The Anshel-Anshel-Goldfeld Key Exchange on the Symmetric Group

Ryan Edelstein

Abstract

The Anshel-Anshel-Goldfeld key exchange is a public key cryptographic protocol using non-abelian groups. Its security relies on the computational difficulty of solving the simultaneous conjugacy problem over such groups. In this paper, we discuss the implementation and security of the key exchange using the permutation group on n elements. We provide a graph-based algorithm to crack this key exchange and show that the exchange is unsafe.

Contents

1 Introduction

A *public key encryption protocol* is a system by which two users obtain a shared key by passing data. This is called *transmission* and requires that data be *encoded* in some rep-

resentation on a computer. The assumption is that an adversary is able to view every transmission of data and knows the process by which the two users are generating the key, but the adversary is still unable to determine the shared key easily. The notion of difficulty in cryptography refers to the computational complexity of cracking the code. A problem is considered easy if it can be done in polynomial running time.

Let $G = \langle X \rangle$ be a group with ordered generating set $X = \{x_1, x_2, \ldots, x_n\}$. Elements in the group are encoded as words in the alphabet $X^{-1} \cup X$. The *Anshel-Anshel-Goldfeld cryptographic protocol* [A] on this group is defined by the following key exchange.

- **Step 1.** Alice selects her *private key*

  $$a = x_{i_1}^{\alpha_1} x_{i_2}^{\alpha_2} \ldots x_{i_k}^{\alpha_k} \in G,$$

  where $x_{i_1}, x_{i_2}, \ldots, x_{i_k} \in X$ and $\alpha_1, \alpha_2, \ldots, \alpha_k \in \mathbb{Z}$. Bob selects his private key

  $$b = x_{i_1}^{\beta_1} x_{i_2}^{\beta_2} \ldots x_{i_j}^{\beta_j} \in G,$$

  where $x_{i_1}, x_{i_2}, \ldots, x_{i_j} \in X$ and $\beta_1, \beta_2, \ldots, \beta_j \in \mathbb{Z}$.

- **Step 2.** Alice transmits

  $$a^{-1}Xa = (a^{-1}x_1a, a^{-1}x_2a, \ldots, a^{-1}x_na) \in G^n$$

  Bob transmits

  $$b^{-1}Xb = (b^{-1}x_1b, b^{-1}x_2b, \ldots, b^{-1}x_nb) \in G^n.$$

- **Step 3.** Clearly, Alice can compute

  $$b^{-1}ab = (b^{-1}x_{i_1}b)^{\alpha_1}(b^{-1}x_{i_2}b)^{\alpha_2} \ldots (b^{-1}x_{i_k}b)^{\alpha_k}$$

  and Bob can compute

  $$a^{-1}ba = (a^{-1}x_{i_1}a)^{\beta_1}(a^{-1}x_{i_2}a)^{\beta_2} \ldots (a^{-1}x_{i_k}a)^{\beta_k}.$$

  From this, Alice and Bob can compute the *common key*, which is the *commutator of a and b*

  $$[a, b] = a^{-1}b^{-1}ab$$

  .

This key exchange is based around the computational difficulty of calculating uncovering conjugates in non-abelian groups. In particular, much research has been done on the implementation of the Anshel-Anshel-Goldfeld key exchange over braid groups. These groups pose difficulty for attacks due to being infinite and group elements varying in the space required for representation. In fact, there is at this point no proven polynomial-time

code-breaking algorithm that Eve can use to break the key exchange over a general braid group.

Braid groups have many similarities to the finite permutation groups, both in how the groups are generated and how operations are done within the groups. Although the braid group is significantly more complex, the similarities motivate the question of whether general permutation groups would have any security in the commutator key exchange.

In this paper, we will perform an analysis of the Anshel-Anshel-Goldfeld key exchange using the symmetric group as a platform. We will start with a discussion of the security of the protocol using arbitrary non-abelian groups and then proceed to focus on the symmetric group. In the final section, we will propose a graph based algorithm for cracking the key exchange and show that the symmetric group does not provide security for this cryptographic protocol. Our main result is as follows.

Theorem 1.1 (Main Result)**.** *An adversary Eve can crack the Anshel-Anshel-Goldfeld key exchange protocol on S_n with ordered generating set X in time*

- **a)** $\mathcal{O}(n^2)$ *if* $X = \{(1\ 2\ \dots\ n), (1\ 2)\}$
- **b)** $\mathcal{O}(n^3)$ *if* $X = \{(1\ 2), (2\ 3), \dots, (n-1\ n)\}$
- **c)** $\mathcal{O}(n^3 \log_2 n)$ *if X is any generating set*

It follows from this theorem that Eve can crack the key exchange in polynomial time and thus the protocol is unsafe.

2 Security

Let G be a group.

Problem 2.1 (Simultaneous Conjugacy Problem)**.** *Given two conjugate tuples of elements $A = (a_1, a_2, \dots, a_n) \in G^n$ and $B = (b_1, b_2, \dots, b_n) \in G^n$, find an element $t \in G$ such that*

$$t^{-1} a_i t = b_i \quad \forall\, i \in \{1, \dots, n\}.$$

We call the element t as above a *solution to the Simultaneous Conjugacy Problem for the tuples A and B* and abbreviate the problem as the SCP. A SCP is usually written as $t^{-1}At = B$ which in the notation of Problem 2.1 means

$$\{t^{-1}a_1 t, t^{-1}a_2 t, \dots, t^{-1}a_n t\} = \{b_1, b_2, \dots, b_n\}$$

An adversary Eve is able to see the public group G, the generator $X = \{x_1, x_2, \dots, x_n\}$, and the two transmissions $a^{-1}Xa$ and $b^{-1}Xb$. Eve can crack the code if she can solve the

simultaneous conjugacy problem for the tuples X and $a^{-1}Xa$ and the tuples X and $b^{-1}Xb$, thus uncovering a and b.

It is possible that a conjugacy equation does not have a unique solution, that is

$$\exists\ a_1, a_2 \in G, \quad a_1 \neq a_2$$

such that

$$a_1^{-1}xa_1 = a_2^{-1}xa_2$$

In fact it is possible that the entire simultaneous conjugacy problem does not have a unique solution, so

$$\exists\ a_1, a_2 \in G, \quad a_1 \neq a_2$$

such that

$$a_1^{-1}x_ia_1 = a_2^{-1}x_ia \quad \forall i \leq n$$

However we show that although Eve may recover a different solution to the conjugacy search problem, she will still be able to use her solution to obtain the secret key.

Definition 2.2. The *centralizer* of a subgroup $H \leqslant G$, denoted $C_G(H)$, is the set of elements of G that commute with every element in H.

$$C_G(H) = \{g \in G : gh = hg, \ \forall h \in H\}$$

When referring to the centralizer of the whole group we call it the *center* of G and denote $Z(G)$.

Lemma 2.3. *If $s, t \in G$ are two solutions to the simultaneous conjugacy problem for X and Y, then*

$$ts^{-1} \in Z(G)$$

.

Proof. Let $X = \{x_1, x_2, \ldots, x_k\}$ be an ordered generating set for a group G and $Y = \{y_1, y_2, \ldots, y_k\}$ a conjugate of X. Suppose $s, t \in G$ are two solutions to the simultaneous conjugacy problem for X and Y, so

$$s^{-1}Xs = t^{-1}Xt.$$

X generates G, so $\forall g \in G$, we have

$$s^{-1}gs = t^{-1}gt$$

and thus

$$ts^{-1}g = gts^{-1}.$$

Therefore $ts^{-1} \in Z(G)$. □

It follows that the common key obtained will not be affected by different solutions to the simultaneous conjugacy problem.

Corollary 2.4. *If Eve can solve the two simultaneous conjugacy search problems and obtain solutions a, b that are not the original private keys, she will still correctly compute the commutator.*

Proof. Suppose Eve solves the simultaneous conjugacy problem and obtains solutions a_2, b_2 for the secret keys, where a_1, b_1 are the original private keys: Let $c_a = a_2 a_1^{-1}$ and $c_b = b_2 b_1^{-1}$ Eve calculates

$$[a_2, b_2] = a_2^{-1} b_2^{-1} a_2 b_2 = (c_a a_1)^{-1} (c_b b_1)^{-1} (c_a a_1)(c_b b_1) = a_1^{-1} b_1^{-1} a_1 b_1 = [a_1, b_1].$$

So the non-unique solution to the simultaneous conjugacy problem does not impact Eve's ability to obtain the common key. □

Further, for certain groups there is a guarantee that the solution to the simultaneous conjugacy problem is unique:

Corollary 2.5. *Let X be an ordered generating set for a group G and Y a conjugate of X. If the center of G is trivial then there is exactly one solution to the simultaneous conjugacy problem for X and Y.*

Proof. Suppose $s, t \in G$ are two solutions to the simultaneous conjugacy problem for X and Y, then Lemma 2.3 gives that $ts^{-1} \in Z(G) = \{e\}$. Clearly $t = s$. □

It also is necessary to understand the relationship between two solutions of a general conjugacy search problem.

Lemma 2.6. *For some conjugates $a, b \in G$ let $g \in G$ be a solution to the conjugacy search problem for a and b. Then, the set of all solutions to the conjugacy search problem $t^{-1}at = b$ is $C_G(a)g$.*

Proof. Let t be a solution to the conjugacy search problem for a and b, so $t^{-1}at = g^{-1}ag$. Group operations give that there exists $h \in G$ such that $t = hg$. So

$$g^{-1}h^{-1}ahg = g^{-1}ag$$

and therefore $h^{-1}ah = a$ and thus $h \in C_G(a)$ and $t \in C_G(a)g$.

Additionally, if given $t = hg \in C_G(a)g$, then

$$t^{-1}at = g^{-1}h^{-1}ahg = g^{-1}ag.$$

□

This tells us that the number of solutions to a conjugacy problem in the notation of Lemma 2.6 is $|C_G(a)|$.

Given this relationship between conjugates of an element, we can use the following lemma to best understand how many possible conjugates of an element exist.

Lemma 2.7. *For a group G and element $a \in G$, the set of conjugates $a^G = \{g^{-1}ag : g \in G\}$ has the same cardinality as $G/C_G(a)$.*

Proof. For each $x \in a^G$, there exists $g_x \in G$ such that $g_x^{-1}ag_x = x$. We define the following natural function and show that it is a bijection:

$$h : a^G \to G/C_G(a), \quad h(x) = C_G(a)g_x$$

First, we show that h is independent of the choice of g_x. If $k^{-1}ak = x$ and $k \neq g_x$, then $k = cg_x$ for some $c \in C_G(a)$. Then, since $C_G(a)$ is a subgroup,

$$h(x) = C_G(a)k = C_G(a)cg_x = C_G(a)g_x.$$

Next, observe that if $x, y \in a^G$ and $h(x) = h(y)$, then $C_G(a)g_x = C_G(a)g_y$. It follows that $g_x = cg_y$ for some $c \in C_G(a)$ and thus $x = y$ and so h is injective.

Additionally, it is clear that h is surjective. Any coset $C_G(a)g$ is mapped to by $h(g^{-1}ag)$.

□

Corollary 2.8. *If G is finite, then $|C_a(G)| = |G|/|a^G|$.*

3 Symmetric group

The *symmetric group* on n elements, noted as S_n, is the set of all permutations of n elements. A permutation $\sigma \in S_n$ is a bijection from Z_n to Z_n. We commonly define a member of the symmetric group by its *cycle decomposition*. If for some $k, i < n$, we have that $\sigma^k(i) = i$, then σ contains a *cycle of length k*. We then define this cycle as

$(a_0\ a_1\ \ldots\ a_{k-1})$, where $\sigma(a_j) = a_{j+1(mod\ k)}$

Thus, for any $\sigma \in S_n$, we must have a set of $m \leq n$ disjoint cycles $(\alpha_1, \alpha_2, \ldots, \alpha_m)$ of lengths $(j_1, j_2, \ldots, j_m)$ where for each α_i, we have

$\alpha_i = (a_0\ a_1\ \ldots\ a_{j_i-1})$ and $\sigma(a_p) = a_{p+1(mod\ j_i)}$

In the Anshel-Anshel-Goldfeld key exchange, Eve's goal is to solve the simultaneous conjugacy problem on a generating set. Specifically, Eve wants to solve $a^{-1}Xa = Y$ for a where she knows the generating set X and the conjugate Y. But what are the generating sets for S_n?

3.1 Generating sets of S_n

For a group G, a *generating set* X is a set such that any element of the group can be formed by products of the elements of the generating set. Specifically, for each $g \in G$

$$g = x_1^{\epsilon_1} x_2^{\epsilon_2} \dots x_k^{\epsilon_k}$$

where each $x_i \in X$ and $\epsilon_i = \pm 1$. If X is a generating set for G, we say X generates G and write $\langle X \rangle = G$.

Theorem 3.1 (Common generators of S_n)**.** *The following are generating sets for S_n:*

$X = \{(1\ 2), (2\ 3), \dots, (n-1\ n)\}$

$Y = \{(i\ j) | i, j \in \{1, 2, \dots, n\}\}$

$Z = \{(1\ 2\ 3\ \dots\ n), (1\ 2)\}$.

Proof. First, observe that Y is an obvious generator of S_n. If $\alpha = a_1\ a_2\ \dots\ a_k \in S_n$ where each a_i is a disjoint cycle of length j_i, then a cycle $a_i = (a_{i,1}\ a_{i,2}\ \dots\ a_{i,j_i})$ can be written as

$$a_i = (a_{i,1}\ a_{i,2})(a_{i,2}\ a_{i,3})\ \dots\ (a_{i,j_i-1}\ a_{i,j_i}).$$

Clearly, this is a product of elements of Y, so $\langle Y \rangle = S_n$. Next, we can show that both X and Z can generate Y and thus generate S_n. First, observe that for any $(i\ i+1) \in X$, $(i+1\ i+2)(i\ i+1)(i+1\ i+2) = (i\ i+2)$.

In fact, for any $(i\ j) \in S_n$ we have $(j\ j+1)(i\ j)(j\ j+1) = (i\ j+1)$. Thus, for any $(i\ j) \in Y$, it is straightforward to go from $(i\ i+1)$ to $(i\ j)$ via products of elements in X. Thus, every element of Y can be made through products in X and thus $\langle X \rangle = \langle Y \rangle = S_n$.

Finally, observe that $(1\ 2\ \dots\ n)(1\ 2)(1\ 2\ \dots\ n)^{-1} = (2\ 3)$ and that in fact for any $i \leq n$ we have $(1\ 2\ \dots\ n)(i\ i+1)(1\ 2\ \dots\ n)^{-1} = (i+1\ i+2)$. Thus, we can form any element of X via products in Z by the formula

$$(i+1\ i+2) = (1\ 2\ \dots\ n)^i (1\ 2)(1\ 2\ \dots\ n)^{-i}$$

so $\langle X \rangle = \langle Y \rangle = \langle Z \rangle = S_n$. □

Observe that for the above generating sets of S_n, we have that $X \subset Y$. Since they generate the same group, it appears that some elements of Y are "unnecessary". We formalize this idea next.

Definition 3.2. A generating set X of a group G is said to be *redundant* if there exists a proper subset $Y \subsetneq X$ such that Y also generates G.

Theorem 3.3. *Any non-redundant generating set of S_n has at most $n \log_2 n$ elements.*

Proof. Let $X = \{x_1, x_2, \ldots, x_k\}$ be a non-redundant generating set of S_n. Then, it is clear that for each $1 < i < k$,

$$\langle x_1, x_2, \ldots, x_i \rangle \subsetneq \langle x_1, x_2, \ldots, x_{i+1} \rangle$$

Define $H_i = \langle x_1, x_2, \ldots, x_i \rangle$ and $m_i = |H_i|$. It follows by Lagrange's theorem that

$$m_1 \mid m_2 \mid \ldots \mid X_i \mid X_{i+1} \mid \ldots \mid X_{k-1} \mid |S_n| = n!$$

Therefore, k is bounded from above by the number of non-distinct prime factor of $|S_n| = n!$. The number of prime factors of $n!$ is bounded from above by $\log_2 n!$ since 2 is the smallest prime factor possible. Since $n! < n^n$, we have that $k < \log_2 n^n = n \log_2 n$. □

Therefore, any generating set of S_n with more than $n \log_2 n$ elements contains redundant elements that can be discarded. Thus for this key exchange we can consider that Eve must solve the simultaneous conjugacy problem for a tuple of less than $n \log_2 n$ elements.

Remark: Whiston [C] proved that the size of non-redundant generating sets of S_n is in fact bounded by $n - 1$. The proof is beyond the scope of this paper, and we will see that the slightly worse upper bound provided in our elementary proof is not so important when considering the security of the key exchange.

3.2 Solving conjugacy problems in S_n

If the group G in the Anshel-Anshel-Goldfeld key exchange is the symmetric group S_n, how secure is the exchange? This question is analogous to the question of solving the simultaneous conjugacy problem in S_n. First, we note that for any $\sigma \in S_n$, we can write σ as it's cycle decomposition $\sigma = (a_1\ a_2\ \ldots\ a_{k_1})(a_{k_1+1}\ a_{k_1+2}\ \ldots\ a_{k_2})\ldots$. We can show the following lemma about the relationship between conjugate elements:

Lemma 3.4. *Let σ be an element of S_n. Suppose that σ has a cycle decomposition of $\sigma = (a_1\ a_2\ \ldots\ a_{k_1})(a_{k_1+1}\ a_{k_1+2}\ \ldots\ a_{k_2})\ldots$. Then, for every $\tau \in S_n$, the conjugation of σ by τ satisfies*

$$\tau^{-1}\sigma\tau = (\tau^{-1}(a_1), \tau^{-1}(a_2)\ \ldots\ \tau^{-1}(a_{k_1}))(\tau^{-1}(a_{k_1+1})\ \tau^{-1}(a_{k_1+2})\ \ldots\ \tau^{-1}(a_{k_2}))\ldots$$

Proof. Looking at one disjoint cycle, observe that $\tau(\tau^{-1}(a_i)) = a_i$ and $\sigma(a_i) = a_{i+1}$.

So $\tau^{-1}(\sigma(\tau(\tau^{-1}(a_i)))) = \tau^{-1}(\sigma(a_i)) = \tau^{-1}(a_{i+1})$.

So the conjugation takes $\tau^{-1}(a_i)$ to $\tau^{-1}(a_{i+1})$, proving the lemma. □

This lemma provides the motivation for classifying the solutions to the conjugacy search problem. We now show which elements can be conjugates of each other:

Corollary 3.5. *If $\alpha, \beta \in S_n$ then there exists $\tau \in S_n$ such that $\beta = g^{-1}\alpha g$ if and only if α and β have the same cycle structure.*

Proof. ($\Rightarrow$)It is clear from the lemma that if the two cycles are conjugate then they have the same cycle structure.

($\Leftarrow$)Let $\alpha, \beta \in S_n$ such that they have the same cycle structure $(l_1, l_2, l_3, \ldots, l_k)$, so $\alpha = a_1\ a_2\ a_3\ \ldots\ a_k$ and $\beta = b_1\ b_2\ b_3\ \ldots\ b_k$ where each a_i and each b_i are disjoint cycles of length l_i.

Define for each $i \leq k$ $a_i = (a_{i,1}\ a_{i,2}\ \ldots\ a_{i,l_i})$ and $b_i = (b_{i,1}\ b_{i,2}\ \ldots\ b_{i,l_i})$.

Define τ such that $\tau^{-1}(a_{c,d}) = b_{c,d}$. Therefore,

$$\tau^{-1}a_i\tau = (\tau^{-1}(a_{i,1})\tau^{-1}(a_{i,2})\ldots\tau^{-1}(a_{i,l_i})) = (b_{i,1}b_{i,2}\ldots b_{i,l_i})$$

Therefore τ conjugates α into β. □

We now have the information needed to solve the conjugacy search problem in S_n.

Given two permutations $x, y \in S_n$ that are conjugate, how can Eve determine $a \in S_n$ such that $a^{-1}xa = y$? It turns out this is not very hard. Let

$x = x_1\ x_2\ \ldots\ x_k$ be the cycle decomposition of x and

$y = y_1\ y_2\ \ldots\ y_k$ be the cycle decomposition of y.

Given that x and y are conjugate, let's further assume that this is an ordered cycle decomposition such that for each i, x_i and y_i are cycles of the same length. Note: this is by no means a unique ordering. For example, if both elements are composed of two 3-cycles, there are clearly multiple orderings that satisfy the above condition.

Based on the cycle decomposition we can easily define the conjugating element a. For some cycles $x_i = (x_{i,1}\ x_{i,2}\ \ldots\ x_{i,j})$ and $y_i = (y_{i,1}\ y_{i,2}\ \ldots\ y_{i,j})$ where j is the length of the cycle, we have that

$$a(y_{i,p}) = x_{i,p}, \quad \forall p \leq j.$$

This simple expression is well defined to solve the conjugacy search problem.

3.3 Solving the simultaneous conjugacy problem in S_n

Although we have shown an efficient way to obtain a solution to a single conjugacy search problem, we have shown that such a solution is by no means unique. In fact, it follows immediately from Lemma 2.6 that the conjugacy search problem $t^{-1}at = b$ has $|C_{S_n}(a)|$ solutions. Let's understand what this means in the group S_n.

Lemma 2.7 gives that $|C_{S_n}(a)| = |S_n|/|a^{S_n}|$, and we know $|S_n| = n!$, we just need to figure out the cardinality of the conjugate class of a. Corollary 3.5 allows us to specify that this is exactly the number of elements in S_n that have the same cycle structure as a.

We will define the cycle structure of a in the following way: Let $(k_1, k_2, \ldots, k_n)$ denote the number of disjoint cycles of each length. Specifically, a has k_i disjoint cycles of length i. We can show the following calculation for the size of a^{S_n}.

Theorem 3.6. *Let $a^{S_n} \subset S_n$ denote conjugacy class for a. If a is composed of k_i disjoint cycles of length m_i for each $k_i \in \{k_1, k_2, \ldots, k_j\}$ and $m_i \in \{m_1, m_2, \ldots, m_j\}$, then*

$$|a^{S_n}| = \frac{n!}{\prod_{i=1}^{j} k_i! m_i^{k_i}}$$

Proof. We are looking to count the number of ways to arrange n elements with the same cycle structure as a. There are $n!$ potential ways to arrange the n elements. For each cycle of length m_i, we divide by m_i since there are m_i equivalent orders of the cycle. Additionally, given k_i cycles of order m_i, there are $k_i!$ ways to arrange the cycles, so we divide by $k_i!$.

□

Therefore, we obtain the result that

$$|C_{S_n}(a)| = \prod_{i=1}^{j} k_i! m_i^{k_i}$$

So, we know the number of solutions to the conjugacy search problem given an element a. But, when is this number large and when is it small?

Let's observe a few examples. If a is the identity, then there are no cycles, or there are n cycles of length 1. Therefore, $|C_{S_n}(a)| = n!$. Of course, this is trivial and unimportant for our purposes, as the identity would not be included in a realistic generating set.

If a is composed of a single cycle of length n, then $|C_{S_n}(a)| = n$. If a is composed of $n/2$ cycles of length 2, then $|C_{S_n}(a)| = (\frac{n}{2})! 2^{\frac{n}{2}}$.

Why is this important? After all, we proved that a non-unique solution will still allow for Eve to uncover the key. However, it is easily verifiable that S_n for $n \geq 3$ has a trivial center, and thus by Corollary 2.5 the simultaneous conjugacy search problem has exactly one solution. Furthermore, although it is clearly straightforward to find a solution to a conjugacy search problem for one element, solving the simultaneous conjugacy problem requires finding the intersection of the solutions to the conjugacy search problem for each element of the generating set. As we have seen, these solution sets can be large.

3.4 Proposed strategies for solving the SCP for generating sets of S_n

Example 3.7. In S_8 solve the simultaneous conjugacy problem in $a^{-1}Xa = Y$ where $X = \{(1\ 2), (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8)\}$ and $Y = \{(4\ 5), (4\ 5\ 1\ 3\ 2\ 6\ 8\ 7)\}$.

It is clear that X and Y are conjugate by observing the cycle structure. Theorem 3.6 gives us that $a^{-1}(1\ 2\ 3\ 4\ 5\ 6\ 7\ 8)a = (4\ 5\ 1\ 3\ 2\ 6\ 8\ 7)$ has 8 solutions and $a^{-1}(1\ 2)a = (4\ 5)$ has $2 \cdot 6! = 1440$ solutions. The intersection is exactly one element, so we can start by finding solutions to $a^{-1}(1\ 2\ 3\ 4\ 5\ 6\ 7\ 8)a = (4\ 5\ 1\ 3\ 2\ 6\ 8\ 7)$ and then plug in each of the 8 solutions into $a^{-1}(1\ 2)a = (4\ 5)$ to obtain the unique solution $a = (4\ 1\ 3)(5\ 2)(8\ 7)$.

Solving this conjugacy problem is easy. In general for a generating set of this form we can use this algorithm to solve the SCP in $\mathcal{O}(n)$ steps.

This solution shows a nice case where it is easy to solve the simultaneous conjugacy problem.

But, this case is easy because of the fact that there is an element of the generator, namely (1 2 3 4 5 6 7 8), that has very few elements in its center, and thus very few solutions to the conjugacy search problem. But there are other generating sets where this is clearly not the case. Starting with the other standard generating sets:

Example 3.8. $X = \{(1\ 2), (2\ 3), \ldots, (n-1\ n)\}$ is an ordered generating set for S_n. Thus, the solution set to the conjugacy search problem for each $(i-1\ i)$ contains $2 \cdot (n-2)!$ elements. Thus, it is unrealistic to simply calculate all of the solutions for a single element and then attempt to match with the others. However, the structure of the generating set allows for a second strategy:

Starting with the $t^{-1}(1\ 2)t = (a\ b)$, it is clear that either $t(a) = 2$ or $t(b) = 2$. Simply check whether a or b is equal to either c or d in $t^{-1}(2\ 3)t = (c\ d)$, and that one must be mapped by t to 2. It is straightforward then to read off the entire unique solution.

Once again, this strategy allows for Eve to recover a solution in $\mathcal{O}(n)$ time. Clearly, the same strategy can be applied to the generating set $Y = \{(i\ j) : i, j \leq n\}$, and thus it is also easily solved.

Given that the three most natural generating sets yield easily solvable simultaneous conjugacy problem, it begins to appear that Eve will have no difficulty cracking the key exchange on the symmetric group. However, consider the following example:

Example 3.9. Let $X = \{(1\ 2)(3\ 4)(5\ 6)\ldots(n-1\ n), (2\ 3)(4\ 5)\ldots(n-2\ n-1), (1\ 2)\}$

First, looking at this generating set, it is clear that solving for the entire solution set for one element is not realistic. The transposition has $2 \cdot (n-2)!$ solutions, and the other two have order $\frac{n}{2}! \cdot 2^{\frac{n}{2}}$ solutions.

Additionally, there is not an immediately obvious strategy to read off the solution piece by piece as in Example 3.9, as there is not way of differentiating between two transpositions to identify them.

However, the strategy can be used in some way. Notice that for $t^{-1}(1\ 2)t = (a\ b)$, either $t(a) = 2$ or $t(b) = 2$. We can assume $t(a) = 2$ for this, and then if wrong return and do the same strategy knowing that $t(b) = 2$. Since we assume $t(a) = 2$, we can then observe the conjugate of $(2\ 3)(4\ 5)\ldots(n-2\ n-1)$ and recognize that there must exist a transposition containing a of the form $(a\ x)$ where $t(x) = 3$. Further, in the conjugate of $(1\ 2)(3\ 4)(5\ 6)\ldots(n-1\ n)$, we can find a transposition containing x of the form $(x\ y)$ where $t(y) = 4$. This procedure can be repeated until the entire conjugating element is uncovered or a contradiction is reached, which would indicate an incorrect choice of $t^{-1}(2)$.

These examples give the feeling that, although it may be impossible to compare all solutions to the SCP for each element of the generating set, it is possible to piece by piece

uncover the conjugate. In each case, it requires some sort of smart traversal through the different elements of the generating set.

4 A polynomial time algorithm for solving the SCP in S_n

Much of the following section is based off of the paper by Andrej Brodnik, Aleksander Malnik, and Rok Pozar on *A subquadratic algorithm for the simultaneous conjugacy problem* [B].

4.1 The SCP in S_n as a graph colouring problem

Definition 4.1. A *permutation digraph* on a set of permutations $a = \{a_1, a_2, \ldots, a_d\}$ is a pair $G_a = (V_a, E_a)$ of vertexes and edges where: $V_a = \{1, 2, \ldots, n\}$ and E_a is the set of ordered pairs (i, a_k) where $i \in V$ and $a_k \in a$. An edge e in G_a thus is defined for each pair $e = (i, a_k) \in E_a$ where the initial vertex is $ini(e) = i$, the terminal vertex is $ter(e) = a_k(i)$, and the color of the edge is $c(e) = k$.

Definition 4.2. A *color isomorphism* between two permutation digraphs G_a and G_b is a pair of bijections $\phi_V : V_a \to V_b$ and $\phi_E : E_a \to V_b$ preserving the graph structure. In particular, we require that for each edge $e \in E_a$, $\phi_V(ini(e)) = ini(\phi_E(e))$, that $\phi_V(ter(e)) = ter(\phi_E(e))$, and that $c(e) = c(\phi_E(e))$. Two d-tuples a and b are conjugate if and only if G_a and G_b are color isomorphic.

Definition 4.3. A *labeling function* is a function $\mathcal{L} : \mathcal{G} \to L$, where $\mathcal{G}$ is the set of permutation digraphs and L is a set of strings on a fixed size alphabet. In this algorithm we will consider a *canonical labeling* in which for two permutation digraphs $G_a, G_b \in \mathcal{G}$, we have that $\mathcal{L}(G_a) = \mathcal{L}(G_b)$ if and only if G_a and G_b are color isomorphic.

So, we can solve our simultaneous conjugacy problem $t^{-1}at = b$, if we can find a color isomorphism between G_a and G_b.

4.2 A canonical labeling algorithm

To find a color isomorphism, we define the following labeling algorithm and then show it is canonical. Given a permutation digraph $G_a = (V_a, E_a)$ and a vertex $v \in V_a$, we perform a breadth-first-search starting at v and using the color of edges as priority for which vertex we visit next from a given vertex. We relabel the vertices in order of when we visit them, storing our relabeling function as $\gamma_v : V \to V$. The resulting relabeling is the permutation digraph G_a^v, where we define $a^v = \gamma_v a \gamma_v^{-1}$.

Example 4.4. Let $a = \{(1\ 2\ 3\ 4\ 5\ 6), (1\ 4\ 6)\}$ and $b = \{(1\ 5\ 3\ 2\ 6\ 4), (1\ 2\ 4)\}$. It is straightforward to see that these two tuples are conjugate. The first figure shows the

Algorithm 1 Relabel a single permutation digraph

Require: G_a permutation digraph, $v \in G_a$
  Initialize empty queue Q
  $Visited = \{v\}$
  Initialize empty array γ_v of size n
  ▷ This array will hold the vertex isomorphism from G_a to G_{a^v}
  $\gamma_v[v] = 1$ ▷ We start with the chosen vertex labeled as 1
  Enqueue v in Q
  while $|Visited| < n$ **do**
    Dequeue the first item from Q into u
    for $k \leftarrow 1$ **to** d **do**
      if $a_k(u) \notin Visited$ **then**
        Add $a_k(u)$ to $Visited$
        $\gamma_v[a_k(u)] = |Visited|$
        Enqueue $a_k(u)$ in Q
      end if
    end for
  end while
  $a^v = \gamma_v a \gamma_v^{-1}$
  Return a^v and γ_v

permutation digraph b. Calling the relabeling algorithm on this graph with the start vertex 1 returns the graph in Figure 2. This is clearly the same as the standard representation of G_a as in Figure 3, so the color isomorphism has been obtained.

Running the algorithm on a permutation digraph with each of the n initial vertexes gives n permutation digraphs colour isomorphic to G_a and the corresponding n colour isomorphisms. Further, for a permutation digraph G_a we define the *code* to be

$$C(G_a) = a_1(1)a_1(2)\dots a_1(n)a_2(2)\dots a_2(n)\dots a_d(1)a_d(2)\dots a_d(n)$$

This is a string of length dn on the alphabet of size n. If we define a comparison operator $<$ on this alphabet where $C(G_a) < C(G_b)$ if $C(G_a)$ is lexicographically before $C(G_b)$, then we can define the following labeling function.

$$\mathcal{L} : \mathcal{G} \to L, \quad \mathcal{L}(G_a) = min\{C(G_{a^v}) : v \in V_G\}$$

We now wish to show that $\mathcal{L}$ is the canonical labeling function and that it yields the desired colour isomorphism.

Theorem 4.5. *The labeling function $\mathcal{L}$ is canonical.*

Proof. First we show that if G_a, G_b are two permutation digraphs and $\mathcal{L}(G_a) = \mathcal{L}(G_b)$, then G_a is colour isomorphic to G_b. Let u, w be vertices such that $C(G_{a^u}) = \mathcal{L}(G_a)$

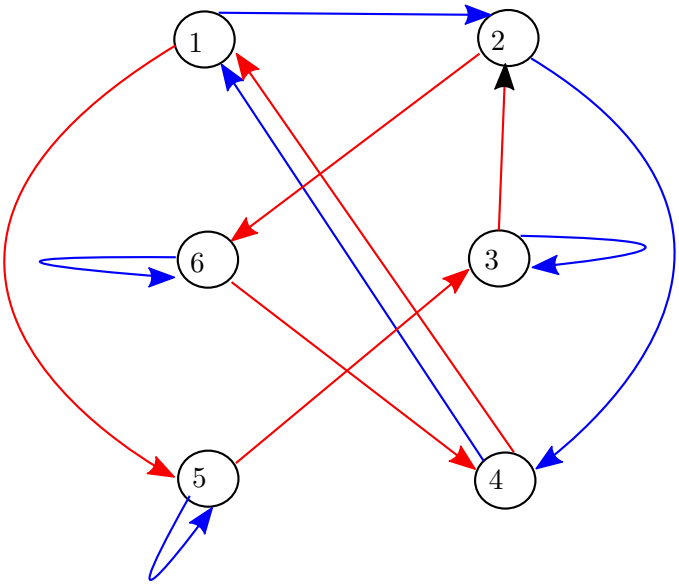

Figure 1: Permutation digraph G_b.

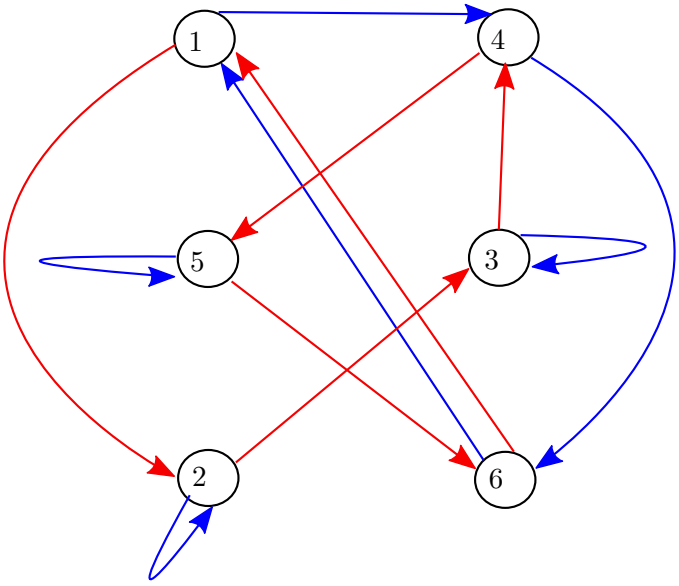

Figure 2: Permutation digraph of G_{b^1}.

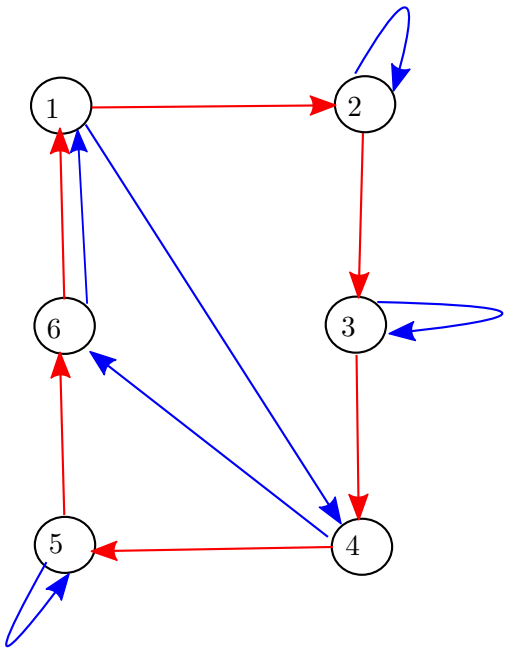

Figure 3: Permutation digraph of G_a.

and $C(G_{b^w}) = \mathcal{L}(G_b)$. Further, let γ_u and γ_w be the associated isomorphisms. Clearly $C(G_{a^u}) = C(G_{b^w})$ implies that $G_{a^u} = G_{b^w}$. So, G_a is isomorphic to G_b and the colour isomorphism is realized by $\gamma_w \gamma_u^{-1}$.

In the other direction, if f is a colour isomorphism from G_a to G_b , let u be a vertex satisfying $C(G_{a^u}) = \mathcal{L}(G_a)$. Let $w = f(u)$, and thus $G_{a^u} = G_{b^w}$, so $C(G_{a^u}) = C(G_{b^w})$. We look to show that $C(G_{b^w}) = \mathcal{L}(G_b)$. Suppose this is not true, so there exists a vertex $z \neq w$ such that $C(G_{b^z}) < C(G_{b^w})$. Then, $G_{a^{f^{-1}(z)}} = G_{b^z}$ and thus $C(G_{a^{f^{-1}(z)}} = C(G_{b^z})$. But then $C(G_{a^{f^{-1}(z)}} < C(G_{b^w}) = C(G_{a^u})$, a contradiction. □

The algorithm thus solves the simultaneous conjugacy problem. We now analyze the runtime.

Theorem 4.6. *The canonical label for a permutation digraph on n vertices and of degree d can be computed in $\mathcal{O}(dn^2)$.*

Proof. The labeling algorithm runs in $\mathcal{O}(dn)$ time to compute G_a^v. Computing the label then can be done in $\mathcal{O}(n)$ time. This must be done for each of n vertices, so the overall runtime is $\mathcal{O}(dn^2)$. □

4.3 Cracking the key exchange on S_n in $\mathcal{O}(n^3 logn)$ time

Proof of Theorem 1.1

Proof. For some ordered generating set X of size d, Eve must solve the simultaneous conjugacy problems $a^{-1}Xa = Y$ and $b^{-1}Xb = Z$ for Alice and Bob's secret keys, a and b, respectively. For each such SCP, taking the first as example, Eve can transform X and Y into permutation digraphs. Such a transformation can be done in $\mathcal{O}(dn)$ time. Given these two permutation digraphs, Eve can use the canonical labeling algorithm to compute the labeling of G_X and G_Y in $\mathcal{O}(dn^2)$ time and thus uncover the colour isomorphism between the two, thus yielding the secret key a. Performing this procedure twice obtains both a and b and thus Eve can compute the commutator $a^{-1}b^{-1}ab$ in $\mathcal{O}(dn^2)$ time. We now turn to the 3 cases presented in the theorem.

- **a)** If $X = \{(1\ 2\ \ldots\ n), (1\ 2)\}$, then $d = |X| = 2$, so the algorithm is $\mathcal{O}(n^2)$.
- **b)** If $X = \{(1\ 2), (2\ 3), \ldots, (n-1\ n)\}$, then $d = |X| = n - 1$, so the algorithm is $\mathcal{O}(n^3)$.
- **c)** If X is any generating set, we have shown that after removing redundant elements $d = |X| < n \log_2 n$. So the algorithm is $\mathcal{O}(n^3 \log n)$.

□

5 Concluding remarks

In the original paper by Anshel, Anshel, and Goldfeld the key exchange is performed using two public ordered generating sets of two subgroups $S_A = \langle \mathbf{a} \rangle$ and $S_B = \langle \mathbf{b} \rangle$. In this case, Alice selects her private key

$$a = x_{i_1}^{\alpha_1} x_{i_2}^{\alpha_2} \ldots x_{i_k}^{\alpha_k} \in S_A,$$

where $x_{i_1}, x_{i_2}, \ldots, x_{i_k} \in \mathbf{a}$ and $\alpha_1, \alpha_2, \ldots, \alpha_k \in \mathbb{Z}$. Bob selects his key similarly using $\mathbf{b}$. Alice transmits $a^{-1}\mathbf{b}a$ and Bob transmits $b^{-1}\mathbf{a}b$. It is no different from before how Alice and Bob can each compute the common key $a^{-1}b^{-1}ab$.

It also is no different for Eve in that if she can solve the simultaneous conjugacy problems $a^{-1}\mathbf{b}a = Y$ and $b^{-1}\mathbf{a}b = Z$ she can crack the code. Eve certainly can construct permutation digraphs for, for example, $\mathbf{b}$ and Y. It is additionally straightforward to see that our algorithm continues to work for determining the conjugating element for these subgroups. Thus it is analogous to show that the key exchange is no more secure when using subgroups of S_n.

Clearly the symmetric group is not a suitable platform group for the commutator key exchange. Considering that the braid group has yet to be proven to not be secure, it is clear that the additional structure on the braid group causes the increase in complexity.

Appendix

The purpose of a public key cryptographic protocol is to pass encoded information over a shared channel. Here, we simulate the use of the Anshel-Anshel-Goldfeld key exchange with the symmetric group across a channel where an adversary Eve sees all transmissions. The following is documentation for a Python program simulating the exchange. The code and full documentation can be found at https://github.com/ryandedelstein/AAG_Python.

The code is structured over 4 main files.

1. Exchange: The exchange file holds the main "channel" for the key exchange. The key exchange is initialized by presenting generators for the PublicInformation class. When Alice or Bob transmit data to each other, they do it by posting it in this class. This gives makes such data public.

2. Alice and Bob: Alice and Bob access the PublicInformation channel in order to get the set of generators and each compute a private key. This is done within the class, so it is not visible to any other objects. They then transmit the conjugate of the generator set and get each others transmissions. They then privately compute the shared key.

3. Eve: Eve accesses the PublicInformation channel and gets the set of generators, as well as each of Alice and Bob's conjugates of the generators. Eve then looks to find the private keys based on this information. The work of this is done by solving the simultaneous conjugacy problem using the relabel function.

4. AAG_sim.py: This is the driver code in which users can input data representing a generating set and then observe Eve cracking the subsequent key exchange.

References

[A] I. Anshel, M. Anshel, D. Goldfeld, An Algebraic Method for Public-Key Cryptography, *Mathematical Research Letters* **6** (1999), 1-5.

[B] A. Brodnik, A. Malnik, R. Pozar, A subquadratic algorithm for the simultaneous conjugacy problem, *Slovenian Research Agency* (2020).

[C] J. Whiston, Maximal Independent Generating Sets of the Symmetric Group, *Journal of Algebra* **232** (2000) 255-268.