9-30-2015

Algorithmic properties of poly-Z groups and secret sharing using non-commutative groups

Bren Cavallo
Graduate Center, City University of New York

ALGORITHMIC PROPERTIES OF POLY-$\mathbb{Z}$ GROUPS AND SECRET SHARING USING NON-COMMUTATIVE GROUPS

by

Bren Cavallo

A dissertation submitted to the Graduate Faculty in Mathematics in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

2015

Abstract

ALGORITHMIC PROPERTIES OF POLY-$\mathbb{Z}$ GROUPS AND SECRET SHARING USING NON-COMMUTATIVE GROUPS

By: BREN CAVALLO
Advisor: Delaram Kahrobaei

Computational aspects of polycyclic groups have been used to study cryptography since 2004 when Eick and Kahrobaei [21] proposed polycyclic groups as a platform for conjugacy based cryptographic protocols.

In the first chapter we study the conjugacy problem in polycyclic groups and construct a family of torsion-free polycyclic groups where the uniform conjugacy problem over the entire family is at least as hard as the subset sum problem. We further show that the conjugacy problem in these groups is in NP, implying that the uniform conjugacy problem is NP-complete over these groups. This is joint work with Delaram Kahrobaei from [13]. We also present an algorithm for the conjugacy problem in groups of the form $\mathbb{Z}^n \rtimes_\phi \mathbb{Z}$ that can be seen in [14].

We continue by studying automorphisms of poly-$\mathbb{Z}$ groups and successive cyclic extensions of arbitrary groups. We study a certain kind of extension that we call "deranged", and show that the automorphisms of the resulting group have a strict form. We also show that the automorphism group of a group obtained by iterated extensions of this type contains a non-abelian free group if and only if the original base group does. Finally we show that it is possible to verify that a finitely presented by infinite cyclic group is finitely presented by infinite cyclic, but that determining that a general finitely presented group is finitely generated by infinite cyclic is undecidable. We then discuss implications the latter result has for calculating the Bieri-Neumann-Strebel invariant (see [5]). This is joint work with Jordi Delgado, Delaram Kahrobaei, Ha Lam, and Enric Ventura and is currently in preparation [12].

In the final chapter we discuss secret sharing schemes and variations. We begin with classical secret sharing schemes and present variations that allow them to be more practical. We then present a secret sharing scheme due to Habeeb, Kahrobaei, and Shpilrain [30]. Finally, we present an original adjustment to their scheme that involves the shortlex order on a group and allows less information to be transmitted each time a secret is shared. Additionally, we propose additional steps that allow participants to update their information independently so that the scheme remains secure over multiple rounds. This is joint work with Delaram Kahrobaei from [15].

Preface

The use of non-commutative group theory in cryptography is a recent development. Its root can perhaps be traced to the work of Magyarik and Wagner in 1985 [64], but the theory became more widespread after Anshel, Anshel, and Goldfed introduced their key exhange protocol in 1999 [1]. See [51] for an in-depth overview on non-commutative group based cryptography. Computational aspects of polycyclic groups have been used to study cryptography since 2004 when Eick and Kahrobaei [21] proposed polycyclic groups as a platform for conjugacy based cryptographic protocols. As such, the study of algorithmic properties of infinite polycyclic groups, particularly evaluating the computational complexity of algorithms, is becoming more relevant.

We start in **Chapter 1** where we investigate the conjugacy problem in polycyclic groups. We begin by introducing key elements of the theory of polycyclic groups that will be used in this thesis along with a brief introduction to NP-completeness and the subset sum problem. We proceed by introducing and constructing a family of groups whose conjugacy problem is at least as hard as a variation on the subset sum problem that we call the twisted subset sum problem. We then study conjugacy further and show that these groups have the property that conjugacy can be performed and checked quickly while viewing elements as exponent vectors. Finally, we prove that the twisted subset sum problem is NP-complete, implying that the conjugacy problem over the family of groups we construct is NP-complete. This is joint work with Delaram Kahrobaei from [13] published in the *International Journal of Algebra and Computation.* Finally we discuss a polynomial time solution to the conjugacy problem in free abelian by infinite cyclic groups from [14] published in the *Reports @ SCM.*

In **Chapter 2** we focus more on infinite cyclic extensions of groups and prove structural results about their automorphism groups. Following [8] we define a certain type of automorphism such that extending by it leaves the base group fixed. This leads to explicit forms for the automorphisms of said extensions. We further show that such extensions extend groups in an essentially unique way and show that the presence of non abelian free groups in the automorphism group of iterated extensions is determined solely by the automorphism group of the base group. We then discuss an algorithm that verifies that a group is finitely presented by infinite cyclic but show that membership in the class of such groups is undecidable. We then review the Bieri-Neumann-Strebel invariant and show implications of its calculation based on the previous results. This is joint work with Jordi Delgado, Delaram Kahrobaei, Ha Lam, and Enric Ventura and is currently in preparation [12].

In **Chapter 3** we discuss secret sharing using non-commutative groups. We begin with an introduction to classical secret sharing and provide a formal definition of a secret sharing scheme. Next,

we present classical secret sharing schemes due to Blakley [6] and Shamir [60] and verifiable variations due to Feldman [24] and Pedersen [53]. We also present a proactive secret sharing scheme due to Herzberg et. al. [32]. Afterwards, we introduce a secret sharing scheme due to Habeeb, Kahrobaei, and Shpilrain [30] and introduce the platform group, namely small cancellation groups. Finally, we present joint work with Delaram Kahrobaei from [15] appearing in *Contemporary Mathematics*, that is a modification of the Habeeb-Kahrobaei-Shpilrain secret sharing scheme, that is more efficient. We also add additional steps, so that information sent at the beginning of the process is secure after multiple secrets are shared.

Acknowledgments

I would like to first and foremost thank my adviser Delaram Kahrobaei. Delaram has been the most amazing and supportive adviser any graduate student could ask for. Without her motivation and enthusiasm, I surely wouldn't be where I am now. I also would like to thank Vladimir Shpilrain for many insightful discussions regarding group theory and cryptography. I am truly grateful for the time Vladimir and Delaram have spent with me as mentors and collaborators, and for the opportunity they gave for me to work with them. I also would like to thank my fellow students at the Graduate Center, with whom I have had so many amazing experiences, and from whom I have learned so much. Finally, I would like to thank Melvyn Nathanson, Alexey Ovchinnikov, and Benjamin Steinberg for serving on my defense committee and providing me with many helpful comments on this manuscript.

I acknowledge the support of the Office of Naval Research through a grant by Delaram Kahrobaei and Vladimir Shpilrain, as well as a PSC CUNY research grant through Delaram Kahrobaei, and a research assistantship through the Center for Logic, Algebra, and Computation (C-LAC) for the 2012 - 2013 academic year.

New York B.B.C.
2015

For my parents

Table of Contents

Chapter 1

The Conjugacy Problem in Poly-$\mathbb{Z}$ Groups

1.1 Introduction

Given a finite presentation representing a group G, the conjugacy decision problem for G takes as input words u and v in the generators of G and asks if there exists x in the generators of G such that xux^{-1} represents the same element in G as v. For the purposes of cryptography, we often consider the search variant: given conjugate $u, v \in G$, find $x \in G$ that conjugates u to v. The first instance of the conjugacy search problem in cryptography appeared in the seminal paper of Anshel, Anshel, and Goldfeld [1] in which the authors introduced a key exchange protocol where the security is largely based on the conjugacy search problem. Since then the conjugacy search problem has become a central part of non-commutative group based cryptography. Other cryptographic protocols that rely on conjugacy search problem include [35, 36, 37, 43].

Polycyclic groups were originally proposed as a platform group for AAG by Eick and Kahrobaei in [21]. Since then Garber, Kahrobaei, and Lam [27] offered computational evidence that indicates that a common heuristic attack against AAG, the length based attack, was not effective on polycyclic groups. It is worth also mentioning that the length based attack was effective in breaking AAG over braid groups (see [28, 49]) which were the proposed platform for AAG from [1].

In this chapter we introduce a family of polycyclic groups over which the uniform conjugacy decision and search problems are NP-complete. What we mean by this, is that there is a family of polycyclic groups, G_n, where G_n has Hirsch length $2n + 1$ and the conjugacy decision and search problems are NP-complete as we vary the word length and n. As such, a polynomial time algorithm that solves either conjugacy problem in these groups would imply P = NP. This is joint work with Delaram Kahrobaei from [13].

The chapter begins by discussing polycyclic groups and in particular poly-$\mathbb{Z}$ groups. We then define the complexity class NP, the notion of NP-completeness and the theory of one of the most famous NP-complete problems, the Subset Sum Problem. We continue by explicitly constructing the groups G_n and show that the uniform conjugacy decision problem over these groups is NP-complete. This involves showing that instances of the conjugacy problem are at least as hard as the Subset Sum Problem and that the conjugacy problem is in NP. We finish by studying the conjugacy problem in groups of the form $\mathbb{Z}^n \rtimes_M \mathbb{Z}$ and introduce a practical algorithm for both the conjugacy decision and search problems. The theorems and proofs and much of the style of exposition for the first nine sections is drawn from [13], whereas the theorems and proofs for 1.10 is drawn from [14].

It is often the case that a solution to the conjugacy decision problem often provides a solution to the search problem. As such, for the remainder of this dissertation, when we refer to the conjugacy problem, we often mean solving both problems if the context is not clear.

1.2 The Complexity of Conjugacy Problem in Polycyclic Groups

Before we present the main results of this chapter, we begin with a brief introduction as to what is known about the conjugacy decision and search problems in polycyclic groups. We first note that the conjugacy problem for finitely presented groups is in general undecidable. See [16] in which the authors construct a finitely presented group with undecidable conjugacy problem which contains an index two subgroup with solvable conjugacy problem. On the other hand, the conjugacy search problem, given conjugate $u, v \in G$ find $x \in G$ that conjugates u to v, can be solved in any recursively presented group for any inputs (see page 12 from [51]).

Polycyclic groups were shown to have solvable conjugacy decision problem independently by Formanek [25] and Remeslennikov [54]. More precisely, they showed that any virtually polycyclic group is conjugacy separable- if $u, v \in G$ are not conjugate, then there exists a finite homomorphic image in which the images of u and v are not conjugate. While their results show decidability, their papers do not provide an algorithm that is amenable to a complexity analysis. Moreover, the algorithm involves brute force computation, so it can be seen that it is inefficient in a worst case scenario.

The first practical algorithm for deciding conjugacy decision and search in polycyclic groups was given by Eick and Ostheimer (see [19, 22]). It was noted in [21] that the algorithm of Eick and Ostheimer likely has a high worst case complexity since it may involve computation of the unit group of an algebraic number field. Additionally, in [21], Eick and Kahrobaei demonstrated that the algorithm applied to conjugacy search takes large amounts of time even over groups of relatively low Hirsch length.

Andrew Sale in [55, 56] studied the geometry of conjugacy in polycyclic groups by studying their conjugacy length function:

$$CLF_G(n) = \max\{\min\{|w| : wu = vw\} :$$
$$|u| + |v| \leq n,\ u \,\text{and}\, v \,\text{are conjugate in}\, G\}$$

The polycyclic groups Sale studied were ones of the form $\mathbb{Z}^n \rtimes_\phi \mathbb{Z}^m$ where certain technical conditions were imposed on ϕ. He showed that when $m = 1$ the conjugacy length function is bounded from above by a linear function and for $m > 1$ that it is bounded from above by an exponential function. Due to the exponential growth of these groups, this implies that there is a exponential algorithm to solve both variants of the conjugacy problem in the case that $m = 1$ and doubly exponential when $m > 1$.

Altogether, much is still unknown about the complexity of the conjugacy decision and search problems over a general polycyclic group. In [21] and [27] the authors conjecture that the conjugacy problem is exponential over the family of polycyclic groups as the Hirsch length increases. In the remainder of this chapter, we provide evidence for their view by showing that if an algorithm for conjugacy that varies polynomially with respect to the Hirsch length exists, then P = NP. It is still unknown if there exists a polynomial time algorithm that solves conjugacy in a single general polycyclic group.

1.3 Polycyclic Groups

In this section we will give an overview of many of the basic properties polycyclic groups and split extensions. Many of these results and their proofs can be found in [18, 19].

Definition 1.3.1. *A group G is* polycyclic *if it has a subnormal series with cyclic quotients. Namely, G has subgroups $G_0, G_1, \cdots, G_n$*

such that

$$\{1\} = G_0 \lhd G_1 \lhd \cdots \lhd G_n = G$$

and G_{i+1}/G_i is a cyclic group.

1.3.1 Normal Forms

Lemma 1.3.2. *Given a polycyclic group G with subnormal series $\{1\} = G_0 \lhd G_1 \lhd \cdots \lhd G_n = G$, there exist elements $g_1, g_2, \cdots, g_n \in G$ such that for any word $w \in G$, $w = g_1^{e_1} g_2^{e_2} \cdots g_n^{e_n}$ for some $e_i \in \mathbb{Z}$.*

Proof. Let $\bar{g}_i$ generate the quotient G_i/G_{i-1} and let g_i denote a fixed pre-image of $\bar{g}_i$ in G_i. Given any word $w \in G$, w is equal to $w_{n-1}g_n^{e_n}$ for some $w_{n-1} \in G_{n-1}$ and $e_n \in \mathbb{Z}$. In fact, the image of w in the quotient G_n/G_{n-1} is $\bar{g}_n^{e_n}$. We continue by then writing $w_{n-1} = w_{n-2}g_{n-1}^{e_{n-1}}$, with $w_{n-2} \in G_{n-2}$, which then makes $w = w_{n-2}g_{n-1}^{e_{n-1}}g_n^{e_n}$. Iterating this computation down the subnormal series, we eventually end with a prefix $w_1 \in G_1$, but since G_1 is cyclic with generator g_1, we can write $w_1 = g_1^{e_1}$. Therefore $w = g_1^{e_1} g_2^{e_2} \cdots g_n^{e_n}$. □

We call such a representative of an element of G, its *normal form.* Notice that the choice of the g_i is not necessarily unique.

In addition to multiple choices of pre-images existing, a subnormal series with cyclic quotients is also not in general unique. For a polycyclic group G, we fix a subnormal series that we call the *polycyclic series* and fix a corresponding set of generators, $\{g_1, \cdots, g_n\}$, which we call its *polycyclic generating set.* We also call the process of converting a general word into its normal form representative, *collection*.

1.3.2 Semi-direct Products

Given groups, N and K, and a homomorphism $\phi : K \to \mathrm{Aut}(N)$ we can define the semi-direct product, $N \rtimes_\phi K$. As a set, $N \rtimes_\phi K$, is equal to $N \times K$, but the group structure is:

$$(n_1, k_1)(n_2, k_2) = (n_1\phi(k_1)(n_2), k_1k_2).$$

The multiplication in the first and second coordinates are multiplication in N and K respectively. As such, it can be seen that if ϕ sends every element of K to the identity on N, then the semi-direct product is the direct product.

If we let $N = \langle n_1, \cdots, n_l | R\rangle$ and $K = \langle k_1, \cdots k_p | Q\rangle$, then we can present $N \rtimes_\phi K$ as

$$\langle n_1, \cdots, n_l, k_1 \cdots, k_p | R, Q, k_j n_i k_j^{-1} = \phi(k_j)(n_i) \rangle$$

for $1 \leq i \leq l$ and $1 \leq j \leq p$.

Finally, if we let $G \simeq N \rtimes_\phi K$, then $G/N \simeq K$. This can be seen by noting that N is normal and inspecting the above presentation.

1.3.3 Poly-$\mathbb{Z}$ Groups

Of specific interest to us in this chapter are polycyclic groups where each quotient $G_i/G_{i-1} \simeq \mathbb{Z}$. We call such groups *poly-*$\mathbb{Z}$. To aid us with the study of these groups we prove the following lemma:

Lemma 1.3.3. *Let $H \lhd G$ such that $G/H \simeq \mathbb{Z}$. Also let t be the pre-image of generator for $\mathbb{Z}$. Then $G \simeq H \rtimes_\phi \mathbb{Z}$ where $\phi(t^k)(h) = t^k h t^{-k}$.*

Proof. First note that since H is normal in G, conjugation by t restricted to H is in $\mathrm{Aut}(H)$. As such, ϕ is well-defined.

Since $G/H = \mathbb{Z}$, we can write each element in G uniquely as ht^k for some $h \in H$ and $k \in \mathbb{Z}$. Our isomorphism $\psi : G \to H \rtimes_\phi \mathbb{Z}$ is given by

$$\psi(ht^k) = (h, t^k).$$

First, let us prove this map is a homomorphism.

$$\psi(h_1t^{k_1})\psi(h_2t^{k_2}) = (h_1, t^{k_1})(h_2, t^{k_2})$$

$$= (h_1t^{k_1}h_2t^{-k_1}, t^{k_1+k_2}) = \psi(h_1h_2't^{k_1+k_2})$$

where h_2' is the element of H given by $t^{k_1}h_2t^{-k_1}$. But

$$h_1t^{k_1}h_2t^{k_2} = h_1t^{k_1}h_2t^{-k_1}t^{k_1+k_2} = h_1h_2't^{k_1+k_2}.$$

Therefore the map is a homomorphism. Injectivity and surjectivity follow from inspecting the map.

□

This lemma shows that poly-$\mathbb{Z}$ groups are created by successive infinite cyclic extensions. If G is a poly-$\mathbb{Z}$ group with polycyclic sequence $\{1\} = G_0 \triangleleft G_1 \triangleleft \cdots \triangleleft G_n = G$, then $G \simeq G_{n-1} \rtimes_{\phi_{n-1}} \mathbb{Z}$ for some $\phi_{n-1} \in \mathrm{Aut}(G_{n-1})$. Proceeding iteratively, we see that

$$G \simeq ((\cdots((\mathbb{Z} \rtimes_{\phi_1} \mathbb{Z}) \rtimes_{\phi_2} \mathbb{Z}) \rtimes_{\phi_3} \cdots) \rtimes_{\phi_{n-2}} \mathbb{Z}) \rtimes_{\phi_{n-1}} \mathbb{Z}$$

where $\phi_i \in \mathrm{Aut}(G_i)$.

If we choose g_i as the generator of the i^{th} $\mathbb{Z}$ in the above iterated semi-direct product, we can make a presentation using the explicit description of a semi-direct product presentation above:

$$G = \langle g_1, g_2, \cdots, g_n | g_j g_i g_j^{-1} = \phi_{j-1}(g_i) \rangle \text{ for } 1 \leq i < j \leq n$$

The relations in this presentation indicate a simple collection algorithm for G. Start with the highest indexed generator, g_n, and apply the relation $g_n g_i = \phi_{n-1}(g_i) g_n$ until all of the g_n's have accumulated at the right end of the word. Notice that we are left with a word of the form $w g_n^k$ where w contains no g_n's. This is because $\phi_{n-1}(g_i)$ for $1 \leq i \leq n-1$ contains no g_n, since $\phi_{n-1} \in \mathrm{Aut}(G_{n-1})$. We then repeat the same process with g_{n-1} moving each g_{n-1} to the right using the identity $g_{n-1} g_i = \phi_{n-2}(g_i) g_{n-1}$ until we have accumulated all of them immediately to the left of the g_n^k we accumulated in the previous step. We then repeat the same process moving down in index until our word is in normal form. Moreover, if we want to multiply words in G and then collect, we simply concatenate the words, reduce, and then start this process. While this method is guaranteed to terminate, it is not the most practical. See [29] and [45] for other collection methods.

We also define the *Hirsch length* as the number of $\mathbb{Z}$'s in the semi-direct product formulation of the poly-$\mathbb{Z}$ group. The Hirsch length is an isomorphism invariant, so while different automorphisms in the construction of the poly-$\mathbb{Z}$ group may lead to isomorphic groups, the number of factors is necessarily the same.

For a given word $w = g_1^{k_1} \cdots g_n^{k_n}$ we define the length of w to be:

$$|w| = nK$$

where $K = \max_{1 \leq i \leq n}\{\log(|k_i|)\}$. As such, we consider the number of digits it takes to represent each exponent rather than the size of the exponent itself. This can be viewed the bitwise length of the unique exponent vector: $[k_1, \cdots, k_n]$. It is worth noting that this is different than the standard geodesic length in a group that is often used to measure algorithmic complexity. We find that this measure is useful in our case, since most practical applications, such as cryptography, use elements in normal form. We then take the length of a conjugacy problem to be sum of the lengths of its inputs.

1.4 NP - Completeness

We start this section with the formal definition of a decision problem.

Definition 1.4.1. *A* decision problem *is a pair* (L, I) *where* I *is the set of inputs to the decision problem, and* L *is the subset of* I *where the decision problem evaluates to "true"*

We call the set L a *language*. In this framework, the decision problem can be interpreted as checking if for some $x \in I$, that $x \in L$. In this way a decision problem is undecidable if the membership problem for L is undecidable.

Example: Consider the conjugacy problem in a poly-$\mathbb{Z}$ group, G. If we say that G has Hirsch length n, we can take $I = \mathbb{Z}^n \times \mathbb{Z}^n$ where we interpret $\mathbb{Z}^n$ as the set of exponent vectors for G. Then, $(x, y) \in I$ is in L, if there exists $a \in G$ such that $ag_xa^{-1} = g_y$, where g_x and g_y are the group elements corresponding to x and y in G.

After fixing a set of inputs, any decision problem can be completely interpreted as the set of strings for which the problem is satisfied. As such, it is customary to set $I = \{0, 1\}^*$, the set of strings made up of 0's and 1's of arbitrary length. By fixing a set of inputs, we can then talk about languages and corresponding decision problems interchangeably.

Definition 1.4.2. *A language, L, is said to be in* NP *if there exists an integer valued polynomial p, and a polynomial time Turing Machine, M, such that for all $x \in I$, $x \in L$ if and only if there exists a string u such that $|u| \leq p(|x|)$ and $M(x,u) = 1$.*

This definition can be interpreted to mean that a language is in NP if there are short "yes" answers that can be verified quickly. Such a u is called a *certificate.*

A central question in theoretical computer science, is whether or not any problem in NP can be answered with a polynomial time Turing machine. This is called the P versus NP problem.

NP admits a certain hierarchy under polynomial time reductions.

Definition 1.4.3. *A decision problem, Q, can be reduced to a decision problem, R, in polynomial time if there exists a polynomial time mapping, f, from inputs of Q to inputs of R, such that an input x is a "yes" instance of Q if and only if $f(x)$ is a "yes"* *instance of R. Such a mapping must also increase the lengths of instances at most polynomially. We write $Q \leq_p R$ to say that Q polynomial time reduces to R.*

It follows from the definition, that if $A \leq_p B$, that a polynomial time algorithm for B can be additionally used to solve A in polynomial time. One can also see that polynomial time reductions are transitive: $A \leq_p B$ and $B \leq_p C$ imply $A \leq_p C$.

Definition 1.4.4. *A decision problem, B, is* NP-complete *if $B \in NP$ and for all $A \in NP$, $A \leq_p B$.*

NP-complete problems can be thought of as the most difficult problems in NP since an algorithm that solves one can be used to solve any other problem in NP. Moreover, if there exists a polynomial time Turing machine that solves an NP-complete problem, then P = NP. The notion of NP-completeness was independently discovered by Stephen Cook and Leonid Levin in 1971 when they showed that the Boolean Satisfiability problem was NP-complete. In 1972, Richard Karp made a list of 21 NP-complete problems [40], showing that the notion of NP-completeness could be seen in problems having to do with integer linear programming and graph theory.

1.5 Subset Sum Problem

The *Subset Sum Problem*, or SSP, is the following: given a set of integers, $L = \{k_1, k_2, \cdots, k_n\}$, and an integer, M, determine if there exists a subset of L that sums to M. This can also be rephrased as determining if there is a solution to the equation:

$$k_1x_1 + \cdots + k_nx_n = M \text{ where } x_i \in \{0, 1\}.$$

The size of a subset sum problem is given by:

$$|SSP(L, M)| = nK$$

where $|L| = n$ and K is the maximal number of bits needed to represent any element of L or M. As such, this can be thought of as the minimal number of bits needed to represent the list and target for the subset sum problem. The SSP is well known to be NP-complete [40] as we vary n and K.

It is important that we view the lengths of the coefficients in binary. If we instead view the size of the coefficients to be their absolute values, there is a polynomial time solution utilizing dynamic programming (see [42]). Such a form for the coefficients is called *unary*. This is due to the fact that coefficients are not represented as strings of ones and zeros, but rather as strings of just one symbol where the length of the string corresponds to the coefficient. We provide a similar dynamic programming algorithm applied to a variation of the SSP in **Section 1.6**.

So far we have only referred to the decision variant of the SSP rather than its search variant: given that we have a subset with the desired sum, find such a subset. While NP-completeness only exists for decision problems, we can show that a polynomial time algorithm for a search variant of the SSP would imply that there is a polynomial time algorithm for the decision problem. As such, for much of this chapter, we slightly abuse concepts by referring to the NP-completeness of the SSP search problem. In the remainder of this section, we sketch a proof that a polynomial time algorithm for the SSP search problem implies existence of a polynomial time algorithm for the SSP.

If there is a polynomial time Turing machine, M, solving the SSP search problem, there then exists a polynomial $P(n)$ such that for any input x, the number of steps M takes on input x is less than or equal to $P(|x|)$. We create another Turing machine, M', that on input y performs the same steps as M for at most $P(|y|)$ steps. If M' halts and produces a subset before $P(|y|)$ steps, then M' outputs "yes". If M' has not yet finished, it instead outputs "no" since if there were a positive answer it would have found it already. As such, a polynomial time algorithm for the SSP search problem would imply a polynomial time algorithm for the decision variant.

1.6 The Twisted Subset Sum Problem

Given a list $L = \{k_1, \cdots, k_n\}$ and an integer M, the Twisted Subset Sum Problem ($TSSP$) is determining if the following equation has a solution:

$$k_n x_n (-1)^{x_1+x_2+\cdots x_{n-1}} +$$

$$k_{n-1}x_{n-1}(-1)^{x_1+x_2+\cdots+x_{n-2}} + \cdots + k_2x_2(-1)^{x_1} + k_1x_1 = M$$

where $x_i \in \{0, 1\}$. Note that we could trivially let x_i be any number and replace x_i with $x_i \bmod 2$ in the above equation. For the remainder of this chapter, we write $x'_i = x_i \bmod 2$. We also take

$$|TSSP(L, M)| = nK$$

where n and K are the same parameters is used in the determining the length of the SSP.

We now now provide a dynamic programming algorithm from [13] for the $TSSP$ which operates in polynomial time when the coefficients are taken in unary. We will proceed in a similar fashion to the standard dynamic programming algorithm for the subset sum problem, but due to the additional complexity of the $TSSP$, each entry in the array we create will either be a list of pairs of the form (T, ϵ) where $\epsilon = \{0, 1\}$ (T corresponds to the boolean "True") or will solely contain F (corresponding to the boolean "False") . The boolean entry in each tuple allows us to know if the corresponding target can be attained by the corresponding sub-list while the integer entry refers the parity of the number items we are including from the list to attain the target. If the entry in the array solely contains F, then such a target cannot be reached by the corresponding sub-list.

1. Allocate an empty two dimensional array, A, where the rows are labeled 1 through n and the columns are labeled $-S$ through S where $S = |k_1| + \cdots + |k_n|$.

2. Set $A(1, j) := (T, 1)$ if $j = k_1$, otherwise, set $A(1, j)$ to F. The 1 indicates that the next item will be subtracted. In the future we may also see $(T, 0)$ to indicate that the next item will be

added. This corresponds to the sum of the x_i modulo 2.

3. For each $A(i,j)$ where $i > 1$ do the following (we do this recursively so we compute row 2 before row 3):

   i. If $k_i = j$ add a $(T,1)$ into the list $A(i,j)$.

   ii. If $(T,\epsilon) \in A(i-1,j)$ where $\epsilon \in \{-1,1\}$, then add a (T,ϵ) into $A(i,j)$. As such, $A(i-1,j) \subset A(i,j)$.

   iii. If $(T,0) \in A(i-1,j-k_i)$ add $(T,1)$ to $A(i,j)$.

   iv. If $(T,1) \in A(i-1,j+k_i)$ add $(T,0)$ to $A(i,j)$.

   v. Otherwise set $A(i,j)$ equal to F.

To solve $TSSP(\{k_1,\cdots,k_n\},M)$, we fill the array and check if $A(n,M)$ contains either $(T,0)$ or $(T,1)$. Filling each box in the array takes polynomially many steps since the algorithm has already filled in all the boxes in the row below it and since filling in the first row is trivial. On the other hand, this algorithm is exponential time when the coefficients are taken in binary since the number of columns would be exponential in size with respect to the coefficients. We later show (**Section 1.9**) that the $TSSP$ is NP-complete when the coefficients are taken in binary.

1.7 The conjugacy problem over the groups G_n

We construct the group G_n as iterated extensions by $\mathbb{Z}$ as follows:

$$(\cdots((\mathbb{Z} \rtimes_{\phi_1} \mathbb{Z}) \rtimes_{\phi_2} \mathbb{Z}) \rtimes_{\phi_3} \cdots) \rtimes_{\phi_{2n}} \mathbb{Z}$$

where

$$\phi_{2i-1}(g_j) = \begin{cases} g_1^{-1} & \text{if } j = 1 \\ g_j & \text{otherwise} \end{cases}$$

and

$$\phi_{2i}(g_j) = \begin{cases} g_1 g_j & \text{if } j = 2i \\ g_j & \text{otherwise} \end{cases}$$

Multiplication in G is defined by the following identities: if j is even, $g_j g_1 = g_1^{-1} g_j$, $g_{j+1} g_j = g_1 g_j g_{j+1}$, and all other generators commute. In this section, we will show that any instance of the $TSSP$ with n indeterminates can be viewed as an instance of conjugacy in G_n. It can additionally be seen that this is a polynomial time reduction due to the definition of lengths of instances given in **Section 1.3** and **Section 1.5**.

The following lemma from [13], is a consequence of these multiplicative identities and will be used in the proofs of **Theorem**

1.7.2 and **Proposition 1.8.1** .

Lemma 1.7.1. *Let j be even and $a, b \in \mathbb{Z}$. Then $g_j^a g_1^b = g_1^{b(-1)^a} g_j^a$ and $g_{j+1}^a g_j^b = g_1^{ab'} g_j^b g_{j+1}^a$ where $b' = b \bmod 2$*

Proof. To prove the first identity, check that $g_j^a g_1^b = \phi_{j-1}^a(g_1^b) g_j^a = (\phi_{j-1}^a(g_1))^b g_j^a = (g_1^{(-1)^a})^b g_j^a = g_1^{b(-1)^a} g_j^a$. For the second identity, we first check the case $a = 1$. We then have $g_{j+1} g_j^b = (\phi_j(g_j))^b g_{j+1} = (g_1 g_j)^b g_{j+1}$. Now note that $(g_1 g_j)^2 = g_1 g_j g_1 g_j = g_1 g_1^{-1} g_j g_j = g_j^2$ from the first part of the lemma. Therefore if $b = 2l$ is even, then $(g_1 g_j)^b = ((g_1 g_j)^2)^l = (g_j^2)^l = g_j^b$. Using the same logic, if b is odd, $(g_1 g_j)^b = g_1 g_j^b$. Therefore, $g_{j+1} g_j^b = (g_1 g_j)^b g_{j+1} = g_1^{b'} g_j^b g_{j+1}$.

To continue with the more general case, $g_{j+1}^a g_j^b = g_{j+1}^{a-1} g_1^{b'} g_j^b g_{j+1} = g_1^{b'} g_{j+1}^{a-1} g_j^b g_{j+1}$ and after iterating this computation one obtains $g_{j+1}^a g_j^b = g_1^{ab'} g_j^b g_{j+1}^a$. □

We now show that $TSSP(\{k_1, k_2 \cdots, k_n\}, M)$, has a solution if and only if $g_3^{k_1} g_5^{k_2} \cdots g_{2n+1}^{k_n} \sim g_1^{-M} g_3^{k_1} g_5^{k_2} \cdots g_{2n+1}^{k_n}$. Our general strategy will be as follows: conjugate $g_3^{k_1} g_5^{k_2} \cdots g_{2n+1}^{k_n}$ by a generic word in G_n and and compute the normal form of the resulting word. First, note that from the multiplication rules above, it can be seen that any g_i where i is odd, commutes with $g_3^{k_1} g_5^{k_2} \cdots g_{2n+1}^{k_n}$. It can also be seen from the lemma that conjugating by one of the generators with even index does not introduce any generators with even index in the collected word. Therefore, without loss of generality, we can assume that our generic word is of the form $g_2^{x_1} g_4^{x_2} \cdots g_{2n}^{x_n}$ because adding in any generators with odd index doesn't affect the conjugated product.

Before we continue, we introduce the notation:

$$p(k_1, \cdots, k_n, x_1, x_2, \cdots, x_n) =$$

$$k_n x_n' (-1)^{x_{n-1}+x_{n-2}+\cdots+x_1} + k_{n-1} x_{n-1}' (-1)^{x_{n-2}+\cdots+x_1} + \cdots$$
$$+ k_2 x_2' (-1)^{x_1} + k_1 x_1'$$

Theorem 1.7.2. *The TSSP with n indeterminates is polynomial time reducible to the conjugacy problem in the group G_n of Hirsch length $2n+1$.*

Proof. Our overall strategy in the proof is to collect $g_3^{k_1} g_5^{k_2} \cdots g_{2n+1}^{k_n}$ conjugated by an arbitrary word in G_n, and show that we end up with $-p(k_1, \cdots, k_n, x_1, x_2, \cdots, x_n)$ as the exponent above g_1.

We proceed by induction on l where we conjugate by the last l syllables of the generic word. Rather than starting with $l = 1$ it may clarify the computation to start with $l = 2$. In this case we collect:

$$(g_{2n-2}^{x_{n-1}} g_{2n}^{x_n})(g_3^{k_1} g_5^{k_2} \cdots g_{2n+1}^{k_n})(g_{2n-2}^{x_{n-1}} g_{2n}^{x_n})^{-1}$$

Conjugating first by $g_{2n}^{x_n}$, we find:

$$g_{2n}^{x_n}(g_3^{k_1} g_5^{k_2} \cdots g_{2n+1}^{k_n}) g_{2n}^{-x_n} =$$

$$g_3^{k_1} g_5^{k_2} \cdots g_{2n}^{x_n}(g_{2n+1}^{k_n} g_{2n}^{-x_n}) =$$

$$g_3^{k_1} g_5^{k_2} \cdots g_{2n}^{x_n}(g_1^{k_n x'_n} g_{2n}^{-x_n} g_{2n+1}^{k_n}) =$$

$$g_1^{k_n x'_n (-1)^{x_n}} g_3^{k_1} g_5^{k_2} \cdots g_{2n+1}^{k_n} =$$

$$g_1^{-k_n x'_n} g_3^{k_1} g_5^{k_2} \cdots g_{2n+1}^{k_n}$$

Next we conjugate by $g_{2n-2}^{x_{n-1}}$:

$$g_{2n-2}^{x_{n-1}}(g_1^{-k_n x'_n} g_3^{k_1} g_5^{k_2} \cdots g_{2n-1}^{k_{n-1}} g_{2n+1}^{k_n}) g_{2n-2}^{-x_{n-1}} =$$

$$g_1^{-k_n x'_n (-1)^{x_{n-1}}} g_3^{k_1} g_5^{k_2} \cdots g_{2n-2}^{x_{n-1}}(g_{2n-1}^{k_{n-1}} g_{2n-2}^{-x_{n-1}}) g_{2n+1}^{k_n} =$$

$$g_1^{-k_n x'_n (-1)^{x_{n-1}}} g_3^{k_1} g_5^{k_2} \cdots g_{2n-2}^{x_{n-1}}(g_1^{k_{n-1} x'_{n-1}} g_{2n-2}^{-x_{n-1}} g_{2n-1}^{k_{n-1}}) g_{2n+1}^{k_n} =$$

$$g_1^{-k_n x'_n (-1)^{x_{n-1}} - k_{n-1} x'_{n-1}} g_3^{k_1} g_5^{k_2} \cdots g_{2n-1}^{k_{n-1}} g_{2n+1}^{k_n} =$$

$$g_1^{-p(k_{n-1},k_n,x_{n-1},x_n)} g_3^{k_1} g_5^{k_2} \cdots g_{2n-1}^{k_{n-1}} g_{2n+1}^{k_n}$$

We now induct and assume the result holds for $l = n-1$ and show it holds for $l = n$. In this case we have:

$$(g_2^{x_1} g_4^{x_2} \cdots g_n^{x_n})(g_3^{k_1} g_5^{k_2} \cdots g_{2n+1}^{k_n})(g_2^{x_1} g_4^{x_2} \cdots g_n^{x_n})^{-1} =$$

$$g_2^{x_1}(g_1^{-p(k_2,\cdots,k_n,x_2,\cdots,x_n)} g_3^{k_1} \cdots g_{2n+1}^{k_n}) g_2^{-x_1}$$

Conjugating by $g_2^{x_1}$ then yields:

$$g_2^{x_1}(g_1^{p(k_2,\cdots,k_n,x_2,\cdots,x_n)} g_3^{k_1} \cdots g_{2n+1}^{k_n}) g_2^{-x_1} =$$

$$g_1^{-p(k_2,\cdots,k_n,x_2,\cdots,x_n)(-1)^{x_1}} g_2^{x_1}(g_3^{k_1} g_2^{-x_1}) \cdots g_{2n+1}^{k_n} =$$

$$g_1^{-p(k_2,\cdots,k_n,x_2,\cdots,x_n)(-1)^{x_1}} g_2^{x_1}(g_1^{k_1 x_1'} g_2^{-x_1} g_3^{k_1}) \cdots g_{2n+1}^{k_n} =$$

$$g_1^{-p(k_2,\cdots,k_n,x_2,\cdots,x_n)(-1)^{x_1} - k_1 x_1'} g_3^{k_1} \cdots g_{2n+1}^{k_n} =$$

$$g_1^{-p(k_1,k_2,\cdots,k_n,x_1,x_2,\cdots,x_n)} g_3^{k_1} \cdots g_{2n+1}^{k_n}$$

Finally, note that $TSSP(\{k_1, \cdots, k_n\}, -M)$ has a solution if and only if $g_3^{k_1} g_5^{k_2} \cdots g_{2n+1}^{k_n} \sim g_1^{-M} g_3^{k_1} g_5^{k_2} \cdots g_{2n+1}^{k_n}$. As we noted at the beginning of the chapter, this reduction from $TSSP$ with n indeterminates to this instance of conjugacy in G_n is polynomial since the lengths of problems are only off by a linear polynomial.

$\square$

It is worth noting, that here is where our measure of length of group elements is crucial. If we had the length of elements in G be measured as $|k_1| + \cdots + |k_n|$, then the conjugacy problem would be exponentially larger than an instance of $TSSP$ and we would not have a polynomial time reduction. Also, as we pointed out in **Section 1.6**, the $TSSP$ has a polynomial time solution using dynamic programming when the elements in the list are taken in unary. This would imply that the instance of the conjugacy problem discussed in **Theorem 1.7.2** with the more standard measure of length of elements in normal form would have a polynomial time solution using the algorithm introduced in **Section 1.6**.

1.8 The Conjugacy Problem In G_n Is In NP

In this section we show that the conjugacy problem in the groups G_n can be verified in polynomial time. To do this we will find closed form expressions for conjugating a word by a power of a single generator. These closed form expressions will be effectively computable with group elements in their normal form and take polynomially many steps with respect to the length defined in **Section 1.3**. Since conjugation can be done by iteratively conjugating by powers of single generators, these closed forms show that conjugacy can be verified efficiently. What follows is the computation of the closed forms from [13].

When conjugating elements in G_n there are three cases to consider: conjugation by powers g_1, conjugation by powers of g_j with j even, and conjugation by powers of g_l where l is odd and larger than 1.

For the first case we collect

$$g_1^k(g_1^{k_1}\cdots g_{2n+1}^{k_{2n+1}})g_1^{-k}.$$

In this case, to collect the above word, we must bring the g_1^{-k} at the right end of the word all the way to the left. This can be done using the identities $g_jg_1 = g_1^{-1}g_j$ for j even and $g_lg_1 = g_1g_l$ for l odd. The first identity states that when we move the g_1^{-k} to the left we switch the sign of the exponent according to the parity of the exponents of the even indexed g_j. The second identity states that the odd g_l commute with g_1, and do not affect the collection process. Therefore we end up with:

$$g_1^{k+k_1-k(-1)^{k_2+k_4+\cdots+k_{2n}}} g_2^{k_2} \cdots g_{2n+1}^{k_{2n+1}} \tag{1.1}$$

The second case is then collecting

$$g_j^k (g_1^{k_1} \cdots g_{2n+1}^{k_{2n+1}}) g_j^{-k}$$

where j is even.

We first move the g_j^k right. Hopping over the $g_1^{k_1}$ may change the sign of the exponent, but after that, each g_i commutes with g_j for $i < j$. Therefore as a first step we end up with:

$$(g_1^{k_1(-1)^k} g_2^{k_2} \cdots g_j^{k+k_j} \cdots g_{2n+1}^{k_{2n+1}}) g_j^{-k}$$

In moving the g_j^{-k} to the left, the only thing that doesn't commute is g_{j+1}. To hop over $g_{j+1}^{k_{j+1}}$ we can use **Lemma 1.7.1** and get

$$g_1^{k_1(-1)^k} g_2^{k_2} \cdots g_j^{k+k_j} (g_{j+1}^{k_{j+1}} g_j^{-k}) \cdots g_{2n+1}^{k_{2n+1}} =$$
$$g_1^{k_1(-1)^k} g_2^{k_2} \cdots g_j^{k+k_j} (g_1^{k_{j+1}k'} g_j^{-k} g_{j+1}^{k_{j+1}}) \cdots g_{2n+1}^{k_{2n+1}}$$

Finally, we move the $g_1^{k_{j+1}k'}$ to the left to end up with:

$$g_1^{k_1(-1)^k+k_{j+1}k'(-1)^{k_2+k_4+\cdots+k_j+k}} g_2^{k_2} \cdots g_j^{k_j} g_{j+1}^{k_{j+1}} \cdots g_{2n+1}^{k_{2n+1}} \tag{1.2}$$

The third case is dealt with similarly to the second. When $l > 1$ is odd:

$$g_l^k(g_1^{k_1} \cdots g_{2n+1}^{k_{2n+1}})g_l^{-k} =$$

$$g_1^{k_1} \cdots (g_l^k g_{l-1}^{k_{l-1}})g_l^{k_l-k} \cdots g_{2n+1}^{k_{2n+1}} =$$

$$g_1^{k_1} \cdots (g_1^{kk'_{l-1}} g_{l-1}^{k_{l-1}} g_l^k)g_l^{k_l-k} \cdots g_{2n+1}^{k_{2n+1}} =$$

$$g_1^{k_1+kk'_{l-1}(-1)^{k_2+k_4+\cdots+k_{l-3}}} g_2^{k_2} \cdots g_{2n+1}^{k_{2n+1}} \tag{1.3}$$

Since conjugation is done by successively conjugating elements of the form of those in (1.1), (1.2), and (1.3) these closed forms can iteratively perform a general conjugation. Such a computation can be performed in polynomial time in terms of nK because computing the normal form after conjugation by each syllable can be done in polynomial time using the closed forms, and need only be performed $2n + 1$ times. This means that we can create a polynomial time verifier for the conjugacy problem in the G_n.

These normal forms also provide us with the following corollaries that describe conjugation in the group. The proofs for both statements can be seen directly be inspecting the closed forms above.

Proposition 1.8.1. *Let $u, v \in G_n$ where $u = g_1^{e_1} \cdots g_{2n+1}^{e_{2n+1}}$ and $v = g_1^{f_1} \cdots g_{2n+1}^{f_{2n+1}}$.*

i. $u \sim v$ implies $e_i = f_i$ for $i \geq 2$.

ii. Let $e_i = f_i$ for $i \geq 2$. If there exists an even $l-1$ such that $e_{l-1} = f_{l-1}$ is odd, the u and v are conjugate. In fact, one such element that conjugates u to v is:

$$g_l^{(f_1-e_1)(-1)^{e_2+e_4+\cdots+e_{l-3}}}$$

Note that part *ii* of the above corollary does not include the difficult cases of the conjugacy problem that we saw in **Section 1.7**.

We now show that for any two conjugate elements, $u, v \in G_n$, there exists a conjugator whose length is bounded polynomially by the lengths of u and v. Let $u = g_1^{e_1} g_2^{e_2} \cdots g_{2n+1}^{e_{2n+1}}$ and $v = g_1^{f_1} g_2^{e_2} \cdots g_{2n+1}^{e_{2n+1}}$ and $w = g_1^{x_1} \cdots g_{2n+1}^{x_{2n+1}}$ such that $wuw^{-1} = v$. In the case that there exists an odd exponent above an even indexed generator, we have a certificate of polynomial length from part *ii* of **Corollary 1.8.1**. Therefore, we can assume that for all j even, e_j is even.

If we assume that for all j even, e_j is even, we have from the closed form (1.3) that there exists a conjugator where $x_l = 0$ for all $l > 1$ odd. Additionally, we can take x_j to be 0 or 1 for j even by looking at the closed forms from (1.2). It remains to put bounds on x_1.

Let y be the exponent above g_1 conjugating by $g_2^{x_2} \cdots g_{2n+1}^{x_{2n+1}}$. Then by repeated applications of (1.2), $|y| \leq |e_1|+|e_3|+\cdots+|e_{2n+1}|$. From (1.1), conjugating by $g_1^{x_1}$ either increases the exponent by $2x_1$ or leaves it unchanged. Therefore if $f_1 = y$, we can take x_1 to be 0 and then clearly a certificate has length $O(n)$. Otherwise, $f_1 = y+2x_1$ implying that $|x_1| \leq |f_1|+|y| \leq |f_1|+|e_1|+|e_3|+\cdots+|e_{2n+1}|$. This means that the length of a certificate is bounded from above by $\log(|f_1|+|e_1|+|e_3|+\cdots+|e_{2n+1}|)+n$ which is of polynomial size in the length of the original conjugacy problem. This now shows that the conjugacy problem in G_n is in NP.

1.9 The Reduction of SSP to $TSSP$

In this section we show that $SSP \leq_p TSSP$. To make this easier we introduce another problem SSP' that is similar to SSP and in fact show that $SSP \leq_p SSP' \leq_p TSSP$. We define SSP' as follows: given a list of integers $\{k_1, \cdots, k_n\}$ and an integer M, decide if there exists a solution to the equation:

$$k_1x_1 + \cdots k_nx_n = M \quad \text{where} \quad x_1, \cdots, x_n \in \{-1, 0, 1\}$$

In this section, the proofs and theorems are taken from [13].

Lemma 1.9.1. *$SSP' \leq_p TSSP$.*

Proof. Consider $SSP'(\{k_1, \cdots, k_n\}, M)$ and $TSSP(\{0, k_1, 0, k_2, \cdots, 0, k_n\}, M)$, instances of SSP' and $TSSP$ respectively. In this case, we have that if $(x_1, x_2, \cdots, x_n)$ is a solution for $SSP'(\{k_1, \cdots, k_n\}, M)$ then $(y_1, y_2, \cdots, y_{2n})$ is a solution for the corresponding $TSSP$ problem where $y_{2i} = |x_i|$ and $y_{2i-1} = 1$ if:

$x_i = -1$ and $y_1 + \cdots + y_{2i-2}$ is even ($-k_i$ appears in the sum)

or

$x_i = 1$ and $y_1 + \cdots + y_{2i-2}$ is odd (k_i appears in the sum)

and is 0 otherwise. □

We now show that $SSP \leq_p SSP'$ by adapting a proof from the appendix of [42] by Kellerer, Pferschy, and Pisinger. Consider the following systems of equations:

$$\begin{cases} \sum_{i=1}^{n} k_i x_i = M \\ x_i \in \{0,1\} \end{cases} \tag{1.4}$$

$$\begin{cases} \sum_{i=1}^{n} k_i x_i = M \\ x_i + y_i = 1 \quad \text{for} \quad i = 1, \cdots, n \\ x_i, y_i \in \{-1,0,1\} \end{cases} \tag{1.5}$$

$$\begin{cases} \sum_{i=1}^{n} (4^{n-i} + 4^n k_i) x_i + \sum_{i=1}^{n} 4^{n-i} y_i = 4^n M + 4^n - 1 \\ x_i, y_i \in \{-1,0,1\} \end{cases} \tag{1.6}$$

First, note that (1.4) and (1.5) have equivalent solutions: any set of x_i that satisfies one will satisfy the other. The constraints $x_i + y_i = 1$ and $x_i, y_i \in \{-1,0,1\}$ prevent x_i from ever being -1. What is less apparent is that (1.5) and (1.6) have the same solution set. If this is the case, we can solve any instance of SSP, (1.4), using an algorithm that solves the equivalent SSP' (1.6). If we also show that the size of (1.6) is only polynomially larger than (1.4) then we will have in fact shown that $SSP \leq_p SSP'$, thus proving that both SSP' and $TSSP$ are NP-complete.

Proposition 1.9.2. *The following systems of equations have the same set of solutions:*

$$\begin{cases} \sum_{i=1}^{n} k_i x_i = M \\ x_1 + y_1 = 1 \\ x_i, y_i \in \{-1, 0, 1\} \end{cases} \tag{1.7}$$

$$\begin{cases} x_1 + y_1 + 4\sum_{i=1}^{n} k_i x_i = 4M + 1 \\ x_i, y_i \in \{-1, 0, 1\} \end{cases} \tag{1.8}$$

Proof. First note that anything that is a solution to (1.7) is a solution to (1.8). In the other direction, assume that $(x_1, \cdots, x_n, y_1)$ is a solution to (1.8). Note that this is equivalent to saying that:

$$4(\sum_{i=1}^{n} k_i x_i - M) = 1 - x_1 - y_1 \tag{1.9}$$

Using the fact that$-1 \leq 1 - x_1 - y_1 \leq 3$ we then get

$$-1 \leq 4(\sum_{i=1}^{n} k_i x_i - M) \leq 3 \tag{1.10}$$

Finally, since $\sum_{i=1}^{n} k_i x_i - M$ is an integer, (1.10) can only be satisfied if $\sum_{i=1}^{n} k_i x_i - M = 0$ implying that $(x_1, \cdots, x_n)$ is a solution for (1.7). □

Proposition 1.9.3. *The systems of equations* (1.5) *and* (1.6) *have the same set of solutions.*

Proof. As we did in the previous proposition, we combine the conditions $x_i + y_i = 1$ to the equation $\sum_{i=1}^{n} k_i x_i = M$ to obtain an instance of SSP' whose solution will yield a solution to the corresponding instance of the SSP.

We then continue as in the proposition, merging our system of equations into just one, by performing the same steps beginning with $x_1 + y_1 = 1$ and ending with $x_n + y_n = 1$. Note, that as we perform each step, we are not changing the solution set. After we have performed the first two steps we have the equation:

$$x_2 + y_2 + 4x_1 + 4y_1 + 4^2 \sum_{i=1}^{n} k_i x_i = 4(4M + 1) + 1$$

and then after n steps, we have obtained:

$$\sum_{i=1}^{n} 4^{n-1} x_i + \sum_{i=1}^{n} 4^{n-1} y_i + 4^n \sum_{i=1}^{n} k_i x_i = 4^n M + 4^{n-1} + 4^{n-2} + \cdots + 1$$

After collecting like terms on the left and summing the geometric series on the right we have (1.6). □

Theorem 1.9.4. $SSP \leq_p SSP' \leq_p TSSP$ *implying that the*

$TSSP$ is NP-complete as is the uniform conjugacy decision problem over the G_n.

Proof. All that is left to show is that the process described in this section turns instances of SSP, (1.4) into instances of $TSSP$, (1.6) in polynomial time and that the size of the $TSSP$ instance is only polynomially larger than the SSP instance. First notice that (1.6) has $2n$ indeterminates and since each of the 4^i can be expressed in $2n$ bits, the number of digits needed to express each coefficient increases polynomially. As such, (1.6) is polynomially larger than (1.4). Additionally, the new coefficients can clearly be computed in polynomial time. $\square$

Furthermore, from the argument in **Section 1.5**, a polynomial time algorithm for the conjugacy search problem over the G_n would imply P = NP.

1.10 The Conjugacy Problem In $\mathbb{Z}^n \rtimes_\phi \mathbb{Z}$

In this section we present an algorithm that solves both the conjugacy decision and search problems in groups of the form $\mathbb{Z}^n \rtimes_\phi \mathbb{Z}$. The algorithm we obtain is exponential time with regards to the length we define in **Section 1.1.3** and polynomial time if we use the actual word length of the normal form as in [27] rather than the length of the exponent vector. This work is found in [14] and is largely based on work due to Bogopolski, Maslakova, Martino, and Ventura from [7] and Bogopolski, Martino, and Ventura from [8, 9]. Additionally, we rely on an algorithm due to Kannan and Lipton from [39].

For reference, for $g = g_1^{k_1} \cdots g_n^{k_n}$, we have:

$$|g| = |k_1| + \cdots + |k_n|$$

1.10.1 The Twisted Conjugacy Problem

Definition 1.10.1. *Given a finitely presented group, G, an autormorphism $\phi \in \mathrm{Aut}(G)$, and $u, v \in G$ we say u and v are* twisted conjugate *by ϕ if there exists $x \in G$ such that*

$$v = xu\phi(x^{-1}).$$

If u and v are twisted conjugate by ϕ we write:

$$u \sim_\phi v.$$

Notice that the standard conjugacy problem is a special case of the twisted conjugacy problem by taking ϕ to be the identity.

In [7] Bogopolski, Martino, Maslakova, and Ventura showed that the conjugacy problem in free-by-infinite cyclic groups can be reduced to the twisted conjugacy problem in free groups. Later, Bogopolski, Martino, and Ventura [9] adapted their previous work from [7] to solve the conjugacy problem in a variety of groups extensions. What follows is an adaptation of their algorithm for free abelian-by-cyclic groups.

1.10.2 The Algorithm

The following lemma and proof is taken directly from the beginning of section 2 in [7] and adapted to free abelian-by-infinite cyclic groups.

Lemma 1.10.2. *Let $u = wt^s$ and $v = xt^r$ in $\mathbb{Z}^n \rtimes_\phi \mathbb{Z}$ be conjugate. Then $s = r$ and there exists $e \in \mathbb{Z}$ such that $\phi^e(w) \sim_{\phi^s} x$ in $\mathbb{Z}^n$. Additionally, if $\phi^s = \phi^r$ is the identity, then $x = \phi^e(w)$ for some $e \in \mathbb{Z}$.*

Proof. Let $a = bt^e \in \mathbb{Z}^n \rtimes_\phi \mathbb{Z}$ such that $v = aua^{-1}$. Therefore

$$xt^r = (bt^e)wt^s(bt^e)^{-1} = bt^ewt^st^{-e}b^{-1} =$$

$$b\phi^e(w)t^sb^{-1} = b\phi^e(w)\phi^s(b^{-1})t^s.$$

As such, we have:

$$xt^r = b\phi^e(w)\phi^s(b^{-1})t^s$$

which implies that $s = r$, and that $\phi^e(w) \sim_{\phi^s} x$ by b. $\square$

Given u and v as above, the lemma shows that there are two cases one must consider to solve the conjugacy decision and search problems in $\mathbb{Z}^n$-by-$\mathbb{Z}$ groups. First check if $s = r$. If not, then u and v are not conjugate. If they are equal, we then have the following two cases:

- If ϕ^s is trivial, we have to decide if $\exists e \in \mathbb{Z}$ such that $x = \phi^e(w)$.
- Otherwise, we have to decide if there exists e such that $\phi^e(w) \sim_{\phi^s} x$.

The first case, namely given two vectors $w, x \in \mathbb{Z}^n$ and $\phi \in \mathrm{GL}_n(\mathbb{Z})$ determine if there exists $e \in \mathbb{Z}$ such that $x = \phi^e(w)$, is known as the orbit problem over $\mathbb{Z}^n$. In [39] Kannan and Lipton provide a polynomial time algorithm that solves the orbit problem over $\mathbb{Q}^n$. The orbit problem over $\mathbb{Z}^n$ is a special case of their work, and as such, their algorithm provides a polynomial time solution to the twisted conjugacy problem over $\mathbb{Z}^n$ in the case that ϕ^s is trivial. If such an e is found that satisfies the orbit problem, then we have that $v = t^e u t^{-e}$.

For the second case, we use the fact from the lemma that $\exists b \in \mathbb{Z}^n$, $e \in \mathbb{Z}$ such that $x = b\phi^e(w)\phi^s(b^{-1})$. Before we begin the algorithm, we state **Lemma 1.7** from [7].

Lemma 1.10.3. *For any group G, $\phi \in \mathrm{Aut}(G)$, and $u \in G$, $u \sim_\phi \phi(u)$.*

Proof. $\phi(u) = u^{-1}u\phi(u)$. Therefore u is twisted conjugate over ϕ to $\phi(u)$ by u^{-1}. □

As such, $\phi^e(w) \sim_{\phi^s} \phi^{e \pm ks}(w)$ for any $k \in \mathbb{Z}$. Therefore, if there exists an e that satisfies the equation $\phi^e(w) \sim_{\phi^s} x$, then we can find such an e among $\{0, 1, \cdots, |s| - 1\}$.

We can now proceed with the full algorithm. Due to the fact that $x, w \in \mathbb{Z}^n$ and $\phi \in \mathrm{GL}_n(\mathbb{Z})$ it is more convenient to put the equation $x = b\phi^e(w)\phi^s(b^{-1})$ into additive notation. We then write:

$$x = b + \phi^e(w) - \phi^s(b).$$

This yields the equation

$$x - \phi^e(w) = (\mathrm{Id}_n - \phi^s)b$$

where Id_n is the $n \times n$ identity matrix. As such, we proceed by solving the system of linear equations given by $0 \leq e \leq |s| - 1$ and then checking if the solution, b is in $\mathbb{Z}^n$.

For a pseudocode version of the algorithm on inputs $wt^s, xt^r \in \mathbb{Z}^n \rtimes_\phi \mathbb{Z}$, see **Algorithm 1** (originally appearing in [14]). The algorithm returns **FALSE** if the elements are not conjugate, and a conjugating element if they are.

Algorithm 1 Conjugacy Algorithm for $\mathbb{Z}^n \rtimes_\phi \mathbb{Z}$

```
if $s \neq r$ then
    return FALSE
else if $\phi^s$ is the identity then
    Run Kannan-Lipton algorithm.
    if Kannan-Lipton returns $k$ then
        return $t^k$
    else return FALSE
    end if
else
    $e := 0$
    while $e < |s|$ do
        if $\exists b \in \mathbb{Z}^n$ such that $x - \phi^e(w) = (\mathrm{Id}_n - \phi^s)b$ then
            return $bt^e$
        else $e := e + 1$
        end if
    end while
    return FALSE
end if
```

1.10.3 Complexity Analysis

In the algorithm above we have two cases each of which can be dealt with in polynomially many steps with respect to n and $|s|$. If $s = r \neq 0$, we find solutions of an $n \times n$ linear system at most $|s|$ times. On the other hand, if $|s| = |r| = 0$, we use the algorithm of Kannan and Lipton which runs in polynomial time. Therefore, this algorithm is at most polynomial in terms of n and the lengths of the input words.

It is worth pointing out that unlike many of the algorithms group theorists study, this algorithm takes as inputs words in their polycyclic normal forms as opposed to in their geodesic form or just in any general form. This affects the complexity of the algorithm since the two different representations of group elements have different lengths.

One reason this is not a major issue is that in a practical setting, converting words to normal forms is fast (see [21]) and that the complexity is largely determined by the exponent above the stable letter (in the notation above, the stable letter is the generator t) which is just the sums of all exponents above the stable letter before collection. It is still open as to whether or not we can improve this case of the algorithm to be polynomial time when we define the length of a conjugacy problem as in **Section 1.3**.

Chapter 2

Properties of Infinite Cyclic Extensions

2.1 Introduction

In this chapter we study infinite cyclic extensions of groups that leave the base group stable under any automorphism. We begin with structural results of infinite cyclic extensions and then proceed by studying their automorphism groups. Inspired by [8] we define an automorphism to be *deranged* if its abelianization has no fixed points that are not torsion elements. For groups obtained through a cyclic extension by a deranged automorphism, we derive an explicit form for any of their automorphisms. We also prove structural results about their automorphism groups and show that an extension by a deranged automorphism is unique up to conjugacy class in the outer automorphism group of the base group.

We proceed by defining the class of H-poly-$\mathbb{Z}$ groups. G is H-poly-$\mathbb{Z}$ if G is obtained by successive infinite cyclic extensions from H. We show then that if the infinite cyclic extensions are done with "deranged" automorphisms, then any subnormal series for G with infinite cyclic quotients essentially begins at H and that G was obtained from H in a unique fashion. We call such a group G, H-rigid.

We finish by proving a version of the Tits Alternative for the automorphism group of an H-rigid poly-$\mathbb{Z}$ group and show that either it contains F_2 or is virtually polycyclic. More generally we give criteria for the automorphism group of a deranged extension to contain a non-abelian free group.

This is joint work with Jordi Delgado, Delaram Kahrobaei, Ha Lam, and Enric Ventura and can be found in [12].

2.2 Structure of Infinite Cyclic Extensions and Deranged Automorphisms

In **Section 1.1.3** we discussed the theory of infinite cyclic extensions in the context of poly-$\mathbb{Z}$ groups. In this section we present more general results pertaining to extending an arbitrary group by $\mathbb{Z}$. In what follows we denote $G_\phi = G \rtimes_\phi \mathbb{Z}$. We also refer to the notation of the last chapter and denote t as the new generator of the infinite cyclic extension.

Lemma 2.2.1. *Let ϕ be an automorphism of an arbitrary group G. Then the abelianization of G_ϕ is:*

$$G_\phi^{ab} \simeq \frac{G^{ab}}{\operatorname{Im}(\phi^{ab} - \operatorname{Id})} \oplus \mathbb{Z}$$

where $\phi^{ab} \in \operatorname{Aut}(G^{ab})$ denote the abelianization of ϕ and Id is the identity map on G^{ab}.

Proof. Let $G = \langle X | R \rangle$. Then from **Section 1.1.3** we have that

$$G_\phi = \langle X, t | R, tx_i t^{-1} = \phi(x_i) : x_i \in X \rangle.$$

When we abelianize, we add in the relations $x_i t = t x_i$ and $x_i x_j = x_j x_i$, which in turn implies that $x_i = \phi(x_i)$. Hence:

$$G_\phi^{ab} = \langle X, t | R, x_i = \phi(x_i), x_i t = t x_i, x_i x_j = x_j x_i : x_i, x_j \in X \rangle$$

$$= \langle X | R, x_i = \phi(x_i), x_i x_j = x_j x_i : x_i, x_j \in X \rangle \oplus \langle t \rangle.$$
$$= \frac{G^{ab}}{\mathrm{Im}(\phi^{ab} - \mathrm{Id})} \oplus \mathbb{Z}.$$

□

If G is finitely generated, then so is G_ϕ implying that G_ϕ^{ab} is a finitely generated abelian group. Therefore, by the fundamental theorem of finitely generated abelian groups we have:

Corollary 2.2.2. *If G is finitely generated then $G_\phi^{ab} \simeq \mathbb{Z}^r \oplus T$ for some $r \in \mathbb{Z}$ and T a finite abelian group.*

For any automorphism $\phi \in \mathrm{Aut}(G)$ for G finitely generated, we have the associated abelianization

$$\phi^{ab} : \mathbb{Z}^r \oplus T \to \mathbb{Z}^r \oplus T$$

which is also an automorphism. Therefore, it sends $\mathbb{Z}^r$ onto $\mathbb{Z}^r$. We can then quotient out G^{ab} by the torsion to obtain a well defined automorphism

$$\phi_*^{ab} : \mathbb{Z}^r \to \mathbb{Z}^r$$

which can be thought of as a matrix in $GL_r(\mathbb{Z})$.

Definition 2.2.3. *We say an automorphism $\phi \in \mathrm{Aut}(G)$ is* deranged *if ϕ_*^{ab} has no non-trivial fixed points.*

This definition is inspired by the type of automorphisms studied in **Theorem 2.4** from [8] where the authors studied free-by-cyclic groups and showed that with a similar definition on automorphisms, G is characteristic in G_ϕ when ϕ is deranged. Namely, $\psi(G) \subset G$ for all $\psi \in \mathrm{Aut}(G_\phi)$. For this chapter we show similar results where G is an arbitrary group and use them to study $\mathrm{Aut}(G_\phi)$.

Theorem 2.2.4. *Let G be a finitely generated group, $\phi \in \mathrm{Aut}(G)$. Then the following are equivalent:*

(a) ϕ is deranged.

(b) $G_\phi^{ab} \simeq \mathbb{Z} \oplus T$ for some finite abelian group T.

(c) G is the only normal subgroup of G_ϕ with quotient isomorphic to $\mathbb{Z}$.

(d) For all endomorphisms $\psi \in End(G_\phi)$, $\psi(G) \subset G$.

Proof. Let $G^{ab} \simeq \mathbb{Z}^r \oplus T$ where T is the torsion part of the group. Then:

$$G_\phi^{ab} \simeq \mathbb{Z} \oplus \frac{G^{ab}}{\mathrm{Im}(\phi^{ab} - \mathrm{Id})} \simeq \mathbb{Z} \oplus \frac{\mathbb{Z}^r \oplus T}{\mathrm{Im}(\phi^{ab} - \mathrm{Id})}.$$

$\mathrm{Im}(\phi^{ab} - \mathrm{Id})$ is a subgroup of $\mathbb{Z}^r \oplus T$ and has its own torsion-free part that is a subgroup of $\mathbb{Z}^r$ and its own torsion part that is a subgroup of T. The torsion-free part is actually equal to $Im(\phi_*^{ab} - Id)$. We then have:

$$\mathbb{Z} \oplus \frac{\mathbb{Z}^r \oplus T}{\mathrm{Im}(\phi^{ab} - \mathrm{Id})} \simeq \mathbb{Z} \oplus \frac{\mathbb{Z}^r}{\mathrm{Im}(\phi_*^{ab} - \mathrm{Id})} \oplus T'$$

where T' is the quotient of T by the torsion part of $\mathrm{Im}(\phi^{ab} - \mathrm{Id})$. Now, $\mathbb{Z}^r/\mathrm{Im}(\phi_*^{ab} - \mathrm{Id})$ is finite if and only if $det(\phi_*^{ab} - \mathrm{Id}) \neq 0$ which happens if and only if ϕ_*^{ab} has no non-trivial fixed points. This proves equivalence of (a) and (b).

To see (b) if and only if (c), first note that every epimorphism from G_ϕ onto $\mathbb{Z}$ factors through G_ϕ^{ab}. Also note that if (b) holds, then G is exactly the kernel of the map that sends $G \subset G_\phi$ to G_ϕ^{ab} and then to $\mathbb{Z}$. This implies that G is exactly the kernel of any map from G to $\mathbb{Z}$ since G gets sent to the torsion part of G_ϕ^{ab}. Therefore, if there existed a second subgroup H such that $G_\phi/H \simeq \mathbb{Z}$, then G would be the kernel of the quotient map implying that $G = H$. On the other hand, if G_ϕ^{ab} had higher rank than 1, there would clearly be multiple groups with quotient $\mathbb{Z}$.

Finally we do (b) if and only if (d). If (b) holds, then $G_\phi^{ab} \simeq \mathbb{Z} \oplus T$ where G is the pre-image of T under the abelianization quotient map. Therefore T is invariant under any endomorphism of G_ϕ^{ab} and

G is invariant under any endomorphism of G_ϕ. Now we prove (d) implies (b) by using the contrapositive. Say that $G_\phi^{ab} \simeq \mathbb{Z}^2 \oplus T$, and let s be the generator of the additional $\mathbb{Z}$. Therefore, there exists an element of G that maps to a power of s under the abelianization quotient map. Then there is an endomorphism that sends G to the cyclic subgroup generated by the pre-image of s implying G is not fully invariant. Note that this proof still holds for when the abelianization has higher rank. □

2.3 Automorphisms of Deranged Extensions

We proved in the previous section that when ϕ is deranged, G is characteristic in G_ϕ. This allows us to prove the following proposition about $\mathrm{Aut}(G_\phi)$:

Proposition 2.3.1. *Let $\Psi \in \mathrm{Aut}(G_\phi)$ with ϕ deranged. Then Ψ is of the following form:*

$$\Psi(g) = \psi(g)$$

$$\Psi(t) = ht^\epsilon$$

where $\epsilon = \pm 1$, $\psi \in Aut(G)$, and $h \in G$ are such that $\gamma_h \psi^\epsilon \psi = \psi\phi$ where $\gamma_h(g) = hgh^{-1}$. Additionally, any homomorphism satisfying these requirements is in $\mathrm{Aut}(G_\phi)$

Proof. First we prove $\Psi|_G = \psi$ is an automorphism of G. All that remains is to show that ψ is surjective. Since $G_\phi^{ab} \simeq \mathbb{Z} \oplus T$, $x \in G_\phi$ gets sent to T under the abelianization quotient map if and only if $x \in G$. Therefore, x^{ab} gets sent to T by Ψ^{ab} which means that $\Psi(x) \in G$ and $\Psi(G) \subset G$. Moreover, it also implies that Ψ also sends everything outside of G outside of G. Since Ψ is onto, $\Psi(G) = \psi(G) = G$.

Since Ψ sends G to G, for $\Psi(G_\phi)$ to contain t, t must get sent to a word of the form ht^ϵ for $\epsilon = \pm 1$.

Now, for this map to be well defined, it must satisfy the following:

$$\Psi(tgt^{-1}) = \Psi\phi(g)$$

$$ht^\epsilon\psi(g)t^{-\epsilon}h^{-1} = \psi\phi(g)$$

$$\gamma_h\phi^\epsilon\psi(g) = \psi\phi(g).$$

Finally, note that any map of the above form is necessarily an automorphism. □

It is also the case that the above proof will actually work whenever G is characteristic in G_ϕ.

Following the form of automorphisms in the above proposition, we will denote $\Psi_{\epsilon,\psi,h}$ as the automorphism that sends $g \in G$ to $\psi(g)$ and t to ht^{ϵ}. Composition in the group can then be seen as:

$$\Psi_{\epsilon_2,\psi_2,h_2}\Psi_{\epsilon_1,\psi_1,h_1} = \Psi_{\epsilon_1\epsilon_2,\psi_2\psi_1,\psi_2(h_1)\phi^{(\epsilon_1-1)\frac{\epsilon_2}{2}}h_2^{\epsilon_1}}$$

This form of automorphisms allows us to characterize automorphisms in $\mathrm{Aut}(G_\phi)$ by whether ϵ is 1 or -1. We call ϵ the *signum* and have $\mathrm{Aut}^+(G_\phi)$ and $\mathrm{Aut}^-(G_\phi)$ denote the set of automorphisms with positive and negative signum respectively. Therefore:

$$\mathrm{Aut}(G_\phi) = \mathrm{Aut}^+(G_\phi) \sqcup \mathrm{Aut}^-(G_\phi).$$

$\mathrm{Aut}^+(G_\phi)$ is never empty. For instance, it contains the identity, inner automorphisms, and $\Psi_{1,\phi,1}$. On the other hand, there may be no automorphisms with signum -1.

Notice that the composition of two automorphisms with positive signum is again positive. Also, the inverse of a positive automorphism is positive. This implies that $\mathrm{Aut}^+(G_\phi)$ is a normal subgroup of $\mathrm{Aut}(G_\phi)$. Due to the partition above, $\mathrm{Aut}^+(G_\phi)$ is of index at most two in $\mathrm{Aut}(G_\phi)$ depending on whether or not there exist automorphisms with signum -1.

Lemma 2.3.2. *The map*

$$\mathrm{Aut}^+(G_\phi) \to \mathrm{Aut}(G)$$

$$\Psi \mapsto \psi = \Psi|_G$$

is a well-defined homomorphism with the center of G *as its kernel and image* $\tilde{C}([\phi])$ *where*

$$\tilde{C}([\phi]) = \{\psi \in \mathrm{Aut}(G) | \psi\phi\psi^{-1}\phi^{-1} \in \mathrm{Inn}(G)\}$$

Therefore, $\mathrm{Aut}^+(G_\phi)$ *fits into the short exact sequence:*

$$1 \to Z(G) \to \mathrm{Aut}^+(G_\phi) \to \tilde{C}([\phi]) \to 1$$

Proof. We first look at the second restriction map which we call π. One can see that π is a homomorphism by the composition laws from **Proposition 2.3.1**. If $\Psi_{1,\psi,h}$ gets sent to the identity, then $\psi = Id$ and $\gamma_h\phi = \phi$ which implies that $h \in Z(G)$. As such, the map from $Z(G)$ into $\mathrm{Aut}^+(G_\phi)$ given by $h \mapsto \gamma_h$ is injective. Moreover, $\Psi_{1,\psi,h} \in \mathrm{Aut}^+(G_\phi)$ implies that ψ commutes with ϕ up to an inner automorphism. □

To finish the section, we show that any two extensions by deranged automorphisms are isomorphic if and only if the two automorphisms are in the same conjugacy class in $\mathrm{Out}(G)$. From the proof, it can be seen that the backward direction holds for an arbitrary group whereas the forward direction will not. See [8] for a counterexample in which G is the free group on three generators. This result was also proven independently in [11] and [3].

Proposition 2.3.3. *Let $G_\alpha = G \rtimes_\alpha \mathbb{Z}$ and $G_\beta = G \rtimes_\beta \mathbb{Z}$. If α is deranged, then $G_\alpha \simeq G_\beta$ if and only if $\exists w \in G$, $\epsilon = \pm 1$, and $\psi \in \mathrm{Aut}(G)$ such that $\beta = \gamma_w \psi \alpha^\epsilon \psi^{-1}$.*

Proof. The proof of the "if" part can be found in [8]. To be brief, an explicit isomorphism between G_α and $G_{\gamma_w \psi \alpha^\epsilon \psi^{-1}}$ is $g \mapsto \psi(g)$ for $g \in G$ and $t \mapsto ws^\epsilon$ where t generates $\mathbb{Z}$ in G_α and s generates $\mathbb{Z}$ in $G_{\gamma_w \psi \alpha^\epsilon \psi^{-1}}$.

In the other direction, let $\psi : G_\alpha \to G_\beta$ be an isomorphism. We first show that $\psi|_G : G \to G$ is an isomorphism. We have that $f|_G$ is linear and injective, so we only need to prove surjectivity. Since α deranged, $G_\alpha^{ab} \simeq \mathbb{Z} \oplus T \simeq G_\beta^{ab}$. Therefore, if we let $g \in G$, then $(f(g))^{ab} = f^{ab}(g^{ab})$ implies that $(f(g))^{ab} \in T$ which means that $f(g) \in G$. Additionally, any word containing the letter t does not abelianize into torsion which means that it does not get sent into G by f. But since f is surjective onto G_β, $f(G) = G$. This in turn means that ψ must send t to an element of the form $hs^{\pm 1}$ where s generates the $\mathbb{Z}$ in G_β.

Applying ψ to the relation $tgt^{-1} = \alpha(g)$ gives us

$$\psi(t)\psi(g)\psi(t)^{-1} = \psi\alpha(g)$$

$$hs^{\epsilon}\psi(g)s^{-\epsilon}h^{-1} = \psi\alpha(g)$$

$$\gamma_h\beta^{\epsilon}\psi(g) = \psi\alpha(g)$$

$$\beta^{\epsilon}(g) = \gamma_{h^{-1}}\psi\alpha\psi^{-1}(g)$$

If $\epsilon = 1$ then we can take $w = h^{-1}$. If $\epsilon = -1$, then we get:

$$\beta(g) = \gamma_{\psi\alpha\psi^{-1}(h)}\psi\alpha^{-1}\psi^{-1}(g)$$

and can take $w = \psi\alpha\psi^{-1}(h)$ $\square$

Corollary 2.3.4. $\mathrm{Aut}^{-}(G_\phi)$ *is non-empty if and only if* $G_\phi \simeq G_{\phi^{-1}}$.

Proof. This follows from **Proposition 2.3.1** in that $\mathrm{Aut}^{-}(G_\phi)$ has an automorphism, ψ, with signum -1, if and only if $\gamma_h\phi^{-1}\psi = \psi\phi$. This is the case if and only if ϕ and ϕ^{-1} are conjugate in $\mathrm{Out}(G)$. $\square$

2.4 Rigid Poly-$\mathbb{Z}$ Groups

Definition 2.4.1. *We say a group G is* poly-$\mathbb{Z}$ relative to a subgroup H *if G admits a finite subnormal series*

$$H = G_0 \lhd G_1 \lhd \cdots \lhd G_n = G$$

such that $G_i/G_{i-1} \simeq \mathbb{Z}$ for $i = 1, \cdots, n$. We also write that G is H-poly-$\mathbb{Z}$. Such a subnormal series can be called an H-poly-$\mathbb{Z}$ series.

In this way, we can view any poly-$\mathbb{Z}$ group as $\{1\}$-poly-$\mathbb{Z}$ or H-poly-$\mathbb{Z}$ where H is itself poly-$\mathbb{Z}$.

Definition 2.4.2. *We say that a G_0-poly-$\mathbb{Z}$ series*

$$G_0 \lhd G_1 \lhd \cdots \lhd G_n = G$$

extends through a subgroup H *if one of the following conditions hold:*

(i) the subgroup H already belongs to the series ($H = G_i$ for some i).

(ii) there exists a poly-$\mathbb{Z}$ series from H to G_0.

Note that in either case, there is a poly-$\mathbb{Z}$ series from H to G that overlaps a final segment of the original series. In particular, G is H-poly-$\mathbb{Z}$.

Definition 2.4.3. *A group G is* rigidly poly-$\mathbb{Z}$ relative to a subgroup H *(or H-rigid), if every poly-$\mathbb{Z}$ series for G extends uniquely through H. Namely, every poly-$\mathbb{Z}$ series extends through H, and the extended series from H to G is also unique.*

Note that this means the extending automorphisms can be different, but the subgroups the same. In fact, considering **Proposition 2.3.3**, there will in general be many automorphisms that give you the same poly-$\mathbb{Z}$ series.

Proposition 2.4.4. *Let G by an H-poly-$\mathbb{Z}$ group. Then the following are equivalent:*

(1) G is H-rigid.

(2) Every poly-$\mathbb{Z}$ tower from H to G is made by extensions by deranged automorphisms.

(3) There exists a deranged poly-$\mathbb{Z}$ tower from H to G.

Proof. [(1) $\implies$ (2)] Let G be H-rigid and suppose that there exists a poly-$\mathbb{Z}$ tower from H to G with some extending automorphism, ϕ, not deranged. Say $G_{i+1} = G_i \rtimes_\phi \mathbb{Z}$. Then, G_{i+1} has a subgroup, $H_i \neq G_i$, with quotient isomorphic to $\mathbb{Z}$. Therefore, G is not H-rigid because either H is not a subgroup of H_i or there are two different poly-$\mathbb{Z}$ series extending through H.

[(2) $\implies$ (3)] is trivial.

[(3) $\implies$ (1)] Let $H = G_0 \lhd G_1 \lhd \cdots \lhd G_n = G$. Then for every $i \geq 1$, there is a unique subgroup of G_i with quotient isomorphic to $\mathbb{Z}$. Therefore, the unique subgroup of G_i is G_{i-1} and this continues all the way down to H being the unique subgroup of G_1 such that $G_1/H \simeq \mathbb{Z}$. This implies that any poly-$\mathbb{Z}$ tower for G can be uniquely extended down to H. $\square$

2.5 A Tits Alternative For Automorphisms of a Rigid Poly-$\mathbb{Z}$ Group

The original Tits Alternative [63] states that any linear group is either virtually solvable or contains a non-abelian free group. Both virtually polycyclic groups and their automorphism groups are linear ([59] chapter 5) . Moreover, it as a theorem of Mal'cev that if the automorphism group a virtually polycyclic group is solvable, then it is in fact polycyclic (see [59] Chapter 2 section B theorem 1). As such, a Tits Alternative for automorphism groups of virtually polycyclic groups states that such automorphism groups are either polycyclic or contain a non-abelian free group. See [20] for a more detailed study of the Tits Alternative for a virtually polycyclic group.

Proposition 2.5.1. *Let G be an H-rigid relatively poly-$\mathbb{Z}$ group. Also, let F_2 be the non-abelian free group on two generators. If $F_2 \not\leq \widetilde{C}([\phi])$ where the first group in the tower is $G_1 = H \rtimes_\phi \mathbb{Z}$, then $F_2 \not\leq \mathrm{Aut}(G)$*

To prove this me make use of the following lemma:

Lemma 2.5.2. *Let G contain subgroups $A = \langle a, b\rangle \simeq F_2$ and K such that $F_2 \cap K = \langle w\rangle$. Then G contains a subgroup isomorphic to F_2, L, such that $L \cap K = \{1\}$.*

Proof. Take a word $k \in A$ such that $k \notin \langle w\rangle$. Then $\langle k, wkw^{-1}\rangle$ is isomorphic to F_2 and does not contain w. □

Now we can prove the proposition:

Proof. To prove this we examine the short exact sequence from **Lemma 2.3.2** and show that if $F_2 \not\leq \widetilde{C}(\phi)$ then $F_2 \not\leq \mathrm{Aut}^+(G_\phi)$. If this holds then $F_2 \not\leq \mathrm{Aut}(G_\phi)$ because $\mathrm{Aut}^+(G_\phi)$ is a finite index subgroup.

We proceed by contradiction. Assume that $A \leq \mathrm{Aut}^+(G_\phi)$ where $A \simeq F_2$, but that $\widetilde{C}([\phi])$ contains no subgroup isomorphic to F_2. Note that since $Z(G)$ is abelian, $A \cap Z(G)$ is either trivial or is an infinite cyclic group $\langle w \rangle$. By applying the previous lemma, we can then assume without loss of generality that $A \cap Z(G) = \{1\}$. Therefore A embeds into $\mathrm{Aut}^+(G_\phi)/Z(G) \simeq \widetilde{C}([\phi])$ which contradicts that $\widetilde{C}([\phi])$ has no subgroups isomorphic to F_2. $\square$

2.6 Recognizing Finitely Presented-by-$\mathbb{Z}$ Groups

In this section we describe an algorithm that takes as input a presentation of a finitely presented-by-$\mathbb{Z}$ group, and recursively enumerates its presentations. Following that, we show that determining that an arbitrary finitely presented group is finitely presented-by-$\mathbb{Z}$ is undecidable. For the remainder of this section we denote finitely presented (generated)-by$\mathbb{Z}$ as f.p(g)-by-$\mathbb{Z}$. Before we begin, let us define what we call the *canonical presentation* for G, an f.p-by-$\mathbb{Z}$ group. A canonical presentation will be one of the form:

$$G = \langle g_1, g_2, \cdots, g_n, t \mid R, tg_1t^{-1} = \phi(g_1), \cdots, tg_nt^{-1} = \phi(g_n) \rangle.$$

Namely, we can represent G as a set of generators $g_1, \cdots, g_n$ and a stable letter t, where relations in the set R do not contain t and the remaining relations can be expressed as conjugation of a generator by t yielding an automorphism of $\langle g_1, \cdots, g_n \mid R\rangle$. From **Section 1.3.2** any f.p-by-$\mathbb{Z}$ group has such a presentation and any group with such a presentation is f.p-by-$\mathbb{Z}$.

Proposition 2.6.1. *There is an algorithm that takes as input an arbitrary presentation of an f.p-by-$\mathbb{Z}$ group, G, and returns a canonical presentation. As such, the set of finite presentations of f.p-by-$\mathbb{Z}$ groups is recursively enumerable.*

Proof. We begin the proof by first describing how to verify that the automorphism ϕ in a canonical presentation is an automorphism of the base group. This process is guaranteed to terminate in a finite number of steps if ϕ is an automorphism (implying that our presentation is canonical) but may never terminate if ϕ is not. We will describe two potentially infinite processes that altogether verify that ϕ is an automorphism. Each will terminate if ϕ is an automorphism.

We first verify that ϕ is well defined. This can be done by checking that the images of the relators in R under ϕ are trivial. Since the base group may have undecidable word problem, we cannot do that by merely checking that each image is trivial. Instead we begin the potentially infinite process of verifying that each word is trivial by enumerating all words in the normal closure of R in the base group and checking if the finite images of the words in R eventually appear. If not all of the images are trivial, this process is not guaranteed to terminate, but if ϕ is a well-defined automorphism we can verify that the images are trivial in a finite number of steps.

To check that ϕ is bijective, we attempt to construct an inverse. This process again cannot be done deterministically, but will produce an inverse if an inverse exists. We begin by enumerating all possible maps, $\psi : g_i \mapsto w_i$ of the base group. We can verify well definedness by using the process of the previous paragraph. If that process terminates, we then verify that $\psi\phi(g_i) = \phi\psi(g_i) = g_i$ which can be done again by enumerating elements in the normal closure of R in the base group and looking for $\psi\phi(g_i)g_i^{-1}$ and $\phi\psi(g_i)g_i^{-1}$. If we find a well defined inverse ψ, we then know that ϕ is a bijection.

In order to find a canonical presentation for our input group, G, we begin by looking at G under all possible Tietze transformations. Since G is f.p-by-$\mathbb{Z}$, we know that a canonical presentation exists, and as such, we will eventually find some combination of Tietze transformations that yield a canonical presentation. As we enumerate, we look for candidate presentations, namely presentations that have the same form as a canonical presentation, but where ϕ may not be an automorphism of the base group. Each time we find a candidate presentation, we begin the parallel process of determining if ϕ is an automorphism. If ϕ is not an automorphism, this process may never terminate. As such, we continue searching for more candidate presentations while simultaneously checking that ϕ is an automorphism. Each time we find a new candidate presentation, we begin a new, potentially infinite, parallel sub-process. This finite number of potential infinite processes can be done simultaneously by alternating steps for each of them. When we finally verify that a candidate presentation is in fact a canonical presentation for G, we then terminate all ongoing processes.

Since we will eventually find a canonical presentation by Tietze transformations and since we can verify that the ϕ for this presentation is an automorphism in a finite number of steps, this algorithm will eventually terminate. $\square$

We will now show that determining if an arbitrary f.p group is f.p-by-$\mathbb{Z}$ (or even f.g-by-$\mathbb{Z}$) is undecidable. To do this, we begin with a lemma and a corollary:

Lemma 2.6.2. *Let H be a finitely generated group and write $*_{i\in\mathbb{Z}}^{\infty}H$ as*

$$\langle \cdots, H_{-2}, H_{-1}, H_0, H_1, \cdots \mid \cdots, R_{-2}, R_{-1}, R_0, R_1, \cdots \rangle$$

where the H_i and R_i are distinct disjoint copies of H and the relators of H respectively and are indexed by $i \in \mathbb{Z}$. We then have that:

$$*_{i\in\mathbb{Z}}^{\infty}H \rtimes_{\phi} \mathbb{Z} \simeq H * \mathbb{Z}$$

where ϕ sends $h_i \in H_i$ to its copy $h_{i+1} \in H_{i+1}$.

Proof. We have that:

$$*_{i\in\mathbb{Z}}^{\infty}H \rtimes_{\phi} \mathbb{Z} = \langle H_i, t \mid R_i, th_it^{-1} = h_{i+1} \rangle$$

for all $h_i \in H_i$ and all $i \in \mathbb{Z}$.

Therefore:

$$h_i \mapsto t^i h t^{-i}$$

$$t \mapsto t$$

is an isomorphism between $*_{i\in\mathbb{Z}}^{\infty}H \rtimes_{\phi} \mathbb{Z}$ and $H * \mathbb{Z}$. $\square$

Corollary 2.6.3. *Let $G = H * \mathbb{Z}$ where H is f.g and has finite abelianization. Then G is f.g-by-$\mathbb{Z}$ if and only if H is trivial.*

Proof. First note that $G^{ab} \simeq H^{ab} \oplus \mathbb{Z} \simeq T \oplus \mathbb{Z}$ where T is a finite abelian group. Based on the proof of **Theorem 2.2.4** we have that G has a unique normal subgroup with quotient $\mathbb{Z}$. From the previous lemma, this unique subgroup is isomorphic to $*_{i \in \mathbb{Z}}^{\infty} H$ which is finitely generated if and only H is trivial. $\square$

We now use the fact that in the variety of f.g groups, triviality is not recursively recognizable from a finite presentation (see [48]). Note that this can be restricted to f.g perfect groups since one can verify from its finite presentation that any non-perfect f.g group is non-trivial by taking its abelianization and verifying that it is non trivial.

Theorem 2.6.4. *There is no algorithm to determine if a finitely presented group given by a finite presentation is f.p-by-$\mathbb{Z}$ or f.g-by-$\mathbb{Z}$.*

Note that being f.p-by-$\mathbb{Z}$ is not a Markov property since any finitely presented group, G, embeds into $G \times \mathbb{Z}$ which is also f.p-by-$\mathbb{Z}$.

Proof. Let P be a perfect group given by a finite presentation and form $G = P * \mathbb{Z}$. From **Corollary 2.6.3** we have that G is f.g-by-$\mathbb{Z}$

and f.p-by-$\mathbb{Z}$ if and only P is non-trivial. Therefore determining that G is f.g-by-$\mathbb{Z}$ or f.p-by-$\mathbb{Z}$ is equivalent to determining if P is trivial. Since determining if f.p perfect groups are trivial is undecidable, determining if G is f.g-by-$\mathbb{Z}$ is undecidable. □

2.7 Undecidability Properties of the BNS Invariant

In this section we point out how our results affect the computation of the Bieri-Neumann-Strebel Invariant from [5]. In what follows, we introduce the BNS invariant using the definition from [44].

For a finitely generated group G with $b_1(G) = n$, we define the *character sphere of* G:

$$S(G) = (\mathrm{Hom}(G, \mathbb{R}) - \{0\})/\mathbb{R}_+ \cong S^{n-1}.$$

Each $[\chi] \in S(G)$ yields the submonoid

$$G_\chi = \{g \in G \mid \chi(g) \geq 0\}$$

of G. Fix a finite generating set X of G. We now define

$$\Sigma = \{[\chi] \in S(G) \mid G_\chi \text{ is a connected subset of } \Gamma(G, X)\}.$$

It is the case that G_χ is connected regardless of the choice of finite generating set. Σ is known to be related to the set of infinite cyclic quotients with finitely generated kernel (see [52]). To see the relation between Σ and finitely generated subgroups we need to give one more definition. For any subgroup $H \leq G$ we define

$$S(G, H) = \{[\chi] \in S(G) \mid \chi(H) = 0\}.$$

Following this we state part of **Theorem B1** from [5].

Theorem 2.7.1. *Let N be a normal subgroup of G with G/N abelian. Then N is finitely generated if and only if $S(G, N) \subseteq \Sigma$.*

In the case where $G = H \rtimes_\phi \mathbb{Z}$ is obtained as a deranged cyclic extension, we have that $S(G) = S^0 = \{-1, 1\}$ since $b_1(G) = 1$. Fix an epimorphism $\chi : G \twoheadrightarrow \mathbb{Z}$. Since H is the unique normal subgroup of G with quotient $\mathbb{Z}$, $S(G, H)$ consists only of χ and $-\chi$. Since the kernel of each map is H, **Theorem 2.7.1** implies that either $\Sigma = \emptyset$, if H is not finitely generated, and $\Sigma = \{\chi, -\chi\} = S(G, H) = S(G)$ if H is finitely generated.

From the proof **Theorem 2.6.4**, we have that there is a family of finitely presented groups with rank one abelianization of the form $P * \mathbb{Z}$, where the problem of deciding if a group within the family is [f.g]-by-$\mathbb{Z}$ is undecidable. Additionally, from the previous paragraph, we have that this problem is equivalent to determining if the kernel of the epimorphism onto $\mathbb{Z}$ from such a group is finitely generated, is then equivalent to determining Σ for G. As such we have:

Theorem 2.7.2. *Given a finite presentation of an f.p-by-$\mathbb{Z}$ group, G, there is no algorithm to compute the Bieri-Neumann-Strebel invariant of G.*

Chapter 3

Secret Sharing Using Non-Commutative Groups

3.1 Introduction

In this chapter we change gears and talk about a secret sharing protocol that involves non-commutative groups. Secret Sharing essentially solves the problem of how to split a piece of information (a secret) amongst multiple parties such that no single party can recover the secret, but that certain subsets of participants can combine their information to recover the secret. Note that this problem is non-trivial because each subset needs to recover the same secret.

Secret sharing schemes are ideal for situations in which it is important that the secret is robust (can be recovered easily when information is lost) and where a high level of security is still needed. The scheme is secure if done correctly, and if shares are lost, the secret can still be recovered. Secret sharing has applications in multiparty encryption, Byzantine encryption, threshold encryption. See [4] for a detailed introduction to secret sharing and its applications.

In this chapter we first detail the classical secret sharing schemes of Blakley and Shamir before moving on to later adjustments by Feldman and Pedersen that account for cheating dealers or participants and Herzberg et. al. that account for long term storage of shares. Finally, we outline recent work in secret sharing using non-commutative groups by Habeeb, Kahrobaei, and Shpilrain [30] and an original adjustment of their scheme.

3.2 Formal Definition

A *secret sharing scheme* consists of a dealer, n participants, that we label $P_1, \cdots, P_n$, and an access structure $\mathcal{A} \subseteq 2^{\{P_1, \cdots, P_n\}}$ such that for all $A \in \mathcal{A}$ and $A \subseteq B$, $B \in \mathcal{A}$.

To share a secret, s, the dealer runs an algorithm:

$$Share(s) = (s_1, \cdots, s_n)$$

and then distributes each share s_i to P_i.

In order to recover the secret, participants can run the algorithm *Recover* which has the property that for all $A \in \mathcal{A}$:

$$Recover(\{s_i : i \in A\}) = s$$

and if $A \notin \mathcal{A}$ then running *Recover* is either computationally in-feasible or impossible.

As such, only sets of participants in $\mathcal{A}$ can perform *Recover* and access the secret. The monotonicity of $\mathcal{A}$ is also apparent in that if $A \in \mathcal{A}$ and $A \subseteq B$ then the set of participants in A could also recover the secret for B. As such, B should also be in $\mathcal{A}$.

Definition 3.2.1. *A secret sharing scheme is called* perfect *if* $\forall A \notin \mathcal{A}$ *the shares* $s_i \in A$ *together give no information about* s*.*

3.3 Classical Secret Sharing

A common access structures in secret sharing is the (k, n) threshold:

$$\mathcal{A} = \{A \in 2^{\{P_1, \cdots, P_n\}} : |A| \geq k\}.$$

Namely, $\mathcal{A}$ consists of all subsets of the n participants of size k or greater. We call such a secret sharing scheme a *(k,n) threshold scheme.* In what follows we first give a trivial example of an (n, n) scheme and then two schemes that solve the problem for arbitrary k.

Example: A trivial (n, n) scheme can be constructed as follows to share a secret $s \in \{0, 1\}^m$:

- The dealer randomly chooses $s_1, \cdots, s_{n-1} \in \{0, 1\}^m$ and sets

  $$s_n = s_1 \oplus \cdots \oplus s_{n-1} \oplus s$$

  where $\oplus$ is the standard XOR.

- The dealer distributes s_i to each P_i over a private channel.

To recover the secret, all the participants must combine their information and compute $s = s_1 \oplus \cdots \oplus s_n$.

The problem of discovering a non-trivial (k, n) scheme was solved independently by G. Blakley [6] and A. Shamir [60] in 1979. Notice that this problem becomes non-trivial in part because the shares have to be consistent. This means any k person subset has to recover the same secret.

3.3.1 Blakley's Scheme

In Blakley's (k, n) scheme, the secret is an element $s \in \mathbb{R}^k$. The dealer distributes to each P_i the equation for a hyperplane in $\mathbb{R}^k$ such that each hyperplane contains the point s and such that any k of the hyperplanes are linearly independent. To recover the secret any k participants can use the equations for their hyperplanes to solve the linear system and get a unique solution. The scheme is consistent because any k of the hyperplanes are linearly independent and each plane contains s.

Example: Consider a (3,5) threshold using Blakely's scheme. Let us have the secret be $(1, 2, 3)$. The shares could be:

$$s_1 = \{2x + y + z = 7\}$$

$$s_2 = \{3x - 6y + z = -9\}$$

$$s_3 = \{-x + 7y + 3z = 22\}$$

$$s_4 = \{4x - 2y - 2z = -6\}$$

$$s_5 = \{x + y + z = 6\}$$

One can verify that this works by checking that $(1, 2, 3)$ is a solution to each system and that the planes are linearly independent.

3.3.2 Shamir's Scheme

Shamir's secret sharing scheme also involves linear algebra at its core, but rather than involving hyperplanes explicitly, it utilizes the linear relationships between polynomials over finite fields. For Shamir's (k, n)-scheme, the secret s is an element of $\mathbb{Z}_p$ where we only require that $p > n$. To share s, the dealer does the following:

- The dealer randomly selects $a_1, \cdots, a_{k-1} \in \mathbb{Z}_p$ such that $a_{k-1} \neq 0$ and constructs the polynomial $f(x) = a_{k-1}x^{k-1} + \cdots + a_1 x + s$.
- For each participant P_i the dealer publishes distinct corresponding $x_i \in \mathbb{Z}_p$. The dealer then distributes the share $s_i = f(x_i)$ to each P_i over a private channel.

Since each share represents a distinct point on the same degree $k - 1$ polynomial, f, and since each share is unique, each subset of k participants can uniquely reconstruct the polynomial and find its constant coefficient. If we view P_i as $(x_i, f(x_i))$, the first k participants can reconstruct the secret by computing (by polynomial interpolation):

$$f(0) = \sum_{i=1}^{k} y_i \prod_{1 \leq j \leq k, j \neq i} \frac{-x_j}{x_i - x_j} = s.$$

Another way of viewing polynomial interpolation is as follows: in order to reconstruct a polynomial $f(x) = s + a_1 x + \cdots a_{k-1} x^{k-1}$ given points $(x_1, f(x_1)), \cdots, (x_k, f(x_k))$, one can solve for the coefficients column in the following system of linear equations:

$$\begin{pmatrix} {x_1}^{k-1} & \cdots & x_1 & 1 \\ {x_2}^{k-1} & \cdots & x_2 & 1 \\ \vdots & \ddots & \vdots & \vdots \\ {x_k}^{k-1} & \cdots & x_k & 1 \end{pmatrix} \begin{pmatrix} a_{k-1} \\ a_{k-2} \\ \vdots \\ s \end{pmatrix} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_k) \end{pmatrix}.$$

This view of interpolation shows that Shamir's scheme is perfect. If there were less than k shares, the above linear system would not have a unique solution for a_0.

Example: Let $k = 3$ and $n = 5$, $p = 7$, and $s = 4$. The dealer also chooses the polynomial

$$f(x) = 5x^2 + x + 4 \bmod 7.$$

Note that we are taking everything in this example over $\mathbb{Z}_7$. The dealer also chooses the public $x_i = i$. Than the shares to $P_1, \cdots, P_5$ are:

$$s_1 = (1, 3)$$

$$s_2 = (2, 5)$$

$$s_3 = (3, 3)$$

$$s_4 = (4, 4)$$

$$s_5 = (5, 1)$$

Now say that P_1, P_2, and P_3 wish to combine their shares to recover the secret. They can then use interpolation to find:

$$f(0) = \sum_{i=1}^{3} y_i \prod_{1 \leq j \leq 3, j \neq i} \frac{-j}{i-j} \bmod 7$$

$$= 3 \cdot \frac{-2}{-1} \cdot \frac{-3}{-2} + 5 \cdot \frac{-1}{1} \cdot \frac{-3}{-1} + 3 \cdot \frac{-1}{2} \cdot \frac{-2}{1} \bmod 7$$

$$2 + 6 + 3 \bmod 7 \equiv 4 = s.$$

3.4 Verifiable Secret Sharing

In the above secret sharing schemes, there are no methods for any one participant to protect themselves from a cheating dealer or participant. For instance, using Shamir's or Blakely's schemes, if a single cheating participant were to lie about their share during the recovery process, their set in the access structure would recover an incorrect secret while the cheating participant would have enough information to recover the secret on his/her own.

Definition 3.4.1. *A* verifiable secret sharing scheme *(VSS) is a secret sharing scheme where extra steps are added to allow participants to verify the consistency of their shares based entirely off of information provided by the dealer before running Recover. It is also necessary that such a verification process requires no interaction between participants.*

In this section we present two verifiable secret sharing schemes based on Shamir's scheme. See also [38] for a verifiable secret sharing scheme using non-commutative groups.

3.4.1 The Discrete Logarithm Problem

Definition 3.4.2. *The* discrete logarithm *problem in a finite cyclic group G of order p is the problem of finding a solution for x in the equation $a^x \equiv b \bmod p$ where $a, b \in G$.*

It is conjectured that there exists no polynomial time algorithm to solve the discrete logarithm problem in certain subgroups of

$(\mathbb{Z}_p)^\times$ where p is a large prime and $p = 2q+1$ where q is also prime. Such prime numbers in cryptography are called *strong primes.*

We do not use $(\mathbb{Z}_p)^\times$ itself for three main reasons. The first two are that its order, $p-1$, is even and therefore not prime and that we want $p-1$ to have large prime factors. Both of these concerns are due to the Pohlig-Hellman algorithm which in some cases can exploit these facts to give more efficient solutions to the discrete logarithm problem. For this reason we work in a subgroup generated by an element of order q, namely the subgroup of quadratic residues. The other advantage is that after $\mathbb{Z}_p$ is determined, it is easy to find a generator for $\mathbb{Z}_q$. Since $\mathbb{Z}_q$ is the subgroup of quadratic residues it consists of all elements of $(\mathbb{Z}_p)^\times$ with square roots. Therefore, a non-trivial element can be obtained, by squaring an element $x \in (\mathbb{Z}_p)^\times$ and checking that x^2 is non-trivial. If it isn't, then it generates the subgroup of quadratic residues since the order of the subgroup is q which is prime.

At this point in time, no efficient algorithms have been implemented to solve the discrete logarithm problem over these groups. As such, at least for now, it is secure for real world cryptography.

3.4.2 Feldman Secret Sharing

In this section we present a VSS scheme due to Feldman [24]. Let p be a large prime such that $p = 2q + 1$ where q is also prime. Also let $g \in \mathbb{Z}_p$ be an element of order q. We choose g and p so that the subgroup generated by g has a difficult discrete logarithm problem. Namely, if $g^x = h \bmod p$, it is difficult to determine x from g and h. In order to verifiably share a secret, $s \in \mathbb{Z}_p$ where n is the number of participants, and k is the threshold, the dealer does the following:

- Assigns to each participant P_i a public $x_i \in \mathbb{Z}_p$ and chooses $a_j \in \mathbb{Z}_p$ randomly for $j = 1, \cdots, k-1$ to create the polynomial
$$f(x) = s + a_1 x + \cdots + a_{k-1} x^{k-1}.$$
- Over a private channel sends each P_i, $f(x_i)$ and further publish q, g^s, and g^{a_i}. Notice that the secret, individual shares, and the coeffecients of the polynomial are kept hidden due to the assumption that the discrete log problem is hard base g.
- In order to verify that their share is consistent, each participant P_i checks that
$$g^{s_i} = g^s g^{a_1 x_i} \cdots g^{a_{k-1} x_i^{k-1}} = g^{s + a_1 x_i + \cdots a_{k-1} x_i^{k-1}} \bmod p.$$

- After each participant has checked their shares, for consistency, they can start the recovery process. Notice that any share P_i encounters can be verified in the above fashion.

3.4.3 Pedersen Secret Sharing

In the above method we rely on the hardness assumptions of the discrete logarithm problem to assume that g^s does not reveal s. In this version due to Pedersen [53], the public information gives no information about s.

Let p, q, g be the same as with Feldman. We also choose $d \in \mathbb{Z}_p$ and set $h = g^d \bmod p$. The dealer generates two random polynomials in $\gamma(x)$ and $\delta(x)$ in $\mathbb{Z}_q[x]$ with coefficients $\{\gamma_i\}_{0\leq i\leq k-1}$ and $\{\delta_i\}_{0\leq i\leq k-1}$ where $\delta_0 = s$ and s is the secret. The dealer publishes $\epsilon_i = g^{\delta_i}h^{\gamma_i} \bmod p$ and privately distributes the share $\{\delta(x_i), \gamma(x_i)\}$ to each P_i. Each participant can then check their shares by checking that:

$$g^{\delta(x_i)}h^{\gamma(x_i)} = (\epsilon_0)(\epsilon_1)^{x_i} \cdots (\epsilon_{k-1})^{{x_i}^{k-1}}.$$

This works since:

$$(\epsilon_0)(\epsilon_1)^{x_i} \cdots (\epsilon_{k-1})^{{x_i}^{k-1}} =$$

$$(g^{\delta_0}h^{\gamma_0})(g^{\delta_1}h^{\gamma_1})^{x_i} \cdots (g^{\delta_{k-1}}h^{\gamma_{k-1}})^{{x_i}^{k-1}} =$$

$$g^{\delta_0+\delta_1 x_i+\cdots+\delta_{k-1}{x_i}^{k-1}} h^{\gamma_0+\gamma_1 x_i+\cdots+\gamma_{k-1}{x_i}^{k-1}} =$$

$$g^{\delta(x_i)} h^{\gamma(x_i)} \bmod p$$

The advantage of this to Feldman's verification is that publishing $g^s h^z = g^s(g^d)^z = g^{s+dz}$ gives information about $s+dz$ which in turn gives no information about s. As such, even an adversary with unbounded computational power could not use that information to find the secret.

3.5 Proactive Secret Sharing

An additional problem with the classical secret sharing schemes of Blakley or Shamir is that there are no provisions on storing shares. When these protocols are enacted using computers it is necessary to store the shares in locations that may be accessed through a network, or are just not perfectly secure. Even if it is difficult to hack into somewhere where secrets are being stored, over a long time period, it may be possible for an adversary to find enough shares to compute the secret themselves.

One possible way to solve this problem would be to update the secret, but in many cases this is not always possible. For instance if the secret were a legal document, you would not want to change it. Even decrypting a ciphertext and then encrypting it with a different key would not solve that problem since decrypting the message multiple times can expose the secret. This may be the case if an adversary is able to hack into a server while it is decrypting.

As such, it makes more sense to update the shares while keeping the secret the same. In this way, any adversary would have to obtain enough shares before all the old shares are erased and the new shares are being used. Any old shares the adversary may have already obtained would then be useless to recover the secret as using them with the new shares would be inconsistent. In this section we present methods that can be seen in Herzberg et. al. [32] and Jarecki [34].

A simple way to update shares using Shamir's scheme would be the following:

- The dealer generates a random polynomial in $\mathbb{Z}_p[x]$,

  $$F(x) = b_1 x + \cdots b_{k-1} x^{k-1}.$$

  In particular, the constant term is 0.

- To each P_i the dealer sends $F(x_i)$ over a private channel.

- Each participant creates their new share $s_i' = s_i + F(x_i)$ where s_i is their original share and then erases s_i.

In this way when the participants want to recover the secret, the polynomial they would obtain after interpolation is $(F + P)(x)$ where $P(x)$ is the original polynomial. Moreover, the constant term in the new polynomial is s meaning that the participants recover the same secret. The above protocol could be amended by not including the dealer, namely having the participants update their shares together. This can be done by doing the following:

- Each participant, P_i randomly generates a polynomial $\delta_i(x) \in \mathbb{Z}_p[x]$ that has constant term 0.
- To each P_j, $i \neq j$, P_i sends P_j, $\delta_i(x_j)$ over a private channel.
- P_i computes its new share $s_i' = s_i + \delta_1(x_i) + \cdots + \delta_n(x_i)$ and erases all the information except for s_i'.

A possible setting for proactive secret sharing is one in which the participants are servers that the shares are stored on. The servers are vulnerable to an adversary that can break into a server and learn secret information it holds, disconnect it, change its behavior, or anything that prevents a server from correctly doing its job. Since any server is vulnerable on some level (very little in an actual setting is perfectly secure) it is in their best interest to update their shares periodically.

In this scenario the servers will update their shares after some specific time period. Each update will consist of share recovery (recovering any corrupted shares) and share renewal. The following can be seen in greater detail in Jarecki's masters thesis [34]. Also note that methods described in the previous section are used to make steps verifiable.

3.5.1 Initialization

We choose, $g, h \in \mathbb{Z}_p$ such that an adversary does not know $d = \log_g h$. We also have $s \in \mathbb{Z}_p$ as the secret and simplify the above protocols so that $x_i = i \bmod p$. To also simplify notation we describe this protocol as having a $(k+1, n)$ access structure.

Each server, P_i, begins with the share $\{{s_i}^{(0)}, {z_i}^{(0)}\}$ such that there are two polynomials $f^{(0)}(x)$ and $g^{(0)}(x)$ where $f^{(0)}(0) = s$, $f^{(0)}(i) = {s_i}^{(0)}$, and $g^{(0)}(i) = {z_i}^{(0)}$.

In addition the servers make use of a secure public key encryption scheme with encryption function *Enc* and a secure signature scheme *Sig*.

3.5.2 Share Renewal

The share renewal protocol borrows elements of Pedersen's VSS in addition to some other technical elements. This is the share renewal protocol for each server at time period t:

- P_i picks $2k$ random elements $\{\delta_{im}, \gamma_{im}\}_{m\in\{1,\cdots,k\}}$ from $\mathbb{Z}_p$ to define the polynomials $\delta_i(x) = \delta_{i1}x + \cdots \delta_{ik}x^k$ and $\gamma_i(x) = \gamma_{i1}x + \cdots \gamma_{ik}x^k$
- P_i computes $\epsilon_{im} = g^{\delta_{im}}h^{\gamma_{im}}$ for all $1 \leq m \leq k$ and $u_{ij} = \delta_i(j)$ and $w_{ij} = \gamma_i(j)$ for $1 \leq j \leq n$.
- P_i broadcasts $VSS_i^{(t)} = (i, t, \{\epsilon_{im}\}_{m=1}^{k}, \{Enc_j(u_{ij}, w_{ij})\}_{j\neq i}^{n})$ and the signature $Sig_i(VSS_i^{(t)})$. In this case Enc_j and Sig_i are encryption using a public key provided by P_j and signing a message with the private key of P_i.
- P_i decrypts the $\{u_{ji}, w_{ji}\}$ and verifies them by checking:

$$g^{u_{ji}}h^{w_{ji}} = \prod_{m=1}^{k}(\epsilon_{jm})^{i^m}(\text{mod} p)$$

P_i then broadcasts a message stating if the verification was successful or unsuccessful.

- Based on a voting procedure the servers can together determine the set of servers that have been corrupted. The details of the process can be found in [34]. It is important to note that different kinds of corruption can result in having a server broadcast a faulty message of success or failure in the previous step. In doing this step each P_i is able to determine which servers sent out correct information in the previous step and use those servers to update their shares.

- P_i then updates its share in the method mentioned previously. Namely, it computes:

  $$s_i^{(t)} = s_i^{(t-1)} + \sum_j u_{ji}, z_i^{(t)} = z_i^{(t-1)} + \sum_j w_{ji}$$

  where the j are taken over the non-corrupted servers identified in the previous step.

3.5.3 Share Recovery

An important part of the proactive secret sharing in [34] is the share recovery phase where corrupted servers get new shares. Although, the method of doing so will not be detailed here, the basic idea is simple to understand. We also let the corrupted servers get indexed by r rather than i or j.

For each corrupted participant P_r, each non-corrupted server P_i creates polynomials $\delta_i(x)$ and $\gamma_i(x)$ such that $\delta_i(r) = \gamma_i(r) = 0$. P_i then creates the new share

$$s_i' = s_i + \sum \delta_j(i), z_i' = z_i + \sum \gamma_j(i)$$

where the j are taken over the non-corrupted servers and x_i is P_i's current share. Each P_i sends this to P_r in a similar fashion to VSS_i in the previous step so that all the necessary information is private and verifiable. After verification, P_r interpolates over the s_i' and the z_i' to determine their new share (s_r, z_r). This share is consistent, namely s_r and z_r lie on the same polynomials as all the other s_i and z_i. This is because each (s_i', z_i') lies along the polynomial

$$(\delta(x) + \sum \delta_j(x), \gamma(x) + \sum \gamma_j(x))$$

where the (s_i, z_i) lie along $(\delta_i(x), \gamma_i(x))$. Now note:

$$(\delta(r) + \sum \delta_j(r), \gamma(r) + \sum \gamma_j(r)) = (\delta(r), \gamma(r)) .$$

3.6 Secret Sharing Using Non-Abelian Groups

We now introduce a secret sharing scheme due to Habeeb, Kahrobaei, and Shpilrain [30]. To use this secret sharing scheme, we require a finitely presented group G, with an efficiently solvable word problem. By this we mean that there exists an efficient algorithm that takes as input a word w written in the generators of G and decides if w represents the trivial word. We first present an (n, n) threshold scheme and then expand it to the (k, n) case for arbitrary k by using Shamir's scheme.

3.6.1 (n, n) Threshold Scheme

In this case the secret, s, is an element of $\{0, 1\}^k$ which we view as a column vector. The setting is initialized by making the set of letters, $X = \{x_1, \cdots, x_n\}$, public. To distribute the shares the dealer does the following:

- Distributes to each P_i over a private channel a set of words R_i in the alphabet $X^{\pm 1}$ that define the group $G_i = \langle X | R_i \rangle$.
- Randomly generates the shares $s_i \in \{0, 1\}^k$ for $i = 1, \cdots, n-1$ and $s_n = s - \sum_{j=0}^{n-1} s_j$ where the addition is bitwise addition in $\mathbb{F}_2^m$. Note that each s_i can be seen as a vector in $\mathbb{F}_2^m$.

- Publishes words w_{ji} over the alphabet $X^{\pm 1}$ such that a word w_{ji} is trivial in G_i if $s_{ji} = 1$ and non-trivial if $s_{ji} = 0$.

The participant P_i then finds their share s_i by deciding if each of the w_{ji} are trivial or non-trivial for $1 \leq j \leq k$. By viewing trivial words as 1 in their vector and non-trivial words as 0, P_i can successfully recreate s_i. Following that, all participants can add their shares to recover the secret.

Note that even though the w_{ji} are sent over an open channel, the shares remain secure since the R_i are private. Therefore no other participant can recover s_i from the w_{ji} since only P_i knows G_i.

3.6.2 (k, n) Threshold Scheme

In [30] the authors extend the above scheme to a (k, n) threshold via Shamir's scheme. As is the case with Shamir's scheme, the secret s is an element of $\mathbb{Z}_p$ and the shares, s_i, correspond to points on a polynomial of degree $k - 1$ with constant term s. In order to use the same logic as the (n, n) threshold presented above, participants view their share as its binary representation and participants fully recover their shares by checking which words are trivial in their individual group. After each participant has reconstructed their share, the secret can be recovered via polynomial interpolation.

- The dealer randomly selects $a_1, \cdots, a_{k-1} \in \mathbb{Z}_p$ such that $a_{k-1} \neq 0$ and constructs the polynomial $f(x) = a_{k-1}x^{k-1} + \cdots + a_1x + s$
- For each participant P_i the dealer publishes a corresponding $x_i \in \mathbb{Z}_p$. The dealer then converts each $s_i = f(x_i)$ into binary. And thus, each s_i can be viewed as a column vector of length $l = \log_2 p + 1$
- As was the case in the (n, n) scheme, the dealer distributes the s_i over an open channel by sending each P_i the words $w_{1i}, \cdots, w_{li}$ over the alphabet $X^{\pm}$ such that w_{ji} is trivial in G_i if $s_{ji} = 1$ and non-trivial if $s_{ji} = 0$.
- The participants reconstruct their own s_i as in the (n, n) scheme and can recover the secret using polynomial interpolation.

A major advantage of the scheme over the classical ones is that the initial private relators are distributed by the dealer, the dealer can continue using them to distribute more secrets. This is as opposed to Blakley's or Shamir's scheme where new shares need to be sent out for each successive secret.

In that vein, the scheme is vulnerable to an adversary determining the relators by potentially seeing patterns in words they learn are trivial. Namely, after a participant reveals their share (possibly while recovering the secret) an adversary could potentially determine which of the w_{ji} were trivial and possibly determine the group presentation of G_i which would allow them to construct P_i's share on their own. In the following sections, we will discuss this weakness, and see that there are adjustments that can be made where it isn't a security concern.

Another advantage to this scheme is that since it is based on the Shamir secret sharing protocol it can benefit from the large amount of research done on Shamir's scheme. For instance, the verification methods or proactive secret sharing protocols from [61] and [32] can still be used in this scheme.

3.6.3 Small Cancellation Groups

In this section we introduce the candidate platform group from [30] for the above secret sharing scheme.

Definition 3.6.1. *A word w is* cyclically reduced *if it is reduced in all of its cyclic permutations.*

Note that this only occurs if the word is freely reduced, it has no subwords of the form $x_i^{-1}x_i$ or $x_ix_i^{-1}$, and the first and last letters are not inverses of each other.

Definition 3.6.2. *A set of words R is called* symmetrized *if each word is cyclically reduced and the entire set and their inverses are closed under cyclic permutation.*

If R is viewed as a set of relators, symmetrizing R does not change the resulting group as the closure R under cyclic permutations and inverses is a subset of the normal closure.

Definition 3.6.3. *Given a set R we say that v is a* piece *if it is a maximal initial subword of two different words, namely if there exist $w_1, w_2 \in R$ such that $w_1 = vr_1$ and $w_2 = vr_2$. A group $G = \langle X|R\rangle$ satisfies the* small cancellation condition $C'(\lambda)$ *for $0 < \lambda < 1$ if for all $r \in R$ such that $r = vw$ where v is a piece, then $|v| < \lambda|r|$.*

Small cancellation groups satisfying $C'(\frac{1}{6})$ have linear time algorithm for the word problem [17] making them an ideal candidate for the HKS secret sharing scheme. Moreover, it can be seen from their definition that if the number of generators and the length of the relators are large compared to the number of relators, it is likely that there will be small cancellation since the probability that any two words have a large maximal initial segment is low. After generating a random set of relators satisfying the above properties, it is also fast to symmetrize the set and then find the pieces and check that they are no larger than one sixth of the word. As such, it is fast to create such groups by repeatedly randomly generating relators, symmetrizing, and checking to see if they satisfy the $C'(\frac{1}{6})$ condition. There are other groups that have an efficient word problem that could also function as candidate groups, but small cancellation groups have the advantage of being efficient to generate randomly.

3.6.4 Secret Sharing and the Shortlex Ordering

In this section we present an adjustment to the HKS secret sharing scheme using the shortlex ordering on a group. This is original work from [15]. Let $X = \{x_1, \cdots, x_n\}$ and $G = \langle X \rangle$. A shortlex ordering on G is induced by an order on $X^{\pm 1}$ as follows. Given reduced $w = x_{i_1} \cdots x_{i_p}$ and $l = x_{j_1} \cdots x_{j_k}$ with $w \neq l$ then $w < l$ if and only if:

- $|w| < |l|$
- or if $p = k$ and $x_{i_a} < x_{j_a}$ where $a = \min_\alpha\{x_{i_\alpha} \neq x_{j_\alpha}\}$

For example, let $X = \{x, y\}$ and give $X^{\pm}$ the ordering $x < x^{-1} < y < y^{-1}$. Then some of the first words in order would be: $e < x < x^{-1} < y < y^{-1} < x^2 < xy < xy^{-1} < x^{-2} < x^{-1}y < x^{-1}y^{-1} < yx < yx^{-1} < y^2 < y^{-1}x < y^{-1}x^{-1} < y^{-2} < x^3 < x^2y < x^2y^{-1} < xyx < xyx^{-1} < \cdots$ This method of counting group elements can be used to combine group theory with the numerical aspects present in many other cryptographic schemes. For instance, the shortlex ordering can be used to modify the secret sharing scheme above:

- In this case, the dealer publishes the letters X and over a private channel sends a set of words, R_i in $X^{\pm 1}$ to each P_i such that $G_i = \langle X | R_i \rangle$ is a group with an efficient algorithm to reduce words to some normal form with respect to the R_i.
- The dealer chooses a secret $s \in \mathbb{Z}_p$ for some large prime $p > n$ and generates a random polynomial, f in $\mathbb{Z}_p[x]$ with constant term s.
- The dealer assigns a public $x_i \in \mathbb{Z}_p$ to each participant, computes $f(x_i)$, and finds $s_i \in F(X)$ such that s_i is the $f(x_i)^{th}$

word in $F(X)$.

- The dealer publishes a word w_i that reduces to s_i in G_i. This can be done efficiently by interspersing conjugated products of relators between the letters of s_i.
- Each participant P_i computes their share by reducing w_i to get s_i and then computing its position in $F(X)$.
- Using their shares they find the secret using polynomial interpolation.

The main advantage of this new method is that participants need only reduce one word rather than a number of words corresponding to the length of the secret. This also prevents more information being released pertaining to private relators. Reducing words is more general than being able to solve the word problem in a finitely presented group and in some cases may be more complex.

It is important to note the following about this scheme:

- Given an algorithm that reduces words, each w_i must reduce uniquely to s_i. This implies that if our reduction algorithm does not terminate at s_i, then it is not a viable share for this scheme. In that case, if a random $f(x_i)$ does not correspond to a fully reduced word, the dealer can always assign P_i a different x_i. It may also be necessary to check that each w_i reduces to s_i give the reduction algorithm before the shares are distributed.

- Some reduction algorithms can be done in multiple ways given the same initial conditions, so it is important to fix a protocol so that whatever process P_i uses to reduce w_i terminates at s_i.

3.6.5 Platform Group

For this variant of the HKS secret sharing scheme, we also propose $C'(\frac{1}{6})$ groups as described in **Section 3.6.3**. Additionally, we propose the parameters $|X| = 40$, $|R| = 4$, and $|r| = 9$ for all $r \in R$. We find that with such parameters, generating a single $C'(\frac{1}{6})$ group can be done in roughly 1 second in GAP [26] by generating random relators of the given length and then checking that the set of relators satisfies the small cancellation condition. In order to reduce the w_i to s_i, participants can use Dehn's algorithm which terminates in linear time [17]. It is not guaranteed in general that Dehn's algorithm will reduce each w_i to s_i, as such it is necessary to check that each w_i reduces to s_i. In order to test the efficacy of Dehn's algorithm in $C'(\frac{1}{6})$ groups for the purposes of this secret sharing scheme, we performed the following tests in GAP [26]:

- Generated 10 small cancellation groups using the parameters from the first paragraph of this section.
- In each group we generated 100 words of length less than 10 and created corresponding large unreduced words of length 500 by inserting conjugated products of relators between letters in our original word.
- Applied an implementation of Dehn's algorithm due to Chris Staecker [62] and checked that our unreduced word successfully reduced to the original word.

After running said tests, we found that Dehn's algorithm successfully reduced every word. The size considerations in the second item were given in part because there are enough non-trivial, Dehn reduced, words of length 10 or less in the free group on 40 generators to be used as shares in a practical setting. If we let B_{10} denote the set of non-trivial words of F_{40} of length 10 or less, we have that:

$$|B_{10}| = 80(79^9 + \cdots 79 + 1) \approx 1.23 \times 10^{17}.$$

Note that among those words only a small fraction are not Dehn reduced. In this vein, when the size of the generating set is large, the lengths of the s_i can be small while the corresponding $f(x_i)$ are large.

3.6.6 Efficiency

Each step in modified HKS scheme can be done efficiently. As mentioned previously, generating $C'(\frac{1}{6})$ groups can be done quickly by repeatedly generating sets of relators and checking to see if they satisfy the necessary small cancellation condition. The necessary computations using the shortlex ordering can be done using basic combinatorial formulas that are very fast for a computer to evaluate. Additionally, the w_i can be created efficiently from the s_i by inserting conjugated products of relators and then reduced in polynomial time using Dehn's algorithm. Hence each additional step to the standard Shamir's scheme can be done efficiently. This is also an improvement over the standard HKS scheme since the amount of words that need to be reduced is independent of the length of the secret, making it possible for larger secrets to be distributed efficiently.

3.6.7 Updating Relators

The main security concern for this protocol is the possibility of an adversary discovering a participant's set of relators. As more secrets are shared and it is revealed which words in a participant's group are trivial, it may eventually be clear what the relators are. As such to optimize security it may be useful to have a secure method of updating relators. In this section, we present such a method.

To update relators we add steps that can take place before any new secret is sent out:

- For each P_i the dealer creates a set of words, R_i', in $X^{\pm 1}$ such that $G_i = \langle X | R_i' \rangle$ satisfies the same desired properties.
- In order to distribute each $r \in R_i'$, the dealer pads r with relators in R_i as done previously and publishes them.
- P_i then reduces r by using the relators in R_i.
- After the full set of words in R_i' is published and reduced, P_i deletes the original R_i and sets $R_i := R_i'$.

If these steps are performed before an adversary can figure out a set of relators, then the protocol remains secure over a long period of time since information about previous relators becomes useless. Note the words in R_i' must be reduced with respect to the original R_i so that P_i can effectively form a new group. In this way R_i and R_i' are not completely unrelated, but as the relators become updated each additional time, they will have less and less to do with the original set of relators.

Bibliography

[1] Iris Anshel, Michael Anshel, and Dorian Goldfeld, *An algebraic method for public-key cryptography*, Mathematical Research Letters **6** (1999), 287–292.

[2] Sanjeev Arora and Boaz Barak, *Computational complexity: a modern approach*, Cambridge University Press, 2009.

[3] Goulnara Arzhantseva, Jean-François Lafont, and Ashot Minasyan, *Isomorphism versus commensurability for a class of finitely presented groups*, Journal of Group Theory **17** (2014), no. 2, 361–378.

[4] Amos Beimel, *Secret-sharing schemes: a survey*, Proceedings of the Third international conference on Coding and cryptology (Berlin, Heidelberg), IWCC'11, Springer-Verlag, 2011, pp. 11–46.

[5] Robert Bieri, Walter D Neumann, and Ralph Strebel, *A geometric invariant of discrete groups*, Inventiones mathematicae **90** (1987), no. 3, 451–477.

[6] George R. Blakley, *Safeguarding cryptographic keys*, Proceedings of the 1979 AFIPS National Computer Conference (Monval, NJ, USA), AFIPS Press, pp. 313–317.

[7] Oleg Bogopolski, Armando Martino, Olga Maslakova, and Enric Ventura, *The conjugacy problem is solvable in free-by-cyclic groups*, Bulletin of the London Mathematical Society **38** (2006), no. 05, 787–794.

[8] Oleg Bogopolski, Armando Martino, and Enric Ventura, *The automorphism group of a free-by-cyclic group in rank 2*, Communications in Algebra **35** (2007), no. 5, 1675–1690.

[9] ______, *Orbit decidability and the conjugacy problem for some extensions of groups*, Transactions of the American Mathematical Society **362** (2010), no. 4, 2003–2036.

[10] Kenneth S Brown, *Trees, valuations, and the bieri-neumann-strebel invariant*, Inventiones mathematicae **90** (1987), no. 3, 479–504.

[11] AM Brunner, James McCool, and Alfred Pietrowski, *Groups which are an infinite cyclic extension of a unique base group*, Journal of the Australian Mathematical Society (Series A) **23** (1977), no. 04, 499–503.

[12] Bren Cavallo, Jorge Delgado, Delaram Kahrobaei, Ha Lam, and Enric Ventura, *Tits alternative for the automorphism group of a rigid poly-$\mathbb{Z}$ group*, Preprint (2014).

[13] Bren Cavallo and Delaram Kahrobaei, *A family of polycyclic groups over which the uniform conjugacy problem is np-complete*, International Journal of Algebra and Computation **24** (2014), no. 4, 515–530.

[14] ______, *A polynomial time algorithm for the conjugacy problem in $\mathbb{Z}^n \rtimes \mathbb{Z}$*, Reports@ SCM **1** (2014), no. 1, 55–60.

[15] ______, *Secret sharing using non-commutative groups and the shortlex order*, Contemporary Mathematics **633** (2015), 1–8.

[16] Donald J Collins and Charles F Miller, *The conjugacy problem and subgroups of finite index*, Proceedings of the London Mathematical Society **3** (1977), no. 3, 535–556.

[17] B. Domanski and M. Anshel, *The complexity of dehn's algorithm for word problems in groups*, J. Algorithms (1985), 543–549.

[18] Cornelia Druţu and Michael Kapovich, *Lectures on geometric group theory*, preprint (2013).

[19] Bettina Eick, *Algorithms for polycyclic groups*, Habilitationsschrift, Universitat Kassel, 2001.

[20] ______, *When is the automorphism group of a virtually polycyclic group virtually polycyclic?*, Glasgow Mathematical Journal **45** (2003), no. 03, 527–533.

[21] Bettina Eick and Delaram Kahrobaei, *Polycyclic groups: A new platform for cryptology?*, arXiv preprint math/0411077 (2004).

[22] Bettina Eick and Gretchen Ostheimer, *On the orbit-stabilizer problem for integral matrix actions of polycyclic groups*, Mathematics of computation **72** (2003), no. 243, 1511–1529.

[23] Joan Feigenbaum (ed.), *Advances in cryptology - CRYPTO '91*, Lecture Notes in Computer Science, vol. 576, Springer, 1992.

[24] Paul Feldman, *A practical scheme for non-interactive verifiable secret sharing*, Proceedings of the 28th Annual Symposium on Foundations of Computer Science (Washington, DC, USA), SFCS '87, IEEE Computer Society, 1987, pp. 427–438.

[25] Edward Formanek, *Conjugate separability in polycyclic groups*, Journal of Algebra **42** (1976), no. 1, 1–10.

[26] The GAP Group, *GAP – Groups, Algorithms, and Programming, Version 4.7.6*, 2014, http://www.gap-system.org.

[27] David Garber, Delaram Kahrobaei, and Ha T Lam, *Length-based attacks in polycyclic groups*, Journal of Mathematical Cryptography (to appear).

[28] David Garber, Shmuel Kaplan, Mina Teicher, Boaz Tsaban, and Uzi Vishne, *Length-based conjugacy search in the braid group*, Contemporary Mathematics **418** (2006), 75–87.

[29] Volker Gebhardt, *Efficient collection in infinite polycyclic groups*, Journal of Symbolic Computation **34** (2002), no. 3, 213–228.

[30] Maggie Habeeb, Delaram Kahrobaei, and Vladimir Shpilrain, *A secret sharing scheme based on group presentations and the word problem*, Contemp. Math., Amer. Math. Soc. **582** (2012), 143– 150.

[31] Maggie E Habeeb, *Groups, complexity, cryptology*, City University of New York, 2012.

[32] Amir Herzberg, Markus Jakobsson, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung, *Proactive public key and signature systems*, Proceedings of the 4th ACM conference on Computer and communications security (New York, NY, USA), CCS '97, ACM, 1997, pp. 100–110.

[33] Derek F. Holt, Bettina Eick, and Eamonn A. O'Brien, *Handbook of computational group theory*, CRC Press, 2005.

[34] Stanislaw M. Jarecki, *Proactive secret sharing and public key cryptosystems*, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 1996.

[35] Delaram Kahrobaei and Michael Anshel, *Decision and search in non-abelian Cramer-Shoup public key cryptosystem.*, Groups Complexity Cryptology **1** (2009), no. 2, 217–225.

[36] Delaram Kahrobaei and Bilal Khan, *A non-commutative generalization of ElGamal key exchange using polycyclic groups.*, GLOBECOM, 2006.

[37] Delaram Kahrobaei and Charalambos Koupparis, *Non-commutative digital signatures using non-commutative groups*, Groups, Complexity, Cryptology **4** (2012), 377–384.

[38] Delaram Kahrobaei and Elizabeth Vidaurre, *Publicly verifiable secret sharing using non-abelian groups*, Contemporary Mathematics **582** (2012), 175–180.

[39] Ravindran Kannan and Richard J Lipton, *Polynomial-time algorithm for the orbit problem*, Journal of the ACM (JACM) **33** (1986), no. 4, 808–821.

[40] Richard M Karp, *Reducibility among combinatorial problems*, Springer, 1972.

[41] Jonathan Katz and Yehuda Lindell, *Introduction to modern cryptography*, Chapman and Hall/CRC Press, 2007.

[42] Hans Kellerer, Ulrich Pferschy, and David Pisinger, *Knapsack problems*, Springer, 2004.

[43] Ki Hyoung Ko, Sang Jin Lee, Jung Hee Cheon, Jae Woo Han, Ju-sung Kang, and Choonsik Park, *New public-key cryptosystem using braid groups*, Advances in Cryptology, CRYPTO 2000, Springer, 2000, pp. 166–183.

[44] Nic Koban, Jon McCammond, and John Meier, *The bns-invariant for the pure braid groups*, arXiv preprint arXiv:1306.4046 (2013).

[45] Charles R. Leedham-Green and Leonard H. Soicher, *Collection from the left and other strategies*, Journal of Symbolic Computation **9** (1990), no. 5, 665–675.

[46] Roger C Lyndon and Paul E Schupp, *Combinatorial group theory*, vol. 89, Springer, 2001.

[47] Ueli M. Maurer (ed.), *Advances in cryptology - eurocrypt '96, international conference on the theory and application of cryptographic techniques, saragossa, spain, may 12-16, 1996, proceeding*, Lecture Notes in Computer Science, vol. 1070, Springer, 1996.

[48] Charles F Miller III, *Decision problems for groupssurvey and reflections*, Algorithms and classification in combinatorial group theory, Springer, 1992, pp. 1–59.

[49] Alex D Myasnikov and Alexander Ushakov, *Length based attack and braid groups: cryptanalysis of anshel-anshel-goldfeld key exchange protocol*, Public Key Cryptography–PKC 2007, Springer, 2007, pp. 76–88.

[50] Alexei Myasnikov, Andrey Nikolaev, and Alexander Ushakov, *Knapsack problems in groups*, To appear in Mathematics of Computation. Available at http://arxiv.org/abs/1302.5671.

[51] Alexei Myasnikov, Vladimir Shpilrain, and Alexander Ushakov, *Group-based cryptography*, Springer, 2008.

[52] Walter D Neumann, *Normal subgroups with infinite cyclic quotient*, Math. Sci **4** (1979), no. 2, 143–148.

[53] Torben P. Pedersen, *Non-interactive and information-theoretic secure verifiable secret sharing*, in Feigenbaum [23], pp. 129–140.

[54] Vladimir N Remeslennikov, *Conjugacy in polycyclic groups*, Algebra and Logic **8** (1969), no. 6, 404–411.

[55] Andrew W Sale, *Short conjugators in solvable groups*, arXiv preprint arXiv:1112.2721 (2011).

[56] ______, *Conjugacy length in group extensions*, arXiv preprint arXiv:1211.3144 (2012).

[57] ______, *The geometry of the conjugacy problem in wreath products and free solvable groups*, arXiv preprint arXiv:1307.6812 (2013).

[58] Dan Segal, *Decidable properties of polycyclic groups*, Proc. London Math. Soc.(3), vol. 61, Citeseer, 1990, pp. 497–528.

[59] Daniel Segal, *Polycyclic groups*, no. 82, Cambridge University Press, 2005.

[60] Adi Shamir, *How to share a secret*, Commun. ACM **22** (1979), no. 11, 612–613.

[61] Markus Stadler, *Publicly verifiable secret sharing*, in Maurer [47], pp. 190–199.

[62] Chris Staecker, *dehn.gap*, http://cstaecker.fairfield.edu/%7Ecstaecker/files/gap/dehn.gap.

[63] Jacques Tits, *Free subgroups in linear groups*, Journal of Algebra **20** (1972), no. 2, 250–270.

[64] Neal R Wagner and Marianne R Magyarik, *A public-key cryptosystem based on the word problem*, Advances in Cryptology, Springer, 1985, pp. 19–36.