difficult to compute all elements in the centraliser) and we think the attacks can be improved. Not considering a brute force algorithm (which is shown in [1] that the AAGL protocol is secure from a brute force algorithm) we have given the only deterministic algorithm to break the AAGL protocol. We have given an algorithms for the MSCSPv is and shown can be reduced to solving the MSCSP using an algorithm of exponential complexity. Further work is

• To implement our deterministic attack or a variant of it for example, try randomised and/or genetic algorithms (for example these can be used to increase the probabilities $\rho, \rho_1$), evolutionary algorithms (e.g. differential evolution) which lead to more probabilistic solutions (an attacker can try our attack even if it is in worst case of exponential complexity).

• To minimize $O_1$ possibly with additional heuristics in algorithm 4.

• Try different length attacks apart from the basic length algorithm (which we have used) in the length-MSCSPv algorithms and to try different refinements for the above length algorithm these include randomised and/or genetic algorithms which lead to more probabilistic solutions. To test/implement the length-MSCSPv algorithms to give experimental results for its success for different parameters. The length-MSCSPv algorithms we have given can be used as the basis of other length-MSCSPv algorithms.

• As described in the attack given in section 5 it is sufficient to find one element that commutes with $z$ to show the protocol is based on the MSCSP (and so the AAGL protocol would be no more secure than using another MSCSP based protocol such as the AAG protocol given in [2]) a natural question now arises which is.

Given the example of the MSCSPv used in the AAGL protocol how easy or how hard is it to find an element $s$ that commutes with $z$ but $s$ does not commute with all choices of $u_i$?

this question needs further investigation.

References

[1] I. Anshel, M. Anshel, D. Goldfeld, S. Lemieux, Key Agreement. The Algebraic ErasorTM and Light Weight Cryptography in Algebraic Methods in Cryptography, 418, 2006, AMS
[2] I. Anshel, M. Anshel, D. Goldfeld. An Algebraic Method for Public-key Cryptography, Mathematical Research Letters, 6, 1999, pp. 287-292.

[3] 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, 1880, 2000, pp. 166-183.

[4] K. H. Ko, Tutorial on Braid Cryptosystems 3, PKC 2001, Korea, February 13-15, 2001. Available at www.ipk.org/pre-conf/pkc2001/PKC@ko.ps

[5] V. Gebhardt. A New approach to the conjugacy problem in Garside groups. J Algebra 292 (1), 2005, pp. 282-302.

[6] K. Ko, D Choi, M. Cho, J. Lee. New Signature Scheme Using Conjugacy Problem, available at http://eprint.iacr.org

[7] E. A. Elrifai, H. R. Morton. Algorithms for positive braids, Quart. J. Math. Oxford., 45, 1994, pp. 479-497.

[8] F. A. Garside. The braid group and other groups, Quart. J. Math. Oxford 78, 1969, pp. 235-254.

[9] N. Franco and J. Gonzalez-Meneses. Computation of centralizers in braid groups and Garside groups, Rev Mat Iberoamericana 19 (2), 2003, pp. 367-384.

[10] V. Shpilrain and A. Ushakov, A new key exchange protocol based on the decomposition problem, available at http://eprint.iacr.org/2005/447.pdf (Accessed 20/05/2006)

[11] D. Garber, S. Kaplan, M. Teicher, B. Tsaban, U. Vishne. Length-Based Conjugacy Search in the Braid Group, available at http://arxiv.org/abs/math.GR/0209267

[12] R. Storn, K. Price. Differential Evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization, 11, 1997, pp. 341 - 359.

[13] M. M. Chowdhury, On the Security of the Cha-Ko-Lee-Han-Cheon Braid Group Public-key Cryptosystem, ArXiv preprint, August 2007

[14] P. Dehornoy. Using Shifted Conjugacy in braid based Cryptography, Contemporary Mathematics.

[15] K. Ko, D Choi, M. Cho, J. Lee. New Signature Scheme Using Conjugacy Problem, available at http://eprint.iacr.org

[16] J. Lomgrigg and A. Ushakov, Cryptanalysis of Shifted Conjugacy Authentication Protocol, ArXiv preprint, August 2006

Appendix

In this appendix we another version algorithm 4 which is presented in the style of the paper [16]. Then we give an attack on the DSC (Dehornoy Shifted Conjugacy) protocol in [14]. This appendix follows the style of the LU paper [16].

**A Probabilistic Algorithm for the Multiple Simultaneous Conjugacy Search Problem Variant**

We can define the length of the element $x \in B_n$ to be the length of its Garside normal form, and we denote an arbitrary length function by $l(x)$ or $l_i(x)$.
Recall, we refer the CSP as the MSCSP with \( u = 1 \) so an example of the CSP is denoted as \((x_1, y_1)\). Informally we refer to \( x_1 \) as the “middle element” of \( y_1 \).

Recall, to break the scheme it is sufficient to find \( z \) or a solution that can be used in place of \( z \), then once the common secret conjugate \( z \) is recovered with our attack, the shared secret key can be computed with the linear algebraic attack given in [1]. Recall the security of the protocol is based on the MSCSPv. The security of the TTP algorithm is based on the MSCSPv with the elements \((x_1, x_2, \ldots, x_u) = (w_1, \ldots, w_\gamma, v_1, \ldots, v_\gamma)\), \((y_1, y_2, \ldots, y_u) = (w'_1, \ldots, w'_\gamma, v'_1, \ldots, v'_\gamma)\) and \( u = 2^\gamma \). We assume the attacker knows this instance of the MSCSPv in the Artin representation.

In this appendix we give algorithms which are a probabilistic reduction from the MSCSPv to the MSCSP this includes another version of algorithm 4.

At the end of the appendix we make suggestions for secure protocols parameters.

To summarize our work.

A. We give an algorithm (another version of algorithm 4) to show the MSCSPv can be probabilistically reduced to the MSCSP.

B. We give a new algorithm to solve a hard problem (which is a generalisation of the SCSP) we refer to as the MSSDP.

C. We give an algorithm for the MSSDPv; the MSSDPv generalises the MSCSPv and the MSSDP simultaneously.

**Definition** - \( C \) is an algorithm that computes elements in the centralisers of given elements in factorial space and time complexity in a worst case.

**Proposition A**

For the MSCSPv with \( u = 1 \) if \( x_1 \) can be feasibly computed then \( z \) can be found by solving the CSP\((x_1, y_1)\).

**Proof**

Follows from the definitions of MSCSPv and CSP above.

**Proposition B**

Let \( u'_i \sim u_i \), for some \( 1 \leq i \leq 2^\gamma \) (i.e. the example of the MSCSPv used in the AAGL protocol) and \( z, k \in B_n \). Then for all \( i \) we have

\[ zk^{-1} \in C_{B_n}(u'_i) = C_{B_n}(zu_iz^{-1}) \]

where \( C_{B_n}(u'_i) \) denotes the centraliser of \( u'_i \) in \( B_n \).

**Proof**

Obvious. □

1 On \( n \)-Centraliser Attacks
Now it follows from the above propositions A, B that the MSCSP \( v((x_1, x_2, \ldots, x_u), (y_1, y_2, \ldots, y_u)) \) can be solved in two steps:

(S1) Find suitable element(s) \( c, c \in C_{B_u}(u'_i) \) for at least two values of \( u'_i \) such that \( u'_i = v_j \) and \( u'_i = w'_k \), and \( c \) are of the form \( z k z^{-1} \) using algorithm \( C' \). The computation of the centraliser may be based on the super summit technique in [7]. We refer to this step as: a \( n \)-centraliser attack or a centraliser(s) attack.

(S2) Find using some algorithm: values of \( k, k = x_i \) (for all \( i \)) in the MSCSP\( v \) then solve the corresponding MSCSP.

The description of super summit sets is described in [7] so we omit the description of that part of (S1) here. But (S1) still requires some elaboration as follows. To be able to work with elements of \( C_{B_u}(u'_i) \) efficiently we need to describe \( C_{B_u}(u'_i) \) in some convenient way, for instance by a set of generators. Hence (S1) itself consists of two smaller steps: capturing (i.e. computing suitable approximations of the centraliser(s)) the union of various centraliser(s) we refer to as \( C' \), and finding the required element \( c \in C' \) in the above union. We formalize the type of attack in (S1) as follows.

**Definition** An \( n \)-centraliser attack or a centraliser(s) attack is an attack where the computation of \( n \) centralisers is involved, then a set of elements from the above \( n \) centraliser(s) is found in connection to some conditions. The elements found above are used as part of the attack. E.g. we refer to step (S1) as a \((2 + d)\)-centralisers attack where elements are found to build an MSCSP, otherwise if \( n = 1 \) we refer to step (S1) as a centraliser attack or a 1-centraliser attack.

Our algorithm for the MSCSP\( v \) is a \((2 + d)\)-centraliser attack where \( 1 \leq d \leq \gamma - 1 \). A centraliser attack is given in [16] so our idea of centraliser(s) attack extends the idea of “a centraliser attack” in [16].

The only known algorithm [9] for computing a generating set for a centralizer reduces to the construction of super summit sets, the size of which is not known to be polynomially bounded, and which is usually hard in practice. Hence the approach of describing the whole generating set is not feasible but we will use a variation of this approach. Another approach to investigate is to find a feasibly computable subgroup as a generating set of when \( k \) is of polynomial length (and hence feasibly computable). By polynomial above we mean the degree of the polynomial is small enough for practical computations. We summarize the ideas of this appendix into a heuristic probabilistic algorithm 1.1 below.

Informally, the algorithm below works because of the following:

i) If \( g_a \) is small enough then the “middle elements, ‘the \( x_i \)’s’” in the MSCSP\( v \) can be found by guessing; the “middle element” is known in the MSCSP; hence we can reduce MSCSP\( v \) to the MSCSP if the above guess is correct.

ii) \( g_a \) is suggested to be small in the AAGL protocol because \( a \) is unknown and AAGL is for a lightweight platform.

iii) The structure of the TTP algorithm in the AAGL protocol implies that we can get \( g_a = 1 \) for a suitable choice of the algorithm \( C \), and if we select
two values of $u_i'$ such that $u_i' = v_j$ and $w_k'$ this implies we can inefficiently
deterministically reduce the MSCSPv to the MSCSP. This is because $BL, BR$
do not have any generators in common and so the middle element must be
correct for one of the two above choices of $u_i'$.

iv) We can use some type of search, such as a heuristic search, to find the
"middle element" more efficiently.

v) We can test that we have found the correct "middle element" by using the
property of the efficiently computable braid invariant $\lambda$, which is an invariant
in conjugation, i.e. $\lambda(x_1) = \lambda(y_1)$ in the CSP, we can get enough information
of the "middle element" easily without solving the triple decomposition problem
to get the "middle element".

The text in italics at each step in the probabilistic algorithm is a suggestion
for an example of that step. The algorithm is for a general example of the
MSCSPv but suggestion are made specifically for the AAGL protocol in our
algorithm. Step A implements (S1). Step B implements (S2).

Algorithm 1.1 - Probabilistic algorithm for MSCSPv

INPUT: An example of the MSCSPv $(y_1, y_2, ..., y_u)$ in $((x_1, x_2, ..., x_u), (y_1, y_2, ..., y_u))$
and the value of $u$ called $t$.

OUTPUT: A solution $g'$ for the MSCSPv.

COMPUTATION:
A. Set $S = M = C = A = \emptyset$. Using an chosen algorithm $C$ compute
$C' \subset C_{B_n}(u_i')$ for some values of $i$, i.e. $C' = \cup C_{B_n}(u_i')$. It follows the $C'$ may
contain elements of the form $F = k z^{-1}$. Hence for choices of $k$ includes all
elements in $BR$ or $BL$ depending if $u_i' = v_j'$ or $u_i' = w_k'$.

There are two choices we suggest for this step:

First choice: compute the centraliser as a generating set using the algorithm
in [9]. Then select random products of the generators to give a word $r$. An
option for this choice is using a suitable length function $l_1$ we would expect if $r$
is conjugated by $z$

$$l_1(r u_i') < l_1(r) + l_1(u_i')$$
$$l_1(u_i' r) < l_1(r) + l_1(u_i')$$

The above idea is based on the Hamming distance between words: i.e. if $r$ and
$u_i$ are both conjugated by the same element then we would expect both the
above inequalities to be true.

Second choice: Use the subalgorithm for this step described below.

B. Repeat steps Bi, Bii, Biii until a solution is found.

We construct the pair $(a, z a z^{-1}) = (a, b)$, find $a$ as follows, and $z a z^{-1} \in C'$
as follows.

Bi. Select $b \in C' \setminus M$ and add $b$ to $M$ if $b$ is selected. If all possible values
for $b$ have been used (i.e. $M = C'$) goto step A. Using a chosen algorithm we
find $b$ in the pair $(a, b)$ as follows. A choice at this step is that because anyone
knows the length $g_z$ in Artin generators of $z$ [1] and we assume the length in
Artin generators of $a$ is less a feasible bound $g_a$, hence we do a type of search
(e.g. linear search or random search) using an algorithm that uses \( l_2(b), g_z, g_a \)
e.g. an algorithm that uses the heuristic \( l_2(b) \leq 2g_z + g_a \).

Bii. Construct using a chosen algorithm, the set \( A \) which is the set of possible values for \( a \). Then using a chosen algorithm we find \( a \in A \). If the relation \( b \sim a \) is true then add the pair \((a, b)\) to \( S \), because this means \( a \) is conjugate to \( b \). \( M \) represents worked out elements for \( b \). \( S \) represents worked out elements that are used in an MSCSP. A choice at this step is that we can try out all possible values of \( a \) up to length \( g_a \) (using a chosen length function) this includes easy to guess choices which exist (follows from the TTP algorithm) for \( a \) which are single Artin generators. We then use the practical algorithm for the CDP in [6] to test the relation \( b \sim a \)
\[
\lambda(b) \sim \lambda(a)
\]
where \( \lambda \) is a braid invariant.

iii. Repeat steps i and ii until the desired value of \( u = t \) is reached in the MSCSP. (If the desired value of \( t \) not reached then at \( i \) goes step A). Otherwise goto step C.

C. Solve the MSCSP for all the pairs \((a, b)\) using an algorithm that works with high probability: if solution has been found terminate algorithm, if the solution of the MSCSP has not been found goto step A.

Algorithm 1.2-Second choice for Subalgorithm in Step A of algorithm 1.1

Here we can use a optimization method (e.g. simulated annealing) by considering the \((y_1, y_2, \ldots, y_u)\) as input to the optimization method and minimising the length (using a chosen length function) of \( a \) in the CSP pair \((a, b)\). For example the simplest choice here is the following simple optimization algorithm. The idea of this subalgorithm is words formed from words in the generators \((y_1, y_2, \ldots, y_u)\) are in \( C' \), and such word(s) may have a small length in \( a \). The subalgorithm can be used at least twice depending on if \( u_i' = v_i' \) or \( u_i' = w_i' \).

1. Initialisation step. Choose a suitable length function \( l \). Set \( \text{fitness} = \min_{1 \leq k \leq u} l(y_k) \). Set \( \text{Solution} = l(y_k) \) with \( y_k \) such that it has minimal fitness (i.e. it is a solution).

2. Select random subsets of the MSCSPv \((y_1, y_2, \ldots, y_u)\) tuple, the simplest choice at this step is to select random \( y_i, y_j \) for two random \( i \) and \( j \).

3. The objective function should grow smaller as \( a \) is smaller. Then the simplest choice is using

\[
\text{Objective function} = l(y_i^{\pm 1} y_j^{\pm 1}).
\]

Then if \( l(y_i^{\pm 1} y_j^{\pm 1}) < \text{fitness} \) we add the element \( y_i^{\pm 1} y_j^{\pm 1} \) to \( C' \) if \( y_i^{\pm 1} y_j^{\pm 1} \notin C' \). Where \( l \) is a length function.

4. Repeat steps 2 and 3 until the desired until the desired size of \(|C'|\) is reached. If the desired size is not reached then proceed to the next step in the main algorithm.

25
Proposition 1.1

Let $\lambda_1, \lambda_2, \omega, c_1, c_2 \in \mathbb{R}$. Solving the MSCSPv can be done with the probabilistic algorithm 1.1, in approximately, time complexity $O(|XSS(u'_i)|^{\lambda_2} \omega n)$ and space complexity $O(c_1 \omega^n + c_2 |XSS(u'_i)|^{\lambda_1})$.

Proposition 1.2

Let $\lambda_1, \lambda_2, \omega, c_1, c_2 \in \mathbb{R}$. This proposition improves the complexity bound of proposition 1.1: with additional reasonable assumptions it can be improved to time complexity $O(|XSS(u'_i)|^{\lambda_1} n^{1+\epsilon})$ and space complexity $O(|XSS(u'_i)|^{\lambda_2})$ using algorithms to solve the MSCSP, CDP efficiently, where the element $u'_i$ is part of the TTP’s public key, $XSS$ is a summit type set and the constant $\omega$ depends on $n$.

Proof for Proposition 1.1

We assume that algorithm 1.1 is successful this means it gives the correct output and in particular at step Biii the value of $t = u$ is reached and that a linear search is used at Bi. We do the complexity analyses by following the algorithm 1 through a successful execution.

**Step A.** The complexity at step A is determined by the algorithm $C'$ which computes elements in the centraliser. It is assumed that $C'$ has factorial space and time complexity: this assumption is based on the fact that such algorithms exists e.g. when $XSS = SSS$, or $XSS = SS$ see [9], [8]; precisely we mean $C'$ has space and time complexity proportional to $|XSS(u'_i)|$. Recall, generically an element of $C'$ contains elements of the form $F = za z^{-1}$; this is an important thing to observe.

**Step Bi.** Since we are trying to construct an MSCSP in $t$ equations it follows we have to do step Bi at least $t$ times (i.e. at step Biii we repeat $t$ times). In the worst case we would have to try out all (i.e. using the linear search) of the elements of $C'$ (stored in the above step) which is of size $O(|XSS(u'_i)|^{\lambda_1})$.

**Step Bii.** Recall we are trying to find $a$ in $F$. We analyse the simplest method which is simply to randomly guess $a$ in its Artin generators. This gives the following straightforward computations.

- From the TTP algorithm it follows a randomly chosen generator has probability $p_\alpha = \frac{l_\alpha}{n^2}$ and $p_\beta = \frac{r_\beta}{n^2}$ of being in $BL$ and $BR$ respectively. Hence an attacker can selects a random word $a$ from $A$ using in length of $g_a$ Artin generators then it has $p_{\alpha,\beta,g_a} = \frac{1}{p_\alpha} + \frac{1}{p_\beta}$ probability of being in $BL$ or $BR$; the above is true because we selected above values of $u'_i$ such that $u'_i = v_j$ and $w'_k$. From step Bi we compute a subset of $C'$ (with words less than a certain bound) we do not have to pick $g_a$ too large because as noted above there exist some $a$ of short length in Artin generators.

- Observe the attacker must on average compute

$$W p_{\alpha,\beta,g_a}^{-1} < \frac{(n - 2)g_a}{\min(l_\alpha, r_\beta)^g_\alpha}$$

before expecting $W$ words to be found in for $a$ in $(a, za z^{-1}) \in (A, C_{B_a}(u'_i))$. 

26
Choosing \( g_a \leq n \) keeps the algorithm in factorial complexity but this is not a good choice, from the above discussion the attacker can take \( g_a = O(\frac{n}{\log(n)}) \); in particular from \( \left\lceil W_{P_{\alpha,\beta},O(\frac{1}{\log(n)})} \right\rceil < O(n^{O(\frac{1}{\log(n)})}) = O(e^n) = \omega^n \) (for some \( \omega \in \mathbb{R} \)), means this part is exponential.

- It follows the attacker needs to estimate \( p_{\alpha}, p_{\beta} \). The attacker may estimate \( p_{\alpha}, p_{\beta} \) if the attacker assumes \( l_{\alpha}, r_{\beta}, n \) are not independent of each other, for example \( l_{\alpha} \approx r_{\beta} \) and assumes for example, all nearly all possible Artin generators are used in \( BL, BR \) : so \( p_{\alpha} \approx p_{\beta} \). Note if all possible Artin generators are used then \( p_{\alpha,\beta,1} = 1 \), for the selection of \( a \) (the single Artin generator). Hence (independent of large enough \( n \) and when \( l_{\alpha} \approx r_{\beta} \)), the attacker needs to compute approximately as few as \( 2^{g_a} \) distinct words for the parameters suggested in [1] to ensure on average a reduction to the MSCSP with at least 2 equations. So we would need to select only approximately 4 distinct random words of length 3 from the set of possibilities of \( k \), before the attacker expects to get one conjugacy equation or the CSP, the example above use little memory and potentially little computing power.

At the next part in step Bii we use an efficient algorithm for the CDP such as the one given in [6], use a linear search, and use an algorithm for the MSCSP that works with high probability. It is expected from Bi the length of \( b \) is less than \( u_i \), this means in general \( O(|SSS(u_i)|) \) is greater than \( O(|SSS(b)|) \).

**Step Biii.** Steps Bi to Bii are repeated \( t \) times hence the complexity in the steps is has a factor of \( t \).

We can now evaluate the total complexity of the algorithm.

**Notation**—The notation \( A_s \) and \( A_t \) means the space complexity and time complexity respectively for an arbitrary algorithm labelled \( A \).

The time complexity is in the worst case is

\[
O(C_{B_n}(u'_i)_t + CDP_t \cdot |C_{B_n}(u'_i)| \cdot \omega^n + MSCSP_t)
\]

hence an upper bound is \( O(|XSS(u'_i)|^{\lambda_1} \omega^n). \)

The explanation is as follows. The term \( C_{B_n}(u'_i)_t \) used in step A is between 2 and \( 2 + d \) times or a constant number of times, and this implies the corresponding constant in the complexity term can be ignored. The term \( MSCSP_t \) is the complexity at the last step and has order less than \( |XSS(u'_i)|^{\lambda_1} \). The term \( CDP_t \cdot |C_{B_n}(u'_i)| \cdot \omega^n \) is the complexity at step Bii: at step Bii means step Bii is repeated \( \omega^n \) times for each possible value of \( a \) in conjunction with each possible value for \( b \); there are \( C_{B_n}(u'_i)_s \) values for \( b \). Here the constant \( CDP_t \) is the average time taken to solve the CDP over all pairs \( (a,b) \). In the worst case space complexity at this sub step is \( O(t|XSS(u'_i)|^{\lambda_1}) \). Clearly the term \( CDP_t \cdot |C_{B_n}(u'_i)| \cdot \omega^n \) dominates the complexity. Because the CDP is done using an efficient probabilistic algorithm such as [6] and \( t \leq \omega^n \) the time complexity follows.
The worst space complexity is

\[ O(|C_{B_n}(u'_i)| + |A| + \max_{a \in A} \max_{b \in C'} CDP_a(a, b) + MSCSP_s) \]

hence an upper bound is \( O(c_1 |XSS(u'_k)|^{\lambda_2} + c_2 \omega^n) \).

The explanation is as follows. The terms \(|C_{B_n}(u'_i)|, MSCSP_s\) both have complexity equal or less than to \( O(c_1 |XSS(u'_k)|^{\lambda_2}) \) (so we have combined both complexities into one term). \(|A|\) is of size \( c_2 \omega^n \). max_{a \in A} max_{b \in C'} CDP_s(a, b) is negligible as the practical algorithm [6] stores only two elements. □

Now we are in a position to prove proposition 1.2.

**Proof for Proposition 1.2**

The above analysis may not be optimal, for example if we make some reasonable assumptions then we get a better bound on the complexities as follows. For this case the average complexities (instead of worst case) for algorithm for the CDP, CSP and MSCSP are considered.

The reasonable assumptions we are as follows.

- From [1] we assume \( l_\alpha, r_\beta \) are linear in \( n \) and the attacker selects \( g_a = O(\log(n)) \).
- We assume we have an efficient algorithm for the CDP which has average linear complexity possibly the one given in [5], this assumption is based on the result that empirically for randomly chosen long random braids which have simple elements randomly chosen, the \(|USS|\) is on average likely to be linear in the supremum and independent of the braid index \( n \) e.g. see [5]. Hence the CDP/CSP in this average case can be solved in linear space and time complexity;
- use an algorithm for the MSCSP that works with high probability such as the one in [5].

It follows (using the assumptions) the dominant term in the time complexity for this term is the dominant term for the time complexity in the related proposition 1.1 with the term \( B^{O(\log n)} \) replacing \( \omega^n \). The time complexity in this better case can be with high probability be (recall computing \( C' \) that in proportional in space an time complexity to \(|XSS|\))

\[ O(c_1 |XSS(u'_i)|^{\lambda_2} \cdot B^{O(\log n)} O(n)) = O(|XSS(u'_i)|^{\lambda_1} n^{1+\epsilon}) \]

for some \( b \in \mathbb{R} \) which depends on equation a. The factor \( O(n) \) is for the complexity for the CDP algorithm.

It follows (using the assumptions) the dominant term in the space complexity for this term is related to the space complexity in the related case above proposition 1.2. The space complexity is

\[ O(c_1 |XSS(u'_i)|^{\lambda_2} + B^{O(\log n)} + |C_{B_n}(u'_i)|) \]

\[ \approx O(c_1 |XSS(u'_i)|^{\lambda_2} + c_2 \omega^n) \approx O(|XSS(u'_i)|^{\lambda_2}). \]

Using straightforward algebra it can be shown \( \epsilon \) can be close to a constant as \( n \) becomes larger and depends on \( b \) and the constants in \( O(\log n) \), if \( b \) is bounded then \( \epsilon \) is bounded.
Note the space can be up to exponential size (so giving a better space bound here), the only requirement for the algorithm to successfully terminate, is the set $C'$ must be non-empty than one as it must contain at least one element with some feasible computable $k$.

The above shows using the AAGL protocol can potentially be as secure as using CSP based protocols such as the AAG protocol [2] as both can be broken with attacks of the same or similar complexity depending on the values $\lambda_1, \lambda_2$ and $\epsilon$, by similar we mean our instantiations of our attack can differ by a factor that is polynomial of the bitlength attack input (i.e. similar) from attacks such as on the AAG protocol, for example the time complexities of attack differ by a factor of $n^{1+\epsilon}$ compared to the $SSS$ attack.

We can try a variant of the above algorithm, which is not use an algorithm for the CDP in step Bii but instead solves the CSP with the guess for $b$ with every possible element in $C'$ and recovers $z$, and hence the shared secret using the linear algebraic attack given in [1], the attacker may test if $z$ is the correct solution: for example by computing if $z^{-1} u'_i z \sim u'_i$.

Another variant we can try is: because it may be that $b \in C'$ (which may be verified using a polynomial time word algorithm in $B_n$), in this case $z$ must be in the centraliser of $b$, call the set of all such stored $b$ as $b_z$, and so $z$ can be found by testing every element of the centraliser of a subset of $b_z$ for the correct element. 

Observe if the attacker assumes his guess of the generators of $BL, BR$ are correct (or manages know these subgroups in a different way) the attacker can compute randomly chosen words computable in polynomial time in $BL, BR$, and in up to factorial time (in approximately the time taken to solve the CDP) find a system of conjugacy equation / reduce the security of the AAGL protocol to the MSCSP, so this is another reason why the users should keep the subgroups $BL, BR$ secret. For the modification to the AAGL which is using general $BR$ and $BL$ the algorithm, then our attack has to be modified to use the publicly known information about the structures of $BL$ and $BR$.

We see to increase the probability of for our attack on the AAGL protocol above to succeed we have to compute a “something like a geodesic of $u'_i$”. A geodesic of a braid is a braid word of minimum length in the Artin generators representing a given braid. It is known that computing the geodesic of a braid is an NP-complete problem. However the version of the geodesic problem the AAGL protocol is based upon is to find a word equivalent to $zaz^{-1}$ such that it is short enough for $a$ to be feasible computed. WLOG assume $za \in B_n^+$, the above problem is (easily) equivalent to replacing $zaz^{-1}$ by $zaz^*$ where $z^* = z^{-1} \Delta_{\sup(z)}$, so it is sufficient to find a geodesic like element of $zaz^*$ in $B_n^+$. Even though computing a geodesic is NP-complete there are two reasons why the above problem may still be easy: the first is the problem is a version of the geodesic problem and not the exact geodesic problem; the second is there are many NP-complete problems have polynomial time average case solutions, e.g. observe it is easy compute a geodesic of a permutations braid hints that there is such a solution of the above problem.
Recall the shift operator in \( B_\infty \) for the word \( w = \sigma_{i_1}^{\epsilon_1} \ldots \sigma_{i_k}^{\epsilon_k} \) as the word
\[
d(w) = \sigma_{i_1+1}^{\epsilon_1} \ldots \sigma_{i_k+1}^{\epsilon_k}
\]
This operator induces a monomorphism on the infinite braid group. Recall the braid \( a \ast b \) is
\[
a \ast b = a \cdot d(b) \cdot \sigma_1 \cdot d(a^{-1}),
\]
and the operator \( \ast \) is the shifted conjugacy operator. Recall the SCSP (shifted conjugacy search problem) is defined as the following hard problem. For braids \( x, y, c \in B_\infty \) find a braid \( x \in B_\infty \) such that \( y = x \ast c \) : where \( c, y \) are publicly known and \( x \) is secret.

We now generalise the SCSP in a straightforward way to a hard decomposition type problem called the SDP.

**Definition.** The SDP (shifted decomposition problem), for braids \( w, x, y, c \in B_\infty \) find braids \( w, x \in B_\infty \) such that \( y = wd(c) \sigma_1 d(x) \) where \( c, y \) are publicly known and \( w, x \) are secret. The SDP is a generalisation because with \( x = w^{-1} \) we recover the SCSP.

**Notation.** We use the notation \( w \ast c \ast x = y \) for the SDP.

**Definition.** The MSSDP (multiple simultaneous SDP), is a set of SDP equations, as follows. Let \( n \geq 1 \) be a fixed integer. For braids \( w, x, y_i, c_i \in B_\infty \), \( 1 \leq i \leq n \) find braids \( w, x \in B_\infty \) such that \( y_i = wd(c_i) \sigma_1 d(x) \) where \( c_i, y \) are publicly known and \( w, x \) are secret.

There are no efficient solutions for solving the MSSDP, one reason for is this would mean the SCSP would be easy. We propose a solution for MSSDP.

Consider \( n = 4 \) in the MSSDP and \( c_1 \neq c_2, c_3 \neq c_4 \), this is the system of equations
\[
\begin{align*}
y_1 &= wd(c_1) \sigma_1 d(x), \quad y_2 = wd(c_2) \sigma_1 d(x) \\
y_3 &= wd(c_3) \sigma_1 d(x) \quad \text{and} \quad y_4 = wd(c_4) \sigma_1 d(x).
\end{align*}
\]
We now use the idea CE (conjugacy extractor) [13] used to transform the MSSDP into a shifted MSCSP type problem very efficiently: the transformation is achieved using CE functions, the concept of CE functions were first introduced in our paper [13]. A CE function is defined as follows, **definition** A CE function uses input from public information in a hard problem and transforms the hard problem into a an example of the CSP. First we give the mathematical background then we give our centralisers attack.

**On an algorithm For the MSSDP**

Define the CE function for one SDP in b as
\[
CE(y_1, y_2) = y_1^{-1} y_2 = d^{-1}(x) \sigma_1^{-1} d^{-1}(c_1) d(c_2) \sigma_1 d(x)
\]
which is equivalent to solving the CSP with \((\sigma_1^{-1} d^{-1}(c_1) w^{-1} wd(c_2) \sigma_1, CE(y_1, y_2))\) with solution \( d(x) \).
For $n \in N$ define the braids

$$\delta_n = \sigma_{n-1} \ldots \sigma_1.$$ 

Then for $i = 1, \ldots, n - 1$

$$\delta_{n+1}^{-1} \sigma_i \delta_{n+1} = b_{n+1} \delta_i = d(\sigma_i). \quad (d)$$

**Proposition 2.1**

Let $x, CE(y_1, y_2), \sigma_1^{-1}d^{-1}(c_1)d(c_2)\sigma_1 \in B_n$. Then $d(x)$ satisfies equation c for the CSP $(\sigma_1^{-1}d^{-1}(c_1)d(c_2)\sigma_1, CE(y_1, y_2))$ if and only if it satisfies the CSP $(\delta_{n+1}^{-1}d^{-1}(c_1)d(c_2)\sigma_1 \delta_{n+1}^{-1}, \delta_{n+1}CE(y_1, y_2)\delta_{n+1}^{-1})$ i.e.

$$\delta_{n+1}CE(y_1, y_2)\delta_{n+1}^{-1} = x\delta_{n+1}^{-1}d^{-1}(c_1)d(c_2)\sigma_1 \delta_{n+1}^{-1}x^{-1}. \quad (e)$$

**Proof**

Follows from d.

**Proposition 2.2**

Let $x, CE(y_1, y_2), \sigma_1^{-1}d^{-1}(c_1)d(c_2)\sigma_1 \in B_n$ be braids satisfying equation c and let $x'_1 \in B_{n+1}$. Then

$$\delta_{n+1}CE(y_1, y_2)\delta_{n+1}^{-1} = x'_1\delta_{n+1}^{-1}d^{-1}(c_1)d(c_2)\sigma_1 \delta_{n+1}^{-1}x'_1 \Leftrightarrow x'_1^{-1}x' \in C_{B_{n+1}}(\delta_{n+1}^{-1}d^{-1}(c_1)d(c_2)\sigma_1 \delta_{n+1}^{-1})$$

where $C_{B_{n+1}}$ is a centraliser in $B_{n+1}$.

**Proof**

The proof for the similar proposition in [16] is “obvious” and so is the proof for this proposition.

Now consider the equation

$$CE(y_3, y_4) = y_3^{-1}y_4 = d^{-1}(x)\sigma_1^{-1}d^{-1}(c_3)d(c_4)\sigma_1 d(x) \quad (f)$$

We can now easily derive two very similar propositions to 2.1, 2.2 where we use $y_3, y_4$ in place of $y_1, y_2$ respectively. To be precise and to be complete the propositions are 2.3 and 2.4.

**Proposition 2.3**

Let $x, CE(y_3, y_4), \sigma_1^{-1}d^{-1}(c_3)d(c_4)\sigma_1 \in B_n$. Then $d(x)$ satisfies the equation f for the CSP $(\sigma_1^{-1}d^{-1}(c_3)d(c_4)\sigma_1, CE(y_3, y_4))$ if and only if it satisfies the CSP $(\delta_{n+1}^{-1}d^{-1}(c_3)d(c_4)\sigma_1 \delta_{n+1}^{-1}, \delta_{n+1}CE(y_3, y_4)\delta_{n+1}^{-1})$ i.e.

$$\delta_{n+1}CE(y_3, y_4)\delta_{n+1}^{-1} = x^{-1}\delta_{n+1}^{-1}d^{-1}(c_3)d(c_4)\sigma_1 \delta_{n+1}^{-1}x. \quad (g)$$
In the we are using the fact that we are solving an MSCSP \( c, f \) and attempting to recover the same value \( x \) of \( [5] \).

To obtain a solution \( s \in \mathcal{X}_s \), this can be done using and \( C \) the derivation for the \( 2 \)-centralisers attack.

We now describe a \( 2 \)-centralisers attack on the MSSDP that recovers \( x \). Once \( x \) is recovered we attempt to find \( w \) by computing \( y_w(d(c_1)d(x))^{-1} = ? \) \( w \).

Now it follows from the above four propositions the MSSDP can be solved using the following steps.

(S1). Find the solution \( x_1', x_2' \in B_{n+1} \)

\[
\delta_{n+1}CE(y_1, y_2)\delta_{n+1}^{-1} = x_1'^{-1}\delta_{n+1}\sigma_1^{-1}d^{-1}(c_3)d(c_4)\sigma_1\delta_{n+1}^{-1}x_1'^{-1} \quad (h)
\]

We now derive a feasibly computable subgroup of \( C_{B_{n+1}}(\delta_{n+1}\sigma_1^{-1}d^{-1}(c_3)d(c_4)\sigma_1\delta_{n+1}^{-1}) \)

to obtain a solution

\[
t = x_1'C_1 = x_2'C_2 \in B_n. \quad (j)
\]

In the we are using the fact that we are solving an MSCSP \( c, f \) and attempting to recover the same value \( x \) (i.e. \( x = t \) here) in that MSCSP. This step is a \( 2 \)-centralisers attack.

We now derive a feasibly computable subgroup of \( C_{B_{n+1}}(\delta_{n+1}\sigma_1^{-1}d^{-1}(c_3)d(c_4)\sigma_1\delta_{n+1}^{-1}) \)

the derivation for \( C_{B_{n+1}}(\delta_{n+1}\sigma_1^{-1}d^{-1}(c_3)d(c_4)\sigma_1\delta_{n+1}^{-1}) \) is similar.

For \( c_1, c_2 \in B_n \) define the braids

\[
d_1 = \Delta_{n+1}^2, d_2 = \sigma_1...\sigma_2d^{-1}(c_1)d(c_2)d_2^{-1}...\sigma_n^{-1}, d_3 = \sigma_1...\sigma_n^2\sigma_{n-1}...\sigma_1
\]

and

\[
d_4 = d_1^{-1}, d_5 = d_2^{-1}, d_6 = d_3^{-1}.
\]

**Proposition 2.5**

There is a similar proposition in [16]. Let \( c_1, c_2 \in B_n \) and \( C = C_{B_{n+1}}(\delta_{n+1}\sigma_1^{-1}d^{-1}(c_3)d(c_4)\sigma_1\delta_{n+1}^{-1}) \).

The following i) and ii) holds.

i) \( d_1, d_2, d_3 \in C \).

ii) \( C' = \langle d_1, d_2, d_3 \rangle \) is an abelian subgroup of \( B_{n+1} \) and hence of polynomial growth.

**Proof**
Observe in $B_{n+1}$

$$
s_n \cdots s_2 d^{-1}(c_1) d(c_2) s_2^{-1} \cdots s_n^{-1} = \delta_{n+1} s_1^{-1} d^{-1}(c_1) d(c_2) s_1 \delta_{n+1}^{-1}
$$

so $d_2 \in C$. We know from [16] for arbitrary $p_1 \in B_n$ that elements of the form $d(p_1) s_2^{-1} \cdots s_n^{-1}$ commute with $d_3$, hence $d_3$ commutes with $d_2$, as $d_2$ is of the form $d_2 = (d(p_1) s_2^{-1} \cdots s_n^{-1})^{-1} d(p_2) s_2^{-1} \cdots s_n^{-1}$. $d_1$ is in the center hence the subgroup $C'$ is abelian. □

Straightforward variations of our attack are possible. One is could be as follows. Let $x'_1 = x'_2$ i.e. solve an MSCSP at (S1) and the correct this solution using $C_1, C_2$ and we may recover the actual value of $r$ from one of the correct solutions.

Our ideas can be summarised into the following algorithm.

Algorithm 2.1 - Heuristic Algorithm for solving the MSSDP.

INPUT: The example of the MSSDP given by the equations $c_i \in C$, $s_j \in S$.

Computations:

A. Compute

$$
CE(y_1, y_2) = y_1^{-1} y_2 = d^{-1}(x) s_1^{-1} d^{-1}(c_1) d(c_2) s_1 d(x)
$$

which is the CSP with $(s_1^{-1} d^{-1}(c_1) w^{-1} d(c_2) s_1, CE(y_1, y_2))$ with solution $d(x)$

$$
CE(y_3, y_4) = y_3^{-1} y_4 = d^{-1}(x) s_1^{-1} d^{-1}(c_3) d(c_4) s_1 d(x)
$$

Note we have transformed the MSSDP into an MSCSP involving the shift operator because equations $k$ and $l$ are an MSCSP in $d(x)$.

B. Using an XSSS algorithm e.g. the USS technique compute the solutions $s'_1, s'_2 \in B_{n+1}$

$$
\delta_{n+1} CE(y_1, y_2) \delta_{n+1}^{-1} = s'_1 \delta_{n+1} s_1^{-1} d^{-1}(c_1) d(c_2) s_1 \delta_{n+1}^{-1} s'_1^{-1}
\delta_{n+1} CE(y_3, y_4) \delta_{n+1}^{-1} = s'_2 \delta_{n+1} s_1^{-1} d^{-1}(c_3) d(c_4) s_1 \delta_{n+1}^{-1} s'_2^{-1}.
$$

C. Put $S = (s'_1, s'_2, f, (y_1^{-1} \cdot ((y_1 \cdot d(s'_1^{-1}) \cdot s_1^{-1} \cdot d(c_1^{-1})) \cdot c_1 \cdot s_1^{-1}))\mid \Delta_{n+1}, |y_2^{-1} \cdot ((y_2 \cdot d(s'_2^{-1}) \cdot s_1^{-1} \cdot d(c_2^{-1})) \cdot c_2 \cdot s_2^{-1})\mid \Delta_{n+1})) = (s'_1, s'_2, f)$ and $M = \emptyset$.

D. Until a solution is found.

1. Choose a tuple $(t, u, l_t)$ from $S$ with the smallest $l_t$.
2. If $f = 0$ then output $t$ (here $t = u$) and

$$
w = y_1^{-1} \cdot ((y_1 \cdot d(s'_1^{-1}) \cdot s_1^{-1} \cdot d(c_1^{-1})).
$$

Note if $f = 0$ then by equations $j$ and $m$ we get the actual value of $w$.  

33
3. Otherwise for each $i = 1, \ldots, K$ and $j = 1, \ldots, L$, for some natural numbers $K$ and $L$.
   
   (i) Compute $t_i = t \cdot C_i$, $u_j = u \cdot C_j$ and $f$
   
   (ii) If $(t_i, u_j, f)$ belongs neither to $S$ nor to $M$ then to add it into $S$.
   
4. Remove the current pair $(t, u, f)$ from $S$ and add it to $M$.

3 An Algorithm for the Multiple Simultaneous Shifted Decomposition Problem Variant MSSDPv

**Definition.** The MSSDPv (multiple simultaneous SDP variant) is as follows. Let $n \geq 1$ be a fixed integer. For braids $w, x, y, c_i \in B_\infty$, $1 \leq i \leq n$ find braids $w, x \in B_\infty$ such that $y_i = wd(c_i)\sigma_1 d(x)$ where $y_i$ are publicly known and $w, x, c_i$ are secret.

Clearly the MSSDPv generalises the MSCSPv. The generalisation of the MSSDPv for the MSSDP is similar to the generalisation of the MSCSPv for the MSCSP.

There are no efficient solutions for solving the MSSDPv one reason for this is it would mean the SCSP would be easy. In this appendix we propose a solution for the MSSDPv.

Consider when $n = 4$ in the MSSDPv and $c_1 \neq c_2, c_3 \neq c_4$ and consider when $n = 4$ in the MSSDP and $c_1 \neq c_2, c_3 \neq c_4$, this is the system of equations

\[
\begin{align*}
y_1 &= wd(c_1)\sigma_1 d(x), \quad y_2 = wd(c_2)\sigma_1 d(x) \quad \text{(n)} \\
y_3 &= wd(c_3)\sigma_1 d(x) \quad \text{and} \quad y_4 = wd(c_4)\sigma_1 d(x)
\end{align*}
\]

Again using CE functions we transform the MSSDPv into a shifted MSCSPv type problem very efficiently.

As the MSSDPv generalises the MSCSPv we can use our algorithm for the MSSDPv to attack the AAGL protocol in [1].

**Algorithm 3.1- Heuristic Algorithm for Solving the MSSDPv**

**INPUT:** The example of the MSSDPv which are equations n above.

**OUTPUT:** A solution of the MSSDPv.

**COMPUTATION:**

A. First we compute

\[
CE(y_1, y_2) = y_1^{-1}y_2 = d^{-1}(x)\sigma_1^{-1}d^{-1}(c_1)d(c_2)\sigma_1 d(x)
\]

which is the CSP with $(\sigma_1^{-1}d^{-1}(c_1)w^{-1}wd(c_2)\sigma_1CE(y_1, y_2))$ with solution $d(x)$

\[
CE(y_3, y_4) = y_3^{-1}y_4 = d^{-1}(x)\sigma_1^{-1}d^{-1}(c_3)\sigma_1 d(x).
\]

Note we have transformed the MSSDPv into the MSCSPv involving the shift operator. We observe equations o and p are an MSCSPv in $d(x), \sigma_1^{-1}d^{-1}(c_1)d(c_2)\sigma_1, \sigma_1^{-1}d^{-1}(c_3)d(c_4)\sigma_1$ and so we then use algorithm 1.1 to recover the “middle elements” $\sigma_1^{-1}d^{-1}(c_1)d(c_2)\sigma_1$ and $\sigma_1^{-1}d^{-1}(c_3)d(c_4)\sigma_1$. If the “middle elements” have not been recovered then the algorithm has failed and stops here. Otherwise if the algorithm 1.1 fails to find $d(x)$ and hence $x$ we goto step B to attempt to find $x$ in a different way.
B. Use the algorithm 2.1 to solve the MSSDP to attempt to find \( x \).
C. If \( x \) has not been found the algorithm has failed otherwise the algorithm is successful.

4 Attack on Dehornoy’s Shifted Conjugacy Protocol

We can apply our algorithm for the MSSDP to attack the shifted CSP based protocol [14] in the following specific scenario. We refer the reader to [14] for details of the DSC protocol.

Alice’s authenticates to Bob using \( r \) as described in [14]. Then Bob reuses Alice’s \( r \) as his random value in the commitment with another user (may be not Alice) because he assumes it is safe to do so. Following the notation in [14] of the DSC protocol we can attack that protocol: by letting in the MSSDP \( x = w^{-1} \), \( w = r \), \( c_1 = x_A, c_2 = x'_A, c_3 = x_B, c_4 = x'_B \); recall from [14] that Alice’s commitment \( (x_A, x'_A) = (r * p, r * p') \) and \( (p, p') \) are publicly known; similarly the notation \( (x_B, x'_B) \) refers to Bob commitment. Then when we have found a value for \( r \) using our algorithm for the MSSDP we would expect (because of equation j) this value to be the actual value of \( r \) used in the protocol instead of a different value satisfying the two equation MSCSP in \( r \). Then when we have this correct value of \( r \) we can recover Alice’s or Bob’s secret key as follows. \( r * s \) is publicly known, the attacker waits for \( b = 1 \) then computes

\[
(r^{-1} \cdot (r * s) \cdot dr \cdot \sigma^{-1}) = ds,
\]

noting we can invert the shift operator on \( ds \) we recover \( s \) hence breaking the scheme.

5 Comparison of Our Attack with the Longrigg-Ushakov Attack

We summarize the differences between our new attack and the LU attack in [16].

i) The LU attack is based on solving the CSP. Our attack is based on the MSCSP.
ii) The LU attack finds Alice’s secret key \( s \) or an equivalent key for \( s \) in a different way compared to our attack. Our attack, when it is used to attack the DSC protocol, finds the random braid \( r \) in the commitment (and not a equivalent value for \( r \)) then using this value of \( r \) we recover \( s \).
iii) The LU attack is for a general scenario. Our attack is for a more specific scenario which implies an MSCSP in \( r \).
iv) Our simple variation of our attack described above is based on solving the MSCSP at (S1) and the similar step in the LU attack is based on CSP. Because the CSP is known to be harder than the MSCSP, hence our attack will succeed in recovering \( s \) when the LU attack fails in some scenarios.
v) The LU attack cannot be used to solve the MSSDP but the LU attack can be used to solve the SCSP. Our attack solves the MSSDP but does not solve the version of the SCSP used in the DSC protocol because the \( CE \) function does not exist for \( s \).
vi) The LU attack only solves one equation which is the SCSP. Our attack can be extended in a straightforward way to solve the MSSDPv, MSSDP for any $n$: to derive the attack for a system of $n$ equations is similar to the examples for $n = 4$ given above. We now give suggestions for selecting secure parameters.

To defend against the attacks for the AAGL attack we suggest the following.

i) Ensure if possible that elements of the centraliser of $u'_i$ are hard with the CSP $(k, zkz^{-1})$.

ii) Ensure that elements of the centraliser of the form $zkz^{-1}$ of $u'_i$ cannot be feasibly computed.

iii) The TTP algorithm may be modified with different choices of $BL, BR$ so that larger generators are used with the constraint of using RFID tags.

To defend against attack for the DSC scheme we suggest the following.

i) The example of the MSSDP in the Dehornoy scheme is hard.

ii) Choose the centraliser $C$ is large.