Master Thesis

Polycyclic groups and applications in cryptography

Katharina Boudgoust

26th March 2018

Supervisors: Prof. Dr. Frank Herrlich,
Prof. Dr. Jörn Müller-Quade,
Brandon Broadnax and
Alexander Koch

Department of Mathematics

Karlsruhe Institute of Technology

Abstract

In this thesis we analyze the conjugacy search problem (CSP) in polycyclic groups to be used in cryptography. Therefore, we prove that it is computable in the more general class of polycyclic-by-finite groups, whereas the presented algorithm needs much time. For the special subclasses of finite polycyclic [MN89], finitely generated nilpotent [Sim94], supersolvable and virtually finitely generated nilpotent groups [Mon17] there exist faster working algorithms for the CSP, which we study in this work. Additionally, we look at the CSP in some semidirect products of polycyclic groups and show that it is Cook equivalent to the CSP in the underlying groups. Furthermore, we analyze the correctness and security of the AAG key exchange protocol [AAG99] instantiating it with polycyclic groups. In order to study the so-called Length-Based Attack (LBA) which attacks the AAG protocol, we look at the length function proposed by D. Garber et al. [GKL15] and test it using several types of polycyclic groups.

Zusammenfassung

In dieser Arbeit untersuchen wir das Konjugationssuchproblem (KSP) in polyzyklischen Gruppen zur Anwendung in der Kryptographie. Hierfür zeigen wir dessen Lösbarkeit in der allgemeineren Klasse der polycyclic-by-finite Gruppen, wobei der vorgestellte Algorithmus sehr langsam ist. Für die speziellen Unterklassen der endlichen polyzyklischen [MN89], der endlich erzeugten nilpotenten [Sim94], der überauflösbaren und der virtuell endlich erzeugten nilpotenten Gruppen [Mon17] gibt es schnellere Lösungsalgorithmen für das KSP, die wir in dieser Arbeit untersuchen. Weiter betrachten wir das KSP in einigen semidirekten Produkten von polyzyklischen Gruppen und zeigen, dass es Cook äquivalent zum KSP in den zugrundeliegenden Gruppen ist. Darüber hinaus überprüfen wir das AAG Schlüsselaustauschprotokoll [AAG99] auf Korrektheit und Sicherheit, sofern es mit polyzyklischen Gruppen instanziiert wird. Um den sogenannten längenbasierten Angriff (LBA) auf das AAG Protokoll zu untersuchen, betrachten wir die von D. Garber et al. [GKL15] vorgeschlagene Längenfunktion und testen diese anhand verschiedener Typen von polyzyklischen Gruppen.

Contents

List of Figures

List of Tables

List of Algorithms

List of symbols

$(\mathbb{Z}/p\mathbb{Z})^\times$	Unit group of a cyclic group of order p with p prime
$\mathrm{GL}(n, \mathbb{Z})$	General linear group of degree n over $\mathbb{Z}$
$\mathrm{Aut}(\mathrm{G})$	Automorphism group of G
$[G : H]$	Index of a subgroup H in a group G
H^G	Normal closure of a subgroup $H \leq G$
H_G	Normal core of a subgroup $H \leq G$
$u \sim v$	u and v are conjugate
u^w	Conjugate of u by w
$[g, h]$	Commutator of two elements g, h of a group G
$[H, K]$	Commutator of two subgroups H, K of a group G
G'	Derived subgroup of a group G
$G^{[i]}$	i-th factor of the derived series of a group G
$\gamma_i(G)$	i-th factor of the lower central series of a group G
$R(X)$	Sequence of relative orders of a polycyclic sequence X
$I(X)$	Set of finite relative orders of a polycyclic sequence X
$h(G)$	Hirsch length of a polycyclic group G
$PC\,\langle X \vert R\rangle$	Polycyclic presentation with generating set X and relation set R
$C_G(g)$	Centralizer of an element g in a group G
$Z(G)$	Center of a group G
$Fitt(G)$	Fitting subgroup of a group G
$[F : \mathbb{Q}]$	Degree of an algebraic number field F
$\mathcal{O}_F$	Maximal order of an algebraic number field F
U_F	Unit group of an algebraic number field F
$H \ltimes_\alpha N$	Semidirect product of N and H with respect to α
$G_1 *_A G_2$	Amalgamated free product of two groups G_1 and G_2 over a group A
$G_{*\phi}$	HNN-extension of a group G with respect to an isomorphism ϕ

Introduction

Overview

A group is called *polycyclic* if it exhibits a normal series of cyclic factors. As a natural generalization of cyclic groups, *"we would expect its structure to be pretty transparent."* [Seg83, p. ix] Nevertheless, they turn out to possess particularly interesting and complex properties which open up a wide field of research. Forming one of the largest classes of finitely presented groups which are closed under forming subgroups, factor groups and extensions, polycyclic groups constitute a very important area of research in group theory. Additionally, many mathematical problems concerning polycyclic groups and even the more general class of *polycyclic-by-finite* groups, like the word decision problem or the conjugacy decision problem, are computable. The latter are groups which have a polycyclic normal subgroup of finite index.

Besides the mathematical motivation, there is also a cryptographic interest to study polycyclic groups. Most of the current public key cryptography (PKC) is based on commutative algebraic structures, like the discrete logarithm problem or the factorization problem, using cyclic groups of the type $(\mathbb{Z}/p\mathbb{Z})^{\times}$ with p prime or subgroups of elliptic curves, see the fundamental work by J. Katz and Y. Lindell [KL15, Chapter 8] for further details. By the time quantum computers are developed, these two problems can be computed in polynomial-time using P. Shor's algorithm [Sho94]. One good quantum resistant alternative seems to be lattice-based cryptography which offers a large catalog of applications and well understood security assumptions, also resting upon commutative structures.

Another interesting approach is to take non-commutative algebraic structures as a base for PKC schemes. Several cryptographic protocols based on non-commutative groups have been introduced in the past 20 years, see, for instance, [KK06], [KLC$^+$00] or [KK12]. One of the first was the Anshel-Anshel-Goldfeld (AAG) key exchange protocol [AAG99] described in more detail in Chapter 7. With the development of new protocols, also attacks against them have been discovered, as for the AAG protocol the so-called Length-Based Attack (LBA) was found, see Section 8.1.

In 2004, B. Eick and D. Kahrobaei [EK04] suggested the use of polycyclic groups in cryptography. Polycyclic groups have computable word, conjugacy and isomorphism decision problems, all of them will be introduced in detail in Section 6.1. Whereas the word problem can be computed using a fast working method called collection, presented in Section 6.2, there is no known polynomial-time solution to the conjugacy problem so far.

Furthermore, the polycyclic groups of a special subclass, namely the non-virtually nilpotent ones, exhibit an exponential growth rate which ensures a fast increasing key space for the AAG protocol, as pointed out in Section 7.3. These properties seem to be necessary for use in non-commutative-group-based cryptography and therefore, one may think of polycyclic groups as suitable platform groups.

As we will see in this work, there are many unanswered questions related to the AAG protocol using polycyclic groups in particular and to cryptographic protocols based on the conjugacy search problem in general. Another open task is to find good candidates to instantiate polycyclic groups for use in PKC.

Structure

The work is structured as follows: In Chapter 1, we introduce polycyclic groups and different ways to represent them. Besides the possibility to define them with the help of a normal series and polycyclic sequences, they also possess a special finite group presentation which is helpful for computational purposes. Additionally, they can be considered as subgroups of matrix groups.

Subsequently, we list some examples of polycyclic groups in Chapter 2 to further our understanding of them. We consider supersolvable, finitely generated nilpotent and poly-infinite cyclic groups, all of them will be introduced in detail.

Polycyclic groups are residually finite, which is an important property needed later on in this work. As this property was shown for a more general class of groups, namely for polycyclic-by-finite groups, we shortly introduce them in Chapter 3 and subsequently state the proof of residually finiteness.

In Chapter 4, a prototype of polycyclic groups which is often used for experimental purposes is introduced. Later on in Section 8.2 a special attack against this type will be presented.

Moreover, we recap in Chapter 5 some important notions of computation theory like the concept of Turing machines, computable functions and reduction, as well as basic notions of computational security, so that we will be able to use it subsequently. For instance, we define what it means for a key exchange protocol to be correct and one-way secure. Such precise definitions are very important in cryptography to ensure the reliability of a protocol.

In order to construct concrete PKC schemes which use polycyclic groups as platform groups, we have to find problems related to polycyclic groups which are supposedly hard to compute. Therefore, we first look in Chapter 6 at the three important group-based decision problems introduced by M. Dehn [Deh11], namely the word, the conjugacy and the isomorphism decision problem. Subsequently, we show that those problems are computable in the more general class of polycyclic-by-finite groups and present a method called collection, which computes the word problem for polycyclic groups much faster.

In cryptography the subgroup-restricted multiple conjugacy search problem, a variation of the conjugacy problem, is used. Whereas this problem is assumed to be hard to compute for polycyclic groups in general, there are better solutions for the multiple conjugacy search problem (MCSP) in some subclasses. These methods are presented for finite polycyclic groups [MN89], for finitely generated nilpotent groups [Sim94] and for virtually finitely generated nilpotent groups [Mon17]. Additionally, we show that supersolvable groups are virtually finitely generated nilpotent, hence they also feature a better solution to the MCSP. We close the chapter with proving that the CSP in some (semi)direct products of polycyclic groups is Cook equivalent to the CSP in the original groups.

Furthermore, we present the AAG key exchange protocol instantiated with polycyclic groups in Chapter 7. We analyze its correctness and security, whereas we only find a necessary assumption for the security of the AAG protocol but no hardness reduction. As a second part of this chapter, we point out that the growth rate of the platform group is also crucial for the security of the AAG protocol.

In Chapter 8 we review two of the known attacks against the AAG protocol, namely the Length-Based Attack (LBA) and the Field-Based Attack (FBA). After we have presented the main idea of the LBA, we discuss the role of the length function proposed by D. Garber et al. [GKL15] when testing it on different types of polycyclic groups. Afterwards, we present two variants of the LBA, the LBA with backtracking and the Memory LBA, two out of many versions which have been developed in the last decade, see [MU07] or [GKT$^+$06]. The chapter closes with a presentation of the FBA.

Chapter 9 contains a discussion about further work within this field of research. There are many unanswered questions concerning algorithms to solve the conjugacy problem in polycyclic groups, e.g., whether the subgroup-restricted MCSP is hard enough and even quantum resistant. We also need to find a hardness assumption which implies the security of the AAG protocol. A precise construction of suitable polycyclic groups to be used in cryptography is also an unsolved problem.

Acknowledgement

First of all, I want to thank Prof. Dr. Frank Herrlich and Prof. Dr. Jörn Müller-Quade who encouraged me to study this special topic for my master thesis. During my work they supported me with helpful ideas and suggestions on how to approach arising problems. Moreover, I would like to express my gratitude to Brandon Broadnax and Alexander Koch who gave me constructive feedback and listened patiently to my thoughts during this thesis. Last but not least I am grateful to my family and friends who never ran out of good advice.

1. Polycyclic groups

As a first step, we will introduce polycyclic groups and recapitulate some important results about them: They can be represented with the help of polycyclic series, polycyclic sequences or polycyclic presentations. We will also show their connection to the class of solvable groups and to subgroups of the matrix group $GL(n, \mathbb{Z})$ for some $n \in \mathbb{N}$.

1.1. Polycyclic series

The definition follows the notation of the *Handbook of computational group theory* by D. Holt et al. [HEO05, Chapter 8]. The study of polycyclic groups was initiated by K.A. Hirsch [Hir38] where he called these groups initially *S-groups*.

Definition 1.1. *A group G is called* polycyclic *if it exhibits a descending chain of subgroups $G = G_1 \geq G_2 \geq \cdots \geq G_{n+1} = 1$ in which each G_{i+1} is a normal subgroup of G_i and the quotient group G_i/G_{i+1} is non-trivial cyclic for $i = 1, \ldots, n$. Such a chain of subgroups is called a* polycyclic series*. We call $n \in \mathbb{N}$ the* length *of the series.*

Every infinite cyclic group is isomorphic to the additive group of the integers $\mathbb{Z}$. Every finite cyclic group of order n is isomorphic to the additive group $\mathbb{Z}/n\mathbb{Z}$. The polycyclic groups with a polycyclic series with a length of at most 1 are just the cyclic groups. That is why polycyclic groups are called a natural generalization of cyclic groups.

The notion *poly*-cyclic arises from a more general point of view: Suppose $\mathcal{P}$ is a property of groups. A group is called *poly-$\mathcal{P}$* if it possesses a descending chain of subgroups

$$G = G_1 \geq G_2 \geq \cdots \geq G_{n+1} = 1$$

such that each G_{i+1} is a normal subgroup of G_i and the quotient group G_i/G_{i+1} has $\mathcal{P}$.

Furthermore, a group G is called *$\mathcal{P}$-by-$\mathcal{L}$* where $\mathcal{P}$ and $\mathcal{L}$ are properties of groups if G has a normal subgroup N such that N has $\mathcal{P}$ and G/N has $\mathcal{L}$. A group is *virtually $\mathcal{P}$* if it exhibits a subgroup of finite index which has $\mathcal{P}$. Being $\mathcal{P}$-by-finite is the same as being virtually $\mathcal{P}$ as we can always construct a normal subgroup of finite index out of a subgroup $H \subset G$ of finite index via the normal closure $H^G := \{g^{-1}Hg \mid g \in G\}$.

Remark 1.2. *As every cyclic group is abelian, every polycyclic group is solvable.*

Let us therefore recapitulate the definition of solvable groups, see [HEO05, Chapter 2.3]. Sometimes solvable groups are also called *soluble*. There are several different possibilities to define solvability. The most common is the following definition as a poly-abelian group.

Definition 1.3. *A group G is called* solvable *if it exhibits a descending chain of subgroups $G = G_1 \geq G_2 \geq \cdots \geq G_{n+1} = 1$ in which each G_{i+1} is a normal subgroup of G_i and the quotient group G_i/G_{i+1} is non-trivial abelian.*

An elementary property of solvable groups is the following:

Proposition 1.4. *The class of solvable groups is closed with respect to subgroups and factor groups.*

Proof. Let G be a solvable group with abelian series $G = G_1 \trianglerighteq G_2 \trianglerighteq \cdots \trianglerighteq G_{n+1} = 1$. If H is a subgroup of G, then one can deduce by applying the Second Isomorphism Theorem

$$H \cap G_i / H \cap G_{i+1} = H \cap G_i/(H \cap G_i) \cap G_{i+1} \cong (H \cap G_i)G_{i+1}/G_{i+1} \leq G_i/G_{i+1},$$

which shows that $\{H \cap G_i \mid i = 1, \ldots, n\}$ is an abelian series of H and hence H is solvable. For $N \trianglelefteq G$ one can deduce by applying the Second Isomorphism Theorem once more

$$G_iN/G_{i+1}N = G_i(G_{i+1}N)/G_{i+1}N \cong G_i/G_i \cap (G_{i+1}N) \leq G_i/G_{i+1}.$$

Via using the Third Isomorphism Theorem $\{G_iN/N \mid i = 1, \ldots, n\}$ is an abelian series of G/N and hence G/N is solvable. □

Let us recap the definition and some properties of commutator groups to introduce a second definition of solvable groups.

Definition 1.5. *For $g, h \in G$ we define the* commutator $[g, h] := g^{-1}h^{-1}gh$. *Let H, K be subgroups of G, we define the* commutator group $[H, K] := \langle [h, k] \mid h \in H, k \in K \rangle \subset G$. *The commutator group $G' := [G, G]$ is called the* derived subgroup *of G. A group is called* perfect *if $G' = G$ and* metabelian *if G' is abelian.*

Lemma 1.6. *If H and K are normal subgroups of G, then $[H, K]$ is normal in G.*

Proof. It is sufficient to show the statement for every generator $[h, k]$ of $[H, K]$. Suppose $g \in G$, then we have to prove that it holds $g^{-1}[h, k]g \in [H, K]$. Using the fact that H, K are normal in G we get elements $h' \in H$ and $k' \in K$ such that

$$g^{-1}[h, k]g = g^{-1}h^{-1}gg^{-1}k^{-1}gg^{-1}hgg^{-1}kg = h'^{-1}k'^{-1}h'k' = [h', k'] \in [H, K].$$

□

Lemma 1.7. *For any element $g, h, k \in G$, the following holds*

$$[g, hk] = [g, k][g, h][[g, h], k].$$

Proof. On the left side we can conclude

$$[g, hk] = g^{-1}(hk)^{-1}g(hk) = g^{-1}k^{-1}h^{-1}ghk.$$

On the right side we can deduce by cancellation

$$\begin{aligned}[g, k][g, h][[g, h], k] &= (g^{-1}k^{-1}gk)(g^{-1}h^{-1}gh)((g^{-1}h^{-1}gh)^{-1}k^{-1}(g^{-1}h^{-1}gh)k) \\ &= g^{-1}k^{-1}h^{-1}ghk.\end{aligned}$$

□

Definition 1.8. *The* derived series *of a group G is the descending series*

$$G = G^{[0]} \trianglerighteq G^{[1]} \trianglerighteq \cdots \trianglerighteq G^{[i-1]} \trianglerighteq G^{[i]} \trianglerighteq \cdots$$

where $G^{[i]} = [G^{[i-1]}, G^{[i-1]}]$ for $i \geq 1$.

D. Robinson shows the following result [Rob82, Proposition 5.1.8]:

Proposition 1.9. *A group G is solvable if and only if a $r \in \mathbb{N}$ exists such that the derived series terminates at $G^{[r]} = 1$.*

The smallest r for which $G^{[r]} = 1$ is called the *derived length* of the solvable group. The solvable groups with a derived length of at most 1 are just the abelian groups. The solvable groups with a derived length of at most 2 are just the metabelian groups. Moreover, using that $G' = [G, G]$ is normal in G and Lemma 1.6, all of the terms in the derived series are normal in G.

Proposition 1.10. *The class of solvable groups is closed with respect to forming extensions.*

Proof. Let N be a normal subgroup of G and both N and G/N solvable. Hence there is an integer i such that $(G/N)^{[i]} = (G^{[i]})N/N = 1$. Thus $G^{[i]} \subset N$. There is also an integer j such that $N^{[j]} = 1$. Subsequently, $G^{[i+j]} \subset (G^{[i]})^{[j]} \subset N^{[j]} = 1$. □

Definition 1.11. *A group G satisfies the* maximal condition *if one of the following holds:*

(i) every subgroup of G is finitely generated (f.g.),

(ii) every strictly ascending chain of subgroups of G is finite,

(iii) every family of subgroups of G has a maximal member.

The notations follow D. Segals book *Polycyclic groups* [Seg83, Chapter 1.A]. For short, we say a group *has max*. Examples of groups satisfying the maximal condition are finite groups and the infinite cyclic group, so all cyclic groups. Furthermore, the class of groups satisfying the maximal condition is closed under forming subgroups, factor groups and extensions.

Proposition 1.12. *Suppose $N \trianglelefteq G$ is a normal subgroup. Then N and G/N satisfy the maximal condition if and only if G does.*

Proof. If G has max, it is clear that N and G/N also have it. Conversely, suppose $G_1 \leq G_2 \leq \cdots \leq G$ is an ascending chain of subgroups of G. If N and G/N satisfy the maximal condition, $n \in \mathbb{N}$ exists such that $G_i \cap N = G_n \cap N$ and $G_i N = G_n N$ for all $i \geq n$. Let $i \geq n$, then it follows

$$G_i = G_i \cap (G_i N) = G_i \cap (G_n N) = G_n(G_i \cap N) = G_n(G_n \cap N) = G_n,$$

where the third equality uses the modular law with $G_n \subset G_i$. So $G_i = G_n$ for all $i \geq n$, hence the ascending chain of subgroups of G is finite. □

As the property of satisfying the maximal condition is closed under forming extensions and as cyclic groups have max, every polycyclic group fulfills the maximal condition. That is why every polycyclic group itself is finitely generated. Conversely, we can show:

Lemma 1.13. *A solvable group G which satisfies the maximal condition is polycyclic.*

Proof. Suppose G is a solvable group which satisfies the maximal condition. The factors of the derived series are abelian groups and as G has max they are also f.g. By refining this series we get one with cyclic factors. $\square$

The classes of finite solvable and finite polycyclic groups are identical, as every finite group satisfies the maximal condition.

$$\begin{tikzcd}
\{ \text{ polycyclic groups } \} \arrow[r, bend left] & \{ \text{ solvable groups } \} \arrow[l, bend left, "\text{maximal condition}"]
\end{tikzcd}$$

Figure 1: Relation between polycyclic and solvable groups

Using the results we presented above, it follows immediately:

Lemma 1.14. *The class of polycyclic groups is closed with respect to forming subgroups, factor groups and extensions.*

1.2. Polycyclic sequences

We stick to the notations of D. Holt [HEO05, Chapter 8]. Let G be a polycyclic group with a polycyclic series $G = G_1 \geq G_2 \geq \cdots \geq G_{n+1} = 1$. As G_i/G_{i+1} is cyclic, elements $x_i \in G$ exist with $\langle x_i G_{i+1} \rangle = G_i/G_{i+1}$ for every index i.

Definition 1.15. *A sequence of elements $X = [x_1, \ldots, x_n]$ such that $\langle x_i G_{i+1} \rangle = G_i/G_{i+1}$ for $1 \leq i \leq n$ is called a* polycyclic sequence *of G.*

It is evident by induction that every tail-sequence $X_i = [x_i, \ldots, x_n]$ is a polycyclic sequence of the subgroup G_i and that $G_i = \langle x_i, \ldots, x_n \rangle$. Thus, the polycyclic series is uniquely determined by X. However, the polycyclic series does not uniquely determine the polycyclic sequence, see Example 1.19.

Definition 1.16. *Let X be a polycyclic sequence of G defining the polycyclic series $G = G_1 \geq G_2 \geq \cdots \geq G_{n+1} = 1$. We call the sequence $R(X) = (r_1, \ldots, r_n)$ defined by $r_i := [G_i : G_{i+1}] \in \mathbb{N} \cup \{\infty\}$ the* sequence of relative orders *of X. We also define the set $I(X) := \{i \in \{1, \ldots, n\} \mid r_i \text{ is finite } \}$.*

The sequence of relative orders $R(X)$ and the set $I(X)$ only depend on the given polycyclic series $G = G_1 \geq G_2 \geq \cdots \geq G_{n+1} = 1$. More precisely, if X and X' are two polycyclic sequences of the same polycyclic series, then $R(X) = R(X')$ and $I(X) = I(X')$ hold. The relative orders give some basic information about the group. For example, the group is finite if and only if $I(X) = \{1, \ldots, n\}$. If G is finite, then $|G| = r_1 \cdots r_n$.

Definition 1.17. *Given a polycyclic group G with polycyclic series*

$$G = G_1 \geq G_2 \geq \cdots \geq G_{n+1} = 1,$$

the number of infinite entries in the relative orders sequence $R(X)$ is called the Hirsch length*.*

Proposition 1.18. *The Hirsch length of a polycyclic group does not depend on the choice of the polycyclic series.*

Proof. Suppose we have a cyclic series $G = G_1 \trianglerighteq G_2 \trianglerighteq \cdots \trianglerighteq G_{n+1} = 1$ of G. If G_i/G_{i+1} is an infinite cyclic factor and we look at a refined term $G_i \geq G'_i > G'_{i+1} \geq G_{i+1}$, then exactly one of the factors $G_i/G'_i, G'_i/G'_{i+1}$ or G'_{i+1}/G_{i+1} is infinite cyclic. Hence a refinement of the cyclic series will have the same number of infinite factors. Since any two series of the same polycyclic group G have isomorphic refinements using Schreiers refinement theorem, they must have the same number of infinite cyclic factors and therefore have the same Hirsch length. □

As a result, we can talk about the Hirsch length of a polycyclic group and notate it as $h(G)$. G. Baumslag et al. show the existence of an algorithm which finds the Hirsch length $h(G)$ of a given polycyclic group [BCRS91, Proposition 3.5]. It is evident by the definition of the Hirsch length, that a subgroup H of G has a Hirsch length less than or equal to that of the parent group. It is the same if and only if the subgroup has finite index $[G : H]$ in G.

Example 1.19. *Let $G = \langle a, b \rangle$ be the group generated by*

$$a = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } b = \begin{pmatrix} -1 & -1 \\ 0 & 1 \end{pmatrix}.$$

G is isomorphic to D_∞ the infinite dihedral group. Furthermore, it is a subgroup of $GL(2, \mathbb{R})$. One possible polycyclic sequence of G is the sequence $X = [a, ab]$ with the series $G = G_1 \trianglerighteq G_2 = \langle ab \rangle \trianglerighteq G_3 = 1$. Another polycyclic sequence of G is the sequence $X = [b, ab]$ defining the same series. We get the relative orders $R(X) = (2, \infty)$ and the finite orders $I(X) = \{1\}$. This sequence shows the structure of G as a semidirect product of an infinite cyclic group $\langle ab \rangle$ with a cyclic group $\langle a \rangle$ of order 2. We recall the definition of a semidirect product in Chapter 4. Nevertheless, there are many others, in fact, infinitely many polycyclic sequences of G. Let $l \in \mathbb{N}$ and let $n_i \in \mathbb{N}$ with $n_i \neq 1$ for $1 \leq i \leq l$. Then $Y = [a, ab, (ab)^{n_1}, \ldots, (ab)^{n_1 \cdots n_l}]$ is a polycyclic sequence with relative orders $R(Y) = (2, n_1, \ldots, n_l, \infty)$. The number of infinite factors in the polycyclic series is in all cases the same, namely 1.

Every element of a polycyclic group can be written in a unique way given a polycyclic sequence X.

Proposition 1.20. *Let $X = [x_1, \ldots, x_n]$ be a polycyclic sequence of G with relative orders $R(X) = (r_1, \ldots, r_n)$. Then every $g \in G$ can be written in a unique* normal form *with $e_i \in \mathbb{Z}$ for $1 \leq i \leq n$ and $0 \leq e_i < r_i$ if $i \in I(X)$, such that*

$$g = x_1^{e_1} \cdots x_n^{e_n}.$$

Proof. Let g be an element of G. We prove the proposition by induction on the length of X. For a polycyclic sequence of length 1 the group itself is cyclic, hence every element clearly has a normal form. For induction we assume that every element of a group with a polycyclic sequence of length $n-1$, for instance every element in $G_2 = \langle x_2, \dots x_n \rangle$ has a unique normal form. Furthermore, it holds $\langle x_1 G_2 \rangle = G_1/G_2$. Hence we can find $e_1 \in \mathbb{Z}$ such that $gG_2 = x_1^{e_1} G_2$. If $r_1 \neq \infty$ we can choose $e_1 \in \{0, \dots, r_1 - 1\}$ which makes this expression unique. Let $h = x_2^{e_2} \cdots x_n^{e_n}$ be the unique normal form of $h = x_1^{-e_1} g \in G_2$. Finally, we get the unique expression $g = x_1^{e_1} \cdots x_n^{e_n}$. $\square$

1.3. Polycyclic presentations

In this section we will show that every polycyclic group exhibits a special finite presentation. *"Unlike a general group presentation, a polycyclic presentation allows efficient structural computation within the group."* [HEO05, p. 273] This special presentation stores the polycyclic structure of the group it represents. The definition still follows the notations of D. Holt [HEO05, Chapter 8].

Definition 1.21. *A presentation $\langle x_1, \dots, x_n | R \rangle$ is called* a polycyclic presentation *if a sequence $S = (s_1, \dots, s_n)$ with $s_i \in \mathbb{N} \cup \{\infty\}$ and integers $a_{i,k}, b_{i,j,k}, c_{i,j,k}$ exists such that R is given by the following relations:*

$$x_i^{s_i} = R_{i,i} \text{ with } R_{i,i} = x_{i+1}^{a_{i,i+1}} \cdots x_n^{a_{i,n}} \text{ for } 1 \leq i \leq n \text{ with } s_i < \infty,$$

$$x_i^{x_j} := x_j^{-1} x_i x_j = R_{i,j} \text{ with } R_{i,j} = x_{j+1}^{b_{i,j,j+1}} \cdots x_n^{b_{i,j,n}} \text{ for } 1 \leq j < i \leq n,$$

$$x_i^{x_j^{-1}} := x_j x_i x_j^{-1} = R'_{i,j} \text{ with } R'_{i,j} = x_{j+1}^{c_{i,j,j+1}} \cdots x_n^{c_{i,j,n}} \text{ for } 1 \leq j < i \leq n.$$

$R_{i,j}$ and $R'_{i,j}$ are words in the generators. The first type are the *power relations* and the second and third ones are the *conjugate relations*. S is called the *sequence of power exponents* of the presentation.

Conjugate relations of the form $x_j^{-1} x_i x_j = x_i$ and $x_j x_i x_j^{-1} = x_i$ are called *trivial polycyclic relations* and they are often omitted. To distinguish general group presentations from polycyclic ones, we write $PC \langle x_1, \dots, x_n | R \rangle$.

If a polycyclic presentation has n generators and k finite entries in S, then R consists of $k + n(n-1) = O(n^2)$ relations. This set of relations is usually highly redundant as a set of defining relations, but it is useful for computational purposes. The example of a polycyclic presentation with one single generator splits in two cases: If $R = \emptyset$, we have the infinite cyclic group. If R contains a single power relation of the form $x_1^{s_1} = 1$, we get a finite cyclic group of order s_1.

Theorem 1.22. *Every polycyclic sequence X of a polycyclic group G determines a polycyclic presentation. The sequence S equals the relative orders $R(X)$. Thus, every polycyclic group can be defined by a polycyclic presentation.*

Proof. As $r_i = [G_i : G_{i+1}]$, we have $x_i^{r_i} \in G_{i+1}$ and hence, the normal form for this power has the required form. Since $j < i$ we have $x_i \in G_{j+1}$. As G_{j+1} is normal in G_j, it yields that the conjugates $x_j x_i x_j^{-1}$ and $x_j^{-1} x_i x_j$ are contained in G_{j+1}. Consequently, their normal forms are of the required form. □

Vice versa, every polycyclic presentation defines a polycyclic group:

Theorem 1.23. *Let $PC\langle x_1, \ldots, x_n | R\rangle$ be a polycyclic presentation with S the set of power exponents and let G be the group defined by this presentation. Then G is polycyclic and $X = [x_1, \ldots, x_n]$ is a polycyclic sequence of G. Its sequence of relative orders $R(X) = (r_1, \ldots, r_n)$ fulfills $r_i \leq s_i$.*

Proof. Suppose $G_i = \langle x_i, \ldots, x_n\rangle \leq G$. Then the conjugate relations of R induce that G_{i+1} is normal in G_i. By construction G_i/G_{i+1} is cyclic and hence G is polycyclic. As $G_i/G_{i+1} = \langle x_i G_{i+1}\rangle$ by definition, it follows that X is a polycyclic sequence of G. Finally, the power relations induce that $r_i = [G_i : G_{i+1}] \leq s_i$. □

Example 1.24. *We consider the group G given by the following polycyclic presentation with $S = (\infty, \infty)$:*

$$G = PC\left\langle x_1, x_2 \mid x_2^{x_1} = x_2^3, x_2^{x_1^{-1}} = x_2^4 \right\rangle.$$

We can transform $x_2 x_1 x_1^{-1}$ in two different ways. Firstly, $x_2 x_1 x_1^{-1} = x_2$ and secondly

$$x_2 x_1 x_1^{-1} = x_1 x_2^3 x_1^{-1} = x_1 x_2^2 x_1^{-1} x_2^4 = x_1 x_2 x_1^{-1} x_2^8 = x_1 x_1^{-1} x_2^{12} = x_2^{12}.$$

As a result $X = [x_1, x_2]$ is a polycyclic sequence of G with the sequence of relative orders $R(X) = (\infty, 11)$ and $r_2 \neq s_2$.

Example 1.25. *We consider the group G given by the following polycyclic presentation with $S = (3, 3, \infty)$:*

$$G = PC\left\langle x_1, x_2, x_3 \mid x_1^3 = x_3, x_2^3 = x_3, x_2^{x_1} = x_2 x_3, x_2^{x_1^{-1}} = x_2 x_3 \right\rangle.$$

As we stated above, the trivial relations

$$x_1 x_3 x_1^{-1} = x_1^{-1} x_3 x_1 = x_2 x_3 x_2^{-1} = x_2^{-1} x_3 x_2 = x_3$$

have been omitted in the notation, but do also hold in G. We can deduce that $X = [x_1, x_2, x_3]$ is a polycyclic sequence of G with relative orders $R(X) \leq (3, 3, \infty)$. It has to hold that

$$(x_1 x_2 x_1^{-1})^3 = x_1 x_2^3 x_1^{-1} = x_1 x_3 x_1^{-1} = x_3$$

is equal to

$$(x_2 x_3)^3 = x_2^3 x_3^3 = x_3^4,$$

as x_2 and x_3 commute. Hence, $x_3^3 = 1$ and therefore $R(X) = (3, 3, 3)$ with $r_3 \neq s_3$.

Definition 1.26. *A polycyclic presentation with power exponents S is called* consistent *or* confluent *if $R(X) = S$.*

In the following, we have a look at a different approach to the property consistent, G. Baumslag et al. [BCRS91] also called them *honest* polycyclic presentations. Let $PC := PC\langle x_1, \ldots, x_n | R\rangle$ be a polycyclic presentation. For $1 \leq i \leq n$ we define the polycyclic presentation $PC_i := PC\langle x_i \ldots, x_n | R_i\rangle$ by deleting the generators $x_1, \ldots, x_{i-1}$ and all relations containing these generators. Then every presentation PC_i is still a polycyclic presentation.

Let G and H_i be the groups defined by PC and PC_i. Like in the previous sections, we set $G_i = \langle x_i, \ldots, x_n\rangle$ such that $G = G_1 \trianglerighteq G_2 \trianglerighteq \cdots \trianglerighteq G_{n+1} = 1$ is a polycyclic series of G.

We get homomorphisms $H_{i+1} \to H_i$ by mapping every generator of H_{i+1} to itself in H_i and epimorphisms $H_i \to G_i$ by mapping every generator of H_i to itself in G_i. The last ones are surjective as each generator of G_i lies in the image of the homomorphism. Clearly, the following diagram commutes:

$$\begin{tikzcd}
1 = H_{n+1} \arrow[r] \arrow[d, "\cong"] & H_n \arrow[r] \arrow[d] & \cdots \arrow[r] & H_2 \arrow[r] \arrow[d] & H_1 = G \arrow[d, "\cong"] \\
1 = G_{n+1} \arrow[r, hook] & G_n \arrow[r, hook] & \cdots \arrow[r, hook] & G_2 \arrow[r, hook] & G_1 = G
\end{tikzcd}$$

It holds $H_1 = G_1$ as nothing has been omitted in the polycyclic presentation. A polycyclic presentation is consistent if H_i and G_i are isomorphic for all i. Evidently this equals the definition above which means that the set of power exponents S and the sequence of relative orders $R(X)$ have to be the same. The morphisms $H_i \to G_i$ are isomorphisms if and only if all homomorphisms $H_{i+1} \to H_i$ are injective.

Theorem 1.27. *There exists an algorithm which can decide if a given polycyclic presentation $PC\langle x_1, \ldots, x_n | R\rangle$ is consistent.*

Proof. We prove the Theorem by induction on the number of generators n. For the base case assume $n = 1$, where every polycyclic presentation is consistent as we only have the trivial group $1 = G_2 = H_2$ and the main group $G = G_1 = H_1$ which is cyclic.

We now assume that a polycyclic presentation PC with n generators is given. By induction we can decide, using the notation from above, if the presentation PC_2 with $n - 1$ generators is a consistent polycyclic presentation. If not, PC itself cannot be consistent. So we suppose that PC_2 is consistent. We have to decide whether PC is also consistent. This is the case if and only if the mappings

$$\phi\colon \begin{cases} H_2 \to H_2 \\ x_i \mapsto R_{i,1} \end{cases} \quad \text{and } \psi\colon \begin{cases} H_2 \to H_2 \\ x_i \mapsto R'_{i,1} \end{cases}$$

determine automorphisms of H_2 and the s_1-th power of the automorphisms just being the conjugation in H_2 by $R_{1,1}$ if $s_1 \neq \infty$. The $R_{i,1}$ and $R'_{i,1}$ arise from the conjugate relations of Definition 1.21, whereas $x_1^{-1}x_ix_1 = R_{i,1}$ and $x_1x_ix_1^{-1} = R'_{i,1}$ and the $R_{1,1}$ comes from the power relation with $x_1^{s_1} = R_{1,1}$.

With the help of the computable word problem for polycyclic groups, see Section 6.1 for further details and in particular Theorem 6.1, we can check if the mappings send the generators to itself and if all relations are sent to the identity which allows us to decide if the mappings define homomorphisms. Using the solution of the generalized word problem we can decide for every generator of H_2 if it lies in the image of $\langle x_1^{-1} x_i x_1 \mid i = 2, \dots, n \rangle$ and hence we are able to decide if these homomorphisms are surjective.

A. Mal'cev showed that all f.g. residually finite groups are hopfian, i.e., groups for which every surjective endomorphism is already an automorphism [Mal40]. Later on we will prove in Theorem 3.2 that polycyclic groups are residually finite and hence hopfian. Therefore, we already know that those maps define automorphisms if they are surjective. Finally, we use the solution of the word problem once more to check if the s_1-th power of the automorphisms are just the conjugation in H_2 by $R_{1,1}$. □

With the help of GAP (Groups, Algorithms, Programming) [Gro15], a system for computational discrete algebra with particular emphasis on computational group theory, it is possible to check if a given polycyclic presentation is confluent. The implementations are part of the *polycyclic* package by B. Eick et al. [ENH04].

Example 1.28. *Let G be the group given in Example 1.25. As we have $r_3 = 3$ but $s_3 = \infty$, the presentation is not consistent. We define the following maps with H_2 given by $H_2 = PC \langle x_2, x_3 \mid x_2^3 = x_3 \rangle$ as*

$$\phi\colon \begin{cases} H_2 \to H_2 \\ x_2 \mapsto R_{2,1} = x_2 x_3 \\ x_3 \mapsto R_{3,1} = x_3 \end{cases} \quad \text{and } \psi\colon \begin{cases} H_2 \to H_2 \\ x_2 \mapsto R'_{2,1} = x_2 x_3 \\ x_3 \mapsto R'_{3,1} = x_3 \end{cases} .$$

Both maps are automorphisms if they respect the relations from H_2, more precisely if $\phi(x_2^3) \overset{!}{=} \phi(x_3)$ and $\psi(x_2^3) \overset{!}{=} \psi(x_3)$. As x_3 and x_2 commute we can deduce that

$$\phi(x_2^3) = \psi(x_2^3) = (x_2 x_3)^3 = x_2^3 x_3^3 = x_3^4,$$

which is in general not $x_3 = \phi(x_3) = \psi(x_3)$. Additionally, as $s_1 = 3$ and $R_{1,1} = x_3$ we have to check if $\phi^3(x_2) = x_2 x_3^3$ equals $x_3^{-1} x_2 x_3 = x_2$. This is only the case if $x_3^3 = 1$ which is not true in general. A consistent polycyclic presentation of G would be

$$G = PC \left\langle x_1, x_2, x_3 \mid x_1^3 = x_3, x_2^3 = x_3, x_3^3 = 1, x_2^{x_1} = x_2 x_3, x_2^{x_1^{-1}} = x_2 x_3 \right\rangle .$$

Theorem 1.29. *There is an algorithm which finds a consistent polycyclic presentation of a given polycyclic group G.*

Proof. With the help of Tietze transformations we can transform every given finite presentation of a group into another finite presentation of this group, proven by W. Magnus et al. [MKS66, Corollary 1.5]. We can list all finite presentations and with the help of Theorem 1.27 check if the given polycyclic presentation is consistent. Hence we get a consistent polycyclic presentation. □

1.4. Polycyclic groups as matrix groups

In this subsection we shortly present some results about polycyclic groups and their linear aspects. D. Segal shows the link between polycyclic groups and linear groups over $\mathbb{Z}$ [Seg83, Chapter 2], more precisely that every solvable subgroup of $GL(n, \mathbb{Z})$ is polycyclic. Conversely, he also shows that every polycyclic group is isomorphic to a subgroup of $GL(n, \mathbb{Z})$ for some $n \in N$ [Seg83, Chapter 5]. This result was conjectured by P. Hall in the 1950s and proven a decade later by L. Auslander.

Furthermore, given a rational polycyclic matrix group G by a generating set, a practical algorithm to determine a polycyclic sequence of G is presented by G. Ostheimer [Ost99]. Practical means, that the algorithm is not proven to be polynomial-time in theory, but in practice it works quite good. She also introduces an algorithm for deciding if a given rational matrix group G is solvable or not, which is equivalent to decide whether or not such a group is polycyclic. It is worth noting, that algorithms to solve these problems have already been developed by G. Baumslag et al. [BCRS91], but they are not practical in general.

2. Examples of polycyclic groups

After having introduced polycyclic groups and the different ways of representing them, we will now list some examples of polycyclic groups which are relevant for the rest of this thesis. Beside supersolvable groups, we will also have a look at f.g. nilpotent and poly-infinite cyclic ones.

2.1. Supersolvable groups

Let us recapitulate the definition of supersolvable groups.

Definition 2.1. *A group G is called* supersolvable *if it exhibits a descending chain of subgroups $G = G_1 \geq G_2 \geq \cdots \geq G_{n+1} = 1$ in which each G_i is a normal subgroup of G and each quotient group G_i/G_{i+1} is non-trivial cyclic.*

Obviously, supersolvable groups are polycyclic and we can deduce from the fact that every G_i is normal in G that they feature a special polycyclic presentation.

Lemma 2.2. *A supersolvable group has a* supersolvable polycyclic presentation *in which the conjugate relations have the following form:*

$$x_j^{-1} x_i x_j = R_{i,j} = x_i^{b_{i,j,i}} \cdots x_n^{b_{i,j,n}},$$

$$x_j x_i x_j^{-1} = R'_{i,j} = x_i^{c_{i,j,i}} \cdots x_n^{c_{i,j,n}}.$$

More precisely, the right-hand sides of the conjugate relations have the depth i. In an arbitrary polycyclic presentation these relations can have a smaller depth $j+1$ with $j+1 \leq i$. In contrast to the abelian case above, a supersolvable group can also possess polycyclic presentations that are not of the supersolvable form.

Example 2.3. *The symmetric group S_3 has a supersolvable polycyclic presentation $S_3 = PC\langle x_1, x_2 \mid x_1^2 = 1, x_2^3 = 1, x_2^{x_1} = x_2^2 \rangle$ with relative orders $R(X) = (2,3)$.*

2.2. Finitely generated nilpotent groups

Let us recapitulate the definition of nilpotent groups via the lower central series.

Definition 2.4. *The* lower central series *of a group G is the descending series*

$$G = \gamma_1(G) \trianglerighteq \gamma_2(G) \trianglerighteq \cdots \trianglerighteq \gamma_i(G) \trianglerighteq \gamma_{i+1}(G) \trianglerighteq \cdots$$

in which each term is defined as $\gamma_{i+1}(G) = [G, \gamma_i(G)]$. The series terminates at $\gamma_r(G)$ and has length $r-1$ if $\gamma_r(G) = [G, \gamma_r(G)]$.

Definition 2.5. *A group G is said to be* nilpotent *if its lower central series terminates at $\gamma_r(G) = 1$.*

With Lemma 1.6 it follows that all terms in the lower central series are normal in G.

Lemma 2.6. *Every f.g. nilpotent group is polycyclic.*

Proof. Suppose G is a f.g. nilpotent group and $\gamma_i(G)$ the factors of the lower central series. It is easy to see that all quotients $\gamma_i(G)/\gamma_{i+1}(G)$ are abelian, so the group is solvable. By refining this series we get one with cyclic factors. □

We take the refinement of the lower central series as polycyclic series. Using the fact that every term $\gamma_i(G)$ is normal in G and that it yields $[G, \gamma_i(G)] = \gamma_{i+1}(G)$, more precisely $x_j^{-1}x_ix_j \in x_i\gamma_{i+1}(G)$ for every $1 \leq j < i \leq n$, we can deduce that every f.g. nilpotent group exhibits a special presentation.

Lemma 2.7. *A f.g. nilpotent group has a* nilpotent polycyclic presentation *in which the conjugate relations have the following form:*

$$\begin{aligned} x_j^{-1}x_ix_j &= R_{i,j} = x_ix_{i+1}^{b_{i,j,i+1}} \cdots x_n^{b_{i,j,n}}, \\ x_jx_ix_j^{-1} &= R'_{i,j} = x_ix_{i+1}^{c_{i,j,i+1}} \cdots x_n^{c_{i,j,n}}. \end{aligned}$$

More precisely, the right-hand sides of the conjugate relations have the depth i and leading exponent 1. In an arbitrary polycyclic presentation those relations can have a smaller depth $j+1 \leq i$ and an arbitrary leading exponent. In contrast to the abelian case, a f.g. nilpotent group can also have polycyclic presentations that are not of the nilpotent form. If the given polycyclic presentation is not nilpotent, we can compute the lower central series of G and refine it to obtain a polycyclic presentation of G which is nilpotent.

Example 2.8. *The following group is given in a nilpotent polycyclic presentation:*

$$G = PC\left\langle x_1, x_2, x_3, x_4, x_5 \mid x_3^{x_1} = x_3x_5, x_3^{x_1^{-1}} = x_3x_5^{-1}, x_4^{x_2} = x_4x_5, x_4^{x_2^{-1}} = x_4x_5^{-1} \right\rangle$$

with relative orders $R(X) = (\infty, \infty, \infty, \infty, \infty)$.

The class of f.g. nilpotent groups is a subclass of supersolvable groups as each factor of the refined lower central series $\gamma_i(G)$ is normal in G. The symmetric group S_3 has its commutator group A_3 as normal cyclic subgroup and their quotient S_3/A_3 is also cyclic. As a result, S_3 is supersolvable. Nevertheless, the commutator group $[S_3, A_3]$ is isomorphic to A_3 so the lower central series does not terminate at 1 but at A_3 and hence the given group is not nilpotent.

Clearly, every supersolvable group is polycyclic. However, there are polycyclic groups which are not supersolvable. The alternating group A_4 has order 12 with the derived subgroup isomorphic to the Klein four-group V_4 which is abelian. Hence, A_4 is finite solvable and thus finite polycyclic, but it is not supersolvable as it has no cyclic normal subgroup. That is why it is impossible to construct a normal series where all successive factors are cyclic. We get the following picture of inclusions:

$$\{\text{ f.g. nilpotent }\} \subsetneq \{\text{ supersolvable }\} \subsetneq \{\text{ polycyclic }\}$$

Figure 2: Relation between f.g. nilpotent, supersolvable and polycyclic groups

2.3. Poly-infinite cyclic groups

We already used the prefix *poly* for *poly*-abelian which means solvable and *poly*-cyclic which means polycyclic. Now we consider *poly-infinite cyclic groups*, i.e., groups which exhibit a normal series of infinite cyclic factors. Clearly, every poly-infinite cyclic group is polycyclic and torsion-free. The trivial group 1 is by convention poly-infinite cyclic of length 0. Furthermore, there are interesting links between general polycyclic groups and poly-infinite cyclic ones as pointed out by D. Robinson [Rob82, Chapter 5]. As the following result will be needed for the proof of Theorem 3.2, we state its detailed proof here.

Theorem 2.9.

(a) Every polycyclic group has a normal poly-infinite cyclic subgroup of finite index.

(b) Every infinite polycyclic group contains a non-trivial torsion-free abelian normal subgroup.

In order to proof this result we need the following lemma.

Lemma 2.10. *A f.g. solvable torsion group is finite.*

Proof. We prove this lemma by induction on the derived length d of a f.g. solvable torsion group G. If $d = 0$, then $G = G^{[0]} = 1$ is obviously finite. So we assume $d > 0$ and set $A := G^{[d-1]}$. By induction we suppose that G/A is finite and as G is f.g., A has also to be f.g. It yields $[A, A] = [G^{[d-1]}, G^{[d-1]}] = G^{[d]} = 1$, hence A is abelian and a f.g. torsion group, thus finite. Therefore, also G has to be finite. □

Proof of Theorem 2.9. (a) Let $G = G_1 \geq G_2 \geq \cdots \geq G_{n+1} = 1$ be a normal series with cyclic factors. We prove the theorem by induction on the length of this series. For the base case assume $n = 1$, hence G is cyclic. If G is infinite cyclic, the group G itself is a normal poly-infinite cyclic subgroup of index 1. If G is finite cyclic, the trivial group 1 is a normal poly-infinite cyclic subgroup of finite index $|G|$. For the induction step we assume that G is given in a normal series with cyclic factors of length $n > 1$. Consider the group $N := G_2$ which is a polycyclic group with a cyclic series of length $n-1$. Thus, by induction a normal subgroup M of N exists such that M is poly-infinite cyclic and N/M is finite. We define the *normal core* as

$$M_G := \bigcap_{g \in G} g^{-1} M g,$$

which is the largest subgroup of M that is normal in G. Additionally, it is the kernel of the homomorphism

$$\phi\colon N \to \bigoplus_{g \in G} N/M^g.$$

As $M \trianglelefteq N$ and $N \trianglelefteq G$ it holds that $M^g \trianglelefteq N$: let $g \in G$, $n \in N$ and $m \in M$, then

$$\begin{aligned} n^{-1}(g^{-1}mg)n &= (g^{-1}g)n^{-1}(g^{-1}mg)n(g^{-1}g) \\ &= g^{-1}(gn^{-1}g^{-1})m(gng^{-1})g \\ &= g^{-1}(\tilde{n}^{-1}m\tilde{n})g \\ &= g^{-1}\tilde{m}g \in M^g. \end{aligned}$$

This induces that N/M^g is finite and hence is the image of this homomorphism a torsion group. Using the First Isomorphism Theorem it follows also that N/M_G is a torsion group. Following Lemma 2.10 this f.g. solvable torsion group has to be finite. Thus, we can assume without loss of generality that $M \trianglelefteq G$.

If G/N is finite, so is G/M and we are done. So we assume that G/N is infinite cyclic, generated by xN. Using that N is normal in G and N/M is finite, there is a positive integer r for which x^r centralizes N/M, i.e.,

$$x^r \in C_G(N/M) = \{g \in G \mid [g,n] \in M \text{ for all } n \in N\}.$$

Define $L := \langle x^r, M\rangle$. Then it holds that $L \trianglelefteq \langle x, N\rangle = G$ as $[x^r, N] \leq M$. Furthermore, G/L is finite because it is the product of the finite subgroups $\langle x, L\rangle /L$ and NL/L. Since no positive power of x can belong to N, the factor L/M is infinite cyclic and thus L is poly-infinite cyclic.

(b) Let G be infinite, then L is non-trivial and as a subgroup of G it is also solvable. The smallest non-trivial term $H = L^{[d-1]}$ of the derived series of L is abelian as the derived group $[H,H] = L^{[d]} = 1$ is trivial and it is also torsion-free as L is poly-infinite cyclic. As a factor of the derived series of L the subgroup H is characteristic in L and as L is normal in G, H is also normal in G. □

G. Baumslag et al. state an algorithm to construct a poly-infinite cyclic normal subgroup of finite index [BCRS91, Proposition 3.10]. We refer the interested reader to the original paper for further details.

3. Polycyclic-by-finite groups

As a generalization of polycyclic groups one can study *polycyclic-by-finite groups*, i.e., groups which have a normal polycyclic subgroup of finite index. Many results for polycyclic groups, for instance the property of being residually finite, are originally given for the more general class of polycyclic-by-finite groups. We will shortly introduce them and state the residually finiteness which is crucial for the rest of this work.

Polycyclic-by-finite groups can equivalently be characterized as groups which exhibit a normal series of finite length whose infinite factors are cyclic. As the property of fulfilling the maximal condition is closed under forming extensions and as all finite and cyclic groups have max, every polycyclic-by-finite group still fulfills the maximal condition.

If G is a polycyclic-by-finite group, then the Hirsch length of G is defined as the Hirsch length of a polycyclic normal subgroup H of G, where H has finite index in G. This is independent of the choice of the subgroup, as all such subgroups will have the same Hirsch length using the same reasoning as in Proposition 1.18. If $N \trianglelefteq G$ is a normal subgroup, then it is evident that $h(G) = h(G/N) + h(N)$, using the factors $\{G_iN/N\}$ and $\{G_i \cap N\}$ of the corresponding series. For a subgroup $H \leq G$ it holds

$$h(G) = h(H) \Leftrightarrow |G : H| < \infty.$$

In general, polycyclic-by-finite groups are not solvable but still f.g. and residually finite as proven by K.A. Hirsch [Hir52]. Let us therefore recap the definition of residually finiteness. We use the notations of D. Segal [Seg83, Chapter 1].

Definition 3.1. *A group G is called* residually finite *if*

$$\bigcap \{N \mid N \trianglelefteq G \textit{ and } G/N \textit{ is finite}\} = 1.$$

Theorem 3.2. *Suppose G is a polycyclic-by-finite group. Then G is residually finite.*

Proof. We show the statement by induction on the Hirsch length $h(G)$ of G. For the base case assume $h(G) = 0$, hence G is a finite group and therefore residually finite as 1 is a normal subgroup of finite index. For the induction step we assume that G has Hirsch length $h(G) > 0$ and every polycyclic-by-finite group with smaller Hirsch length is residually finite. G is infinite as $h(G) \neq 0$ and using Theorem 2.9 we get a non-trivial torsion-free, thus an infinite free, abelian normal subgroup A. For $m \in \mathbb{N}$ the group $A^m = \langle a^m \mid a \in A\rangle$ is normal in G: let a^m be in A^m and g be in G, then $g^{-1}a^mg = (g^{-1}ag)^m \in A^m$ as $(g^{-1}ag) \in A$. Using the fact that the index of A^m in A is m^d with $A \cong \mathbb{Z}^d$, thus finite, and that A itself is infinite, hence $h(A) > 0$, we can deduce

$$h(G/A^m) = h(G) - h(A^m) = h(G) - h(A) < h(G),$$

hence G/A^m is residually finite by the induction hypothesis, more precisely

$$\bigcap \{N/A^m \mid N/A^m \trianglelefteq G/A^m \text{ and } G/N \text{ is finite}\} = 1 = A^m/A^m.$$

This is equivalent to

$$\bigcap \{N \mid A^m \leq N \trianglelefteq G \text{ and } G/N \text{ is finite}\} = A^m.$$

If we take the intersection over all positive integers m, we can deduce

$$\bigcap \{N \mid N \trianglelefteq G \text{ and } G/N \text{ is finite}\} \leq \bigcap_m A^m = 1.$$

This last equality becomes obvious if we represent the free abelian group A as the direct sum $A = \mathbb{Z} \oplus \ \ldots \ \oplus \ \mathbb{Z}$, hence $A^m = m\mathbb{Z} \oplus \cdots \oplus m\mathbb{Z}$ and using

$$\bigcap_m m\mathbb{Z} = 1.$$

□

Lemma 3.3. *A group G is residually finite if and only if*

$$x = y \quad \Leftarrow \quad xN = yN \textit{ for all } N \trianglelefteq G \textit{ and } G/N \textit{ is finite.}$$

Proof. Suppose G is residually finite. Set $\tilde{N} = \bigcap \{N \mid N \trianglelefteq G \text{ and } G/N \text{ is finite}\} = 1$. We can deduce from $x\tilde{N} = y\tilde{N}$ that $x = y$. Conversely, suppose that $x \in N$ and hence $xN = N$ for all $N \trianglelefteq G$ with G/N finite. It follows that $x = 1$ in G and hence that

$$\bigcap \{N \mid N \trianglelefteq G \text{ and } G/N \text{ is finite}\} = 1.$$

□

4. A prototype of polycyclic groups

D. Holt et al. [HEO05, Chapter 8] present a prototype of polycyclic groups and its construction. This prototype is often used for experimental purposes as it is easy to implement and its Hirsch length has a lower bound. Additionally, a special attack against this type of polycyclic groups, called the Field-Based Attack, exists which is presented later in Section 8.2. That is why we introduce this class of polycyclic groups in detail in the following.

Recall that an *algebraic number field* F is an extension of $\mathbb{Q}$ of finite degree $[F : \mathbb{Q}]$. The *maximal order*, also called the *ring of integers* of F is defined as

$$\mathcal{O}_F := \{a \in F \mid a \text{ is a zero of a monic polynomial } f_a(x) \in \mathbb{Z}[x]\} .$$

The *unit group* of F is defined as

$$U_F := \left\{a \in \mathcal{O}_F \mid a \neq 0 \text{ and } a^{-1} \in \mathcal{O}_F\right\} .$$

Furthermore, $\mathcal{O}_F$ forms a ring whose additive group is isomorphic to $\mathbb{Z}^n$ with n being the degree $[F : \mathbb{Q}]$. Following Dirichlet's unit theorem, U_F is a f.g. abelian group of the form $U_F \cong \mathbb{Z}/k\mathbb{Z} \times \mathbb{Z}^{s+t-1}$, where $n = s + 2t$ with s being the number of all real embeddings $F \to \mathbb{R}$ and t being the number of conjugate pairs of all complex embeddings $F \to \mathbb{C}$ and $k \in 2\mathbb{N}$.

Before we continue we recapitulate the definition of a semidirect product.

Definition 4.1. *Given two groups H, N and a group homomorphism $\alpha\colon H \to \mathrm{Aut}(N)$. The* semidirect product G of N and H with respect to α *is the set of all pairs (h, n) with $h \in H$ and $n \in N$, endowed with the group operation $\circ$ given by*

$$(h_1, n_1) \circ_G (h_2, n_2) = (h_1 \circ_H h_2, \alpha(h_1)(n_1) \circ_N n_2).$$

We denote it as $G = H \ltimes_\alpha N$.

Sometimes, this group is also called *external* semidirect product to distinguish it from the special case of an *inner* semidirect product. For more details on inner and outer semidirect products see [Rob82, Section 1.5]. As we outlined in Example 1.19, the infinite dihedral group D_∞ is a semidirect product of an infinite cyclic group with a cyclic group of order 2.

The unit group U_F acts by multiplication from the right on the additive group of the maximal order $\mathcal{O}_F$. Thus, we can define the homomorphism $\alpha\colon U_F \to \mathrm{Aut}(\mathcal{O}_F)$ by

$$U_F \ni u \mapsto (\mathcal{O}_F \ni o \mapsto o \cdot u)$$

and we can define the semidirect product $G_F = U_F \ltimes_\alpha \mathcal{O}_F$. In the following, we simply denote it by $G_F = U_F \ltimes \mathcal{O}_F$.

As the class of polycyclic groups is closed under taking extensions, see Lemma 1.14, the group G_F is an infinite polycyclic group of Hirsch length $\geq n = h(\mathcal{O}_F)$ with $n = [F : \mathbb{Q}]$. Clearly, this group is metabelian, i.e., a solvable group of derived length at most 2, as $\mathcal{O}_F$ and $G_F/\mathcal{O}_F \cong U_F$ are abelian. We can check that it is non-virtually nilpotent with the method presented later on in Section 7.3.

There are two different ways to represent G_F. Firstly, one can represent its elements as tuples of the semidirect product with the induced binary operation. Therefore, we represent G_F as a set of pairs of matrices. Secondly, one can construct a polycyclic presentation for G_F and work with the elements as words over a finite generating set. With $\mathbf{1}$ and $\mathbf{0}$ we denote the identity matrix and the zero matrix.

4.1. Matrix presentation

The notations follow M. Kotov and A. Ushakov [KU15]. Suppose $F = \mathbb{Q}[x]/(f)$ with $f(x) \in \mathbb{Z}[x]$ an irreducible monic polynomial with $n = \deg(f)$ is an algebraic number field. The *companion matrix* for $f = x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0$ with $c_i \in \mathbb{Z}$ for $0 \leq i \leq n$ is the matrix of the form

$$M = \begin{pmatrix} 0 & 0 & \cdots & 0 & -c_0 \\ 1 & 0 & \cdots & 0 & -c_1 \\ 0 & 1 & \cdots & 0 & -c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -c_{n-1} \end{pmatrix}.$$

The characteristic polynomial as well as the minimal polynomial of M are equal to f, see [HJ13, Chapter 3.3] for further details. With e_i we denote the i-th standard basis vector. One can observe that

$$\mathbf{1} \cdot e_1 = e_1 = M^0 \cdot e_1 \text{ and } M \cdot e_1 = e_2 = M^1 \cdot e_1,$$

and according to this

$$M \cdot e_i = e_{i+1} = M^i \cdot e_1 \text{ for } i = 1, \dots n-1.$$

Thus, the field F alternatively can be by presented as

$$F = \left\{ a_0\mathbf{1} + a_1 M + a_2 M^2 + \cdots + a_{n-1}M^{n-1} \mid a_0, \dots, a_{n-1} \in \mathbb{Q} \right\} \tag{4.1}$$

with the usual matrix addition and multiplication. Let $\{O_1, \dots O_n\}$ be a basis of the ring $\mathcal{O}_F$ and $\{U_1, \dots U_m\}$ be a generating set for the group U_F, where each O_i and each U_i is a matrix. It yields

$$\mathcal{O}_F = \{a_1 O_1 + \cdots + a_n O_n \mid a_1, \dots, a_n \in \mathbb{Z}\} \text{ and}$$

$$U_F = \{U_1^{a_1} \cdot \cdots \cdot U_m^{a_m} \mid a_1, \dots, a_m \in \mathbb{Z}\},$$

where $U_F \cong \mathbb{Z}/k\mathbb{Z}\times\mathbb{Z}^{m-1}$ with m and k resulting from Dirichlet's unit theorem. Without loss of generality we can assume $U_1^k = \mathbf{1}$. The group $G_F = U_F \ltimes \mathcal{O}_F$ is a set of pairs of matrices

$$G = \{(C,S) \mid C \in U_F, S \in \mathcal{O}_F\},$$

equipped with multiplication given by $(C,S)\cdot(D,T) = (CD, SD+T)$. The inverse in G_F can be computed as $(C,S)^{-1} := (C^{-1}, -SC^{-1})$, as it yields

$$(C,S)\cdot(C,S)^{-1} = (C,S)\cdot(C^{-1},-SC^{-1}) = (CC^{-1}, SC^{-1}-SC^{-1}) = (\mathbf{1},\mathbf{0}).$$

Using the commutativity of U_F it follows that

$$(D,T)^{(C,S)} = (C,S)^{-1}\cdot(D,T)\cdot(C,S) = (D, S(\mathbf{1}-D)+TC). \tag{4.2}$$

4.2. Polycyclic presentation

Using the considerations above, one can determine a polycyclic presentation for G_F with $G_F = PC\,\langle X|R\rangle$ in the generators

$$X = \{x_1,\dots,x_m,x_{m+1},\dots,x_{m+n}\},$$

where $x_1,\dots,x_m$ correspond to the pairs $(U_1,\mathbf{0}),\dots,(U_m,\mathbf{0})$ and $x_{m+1},\dots,x_{m+n}$ correspond to the pairs $(\mathbf{1},O_1),\dots,(\mathbf{1},O_n)$. Using Equation 4.2 yields

$$(U_j,\mathbf{0})^{-1}(\mathbf{1},O_i)(U_j,\mathbf{0}) = (\mathbf{1},O_iU_j) \text{ and } (U_j,\mathbf{0})(\mathbf{1},O_i)(U_j,\mathbf{0})^{-1} = (\mathbf{1},O_iU_j^{-1}).$$

The set of relations R of the polycyclic presentation for G_F is given as follows:

1. $x_1^k = (\mathbf{1},\mathbf{0})$ as $x_1^k = (U_1,\mathbf{0})^k = (U_1^k,\mathbf{0}) = (\mathbf{1},\mathbf{0})$,
2. $[x_i,x_j] = (\mathbf{1},\mathbf{0})$ for $1 \le j < i \le m$ as U_F is abelian,
3. $[x_i,x_j] = (\mathbf{1},\mathbf{0})$ for $m+1 \le j < i \le m+n$ as $\mathcal{O}_F$ is abelian,
4. $x_{m+i}^{x_j} = x_{m+1}^{b_{i,j,1}}\cdots x_{m+n}^{b_{i,j,n}}$ with $i=1,\dots,n$ and $j=1,\dots,m$ and the coefficients $b_{i,j,k}$ arising from the expression $O_iU_j = b_{i,j,1}O_1+\dots+b_{i,j,n}O_n$,
5. $x_{m+i}^{x_j^{-1}} = x_{m+1}^{c_{i,j,1}}\cdots x_{m+n}^{c_{i,j,n}}$ with $i=1,\dots,n$ and $j=1,\dots,m$ and the coefficients $c_{i,j,k}$ arising from the expression $O_iU_j^{-1} = c_{i,j,1}O_1+\dots+c_{i,j,n}O_n$.

Hence, $G_F = PC\,\langle X|R\rangle$ with $S = (k,\infty,\dots,\infty)$. The second and third type of relations can be omitted as they are trivial. The Hirsch length of G_F is $m+n-1$.

5. Basics of computation theory and cryptography

In this chapter we recapitulate some important notions of computation theory as well as basic notions of computational security, as they will be needed further on.

5.1. Computation theory

In order to study some group-based problems and examine their algorithmic difficulty in Section 6.1, we recap a couple of basic and important notions of computation theory. We use the abstract model of *Turing machines* (TM), first introduced by A.M. Turing [Tur37], to define what it means for a function or a problem to be computable. We follow the notation of M. Sipser [Sip13, Section 3.1] and of S. Arora and B. Barak [AB09, Section 1.1, 1.2, 1.6 and 7.1], two books which give a general introduction to the theory of computation. The authors underline in their introduction that *"computation is not "merely" a practical tool. It is also a major scientific concept."* [AB09, p. xx]

TMs provide a very theoretic, simple and at the same time powerful model of computation which we will introduce here informally. A TM possesses a finite set of states and operates on an infinite tape composed of separate cells, using it as its unlimited memory. It also exhibits a tape head which can read and write symbols or change its position on the tape, depending on its current position, state and the symbol it reads.

At the beginning, the tape stores the input string and is blank everywhere else. If the machine has to store some information it writes it on the tape. The machine continues to compute, following a finite table of transition rules, until it decides either to output *accept*, to output *reject* or it never produces an output and continues forever without halting.

Definition 5.1. *Let $T\colon \mathbb{N} \to \mathbb{N}$ and $f\colon \{0,1\}^* \to \{0,1\}^*$ be some functions and M be a TM. We say that M* computes f in $T(n)$-time *if for every input $x \in \{0,1\}^*$, which determines the start configuration of M, it halts with $f(x)$ written on its output tape after at most $T(|x|)$ steps. The function f is called* computable *if a TM M and a function $T\colon \mathbb{N} \to \mathbb{N}$ exist such that M computes f in $T(n)$-time. If no such TM exists, f is called* uncomputable.

A TM which computes a function in polynomial-time is called *polynomial-time* algorithm and denoted as PT algorithm. A special class of functions $f\colon \{0,1\}^* \to \{0,1\}^*$ is the one of those which only output a single bit.

Definition 5.2. *A function $f\colon \{0,1\}^* \to \{0,1\}$ which maps strings to a single bit is called a* decision problem.

More precisely, a decision problem is the computational problem of computing the bit $f(x)$ for a given $x \in \{0,1\}^*$, in other words the answer to a yes-no-question. We can assign to every decision problem f the language $L_f = \{x \in \{0,1\}^* \mid f(x) = 1\}$ and identify the computational problem of computing $f(x)$ with deciding the language L_f, that is to say, given an element x, decide whether $x \in L_f$.

With regard to complexity, we are interested in the class of decision problems which are computable in polynomial-time.

Definition 5.3. *Let $T\colon \mathbb{N} \to \mathbb{N}$ be some function. We define $\boldsymbol{DTIME}(T(n))$ to be the set of all decision problems $f\colon \{0,1\}^* \to \{0,1\}$ that are computable in $c \cdot T(n)$-time for some constant $c > 0$. The complexity class $\boldsymbol{P}$ is defined as*

$$\boldsymbol{P} := \bigcup_{c \geq 1} \boldsymbol{DTIME}(n^c).$$

Whereas the TMs defined above are deterministic, in cryptography we sometimes need to execute computations which are not deterministic. Therefore, we will introduce the concept of a *probabilistic Turing machine* (PTM), equipped with an additional tape which it uses to choose in every step randomly which transition rule to apply. A PTM which computes a function in polynomial-time is called *probabilistic polynomial-time* algorithm and notated as PPT algorithm.

In order to define the AAG protocol in Chapter 7, we need two TMs to be able to communicate with each other and therefore we equip them with a common communication tape. Such TMs are called *interactive Turing machines* (ITM). For a detailed definition of ITMs see the work of O. Goldreich [Gol01] or R. Canetti [Can00].

In Section 6.3 and 6.9 we need the concept of *reduction. "A reduction is a way of converting one problem to another problem in such a way that a solution to the second problem can be used to solve the first problem."* [Sip13, p. 187] Consequently, the hardness of the first problem induces the hardness of the second problem. In cryptography we often say that the hardness of the second problem *is based* on the hardness of the first problem.

More formally expressed, we define for two functions $f, g\colon \{0,1\}^* \to \{0,1\}^*$:

Definition 5.4. *The function f is* polynomial-time Cook reducible *to the function g if there is an polynomial-time oracle machine that computes f when run with an oracle which computes g.*

In computation theory, an oracle machine is a theoretical black box that responds to every query with a correct output. For convenience we simply say *Cook reducible* or *Cook reduction* and presume that it works in polynomial-time. If f is Cook reducible to g and g is Cook reducible to f, we say that f and g are *Cook equivalent.* Figure 3 illustrates the idea of a Cook reduction.

During a Cook reduction we are allowed to use the oracle for the function g as many times as needed.

As an example we can look at the problems of multiplication and squaring. We assume, that we know how to add, subtract and divide by two. The first problem is reducible to the second problem via the following equation:

$$a \cdot b = \frac{\left((a+b)^2 - a^2 - b^2\right)}{2}.$$

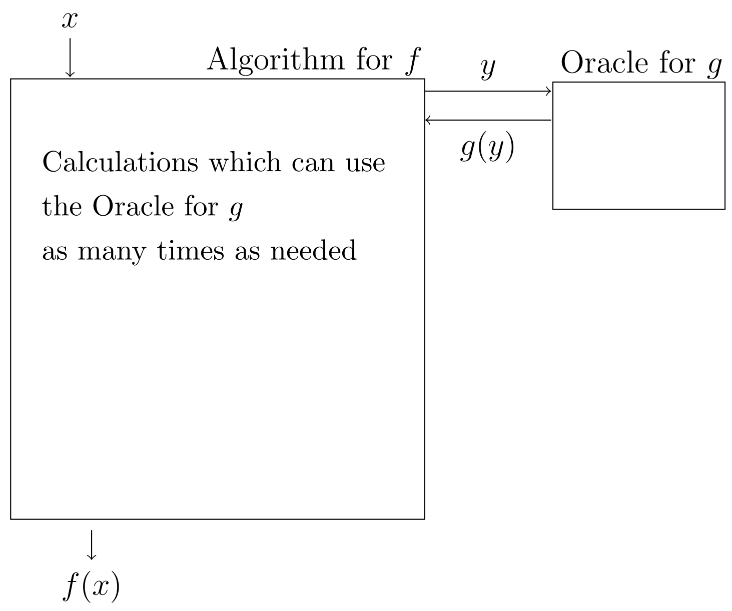
x
Algorithm for f
y
Oracle for g
$g(y)$
Calculations which can use the Oracle for g as many times as needed
$f(x)$

Figure 3: Cook reduction

This is a Cook reduction as we only need to calculate a constant number of squares. Conversely, we can solve the problem of squaring by using the solution to the problem of multiplying once.

In cryptography, two common hardness reductions are given for the so-called *discrete-logarithm problem* (DLog), the *computational Diffie-Hellman problem* (CDH) and the *decisional Diffie-Hellman problem* (DDH). J. Katz and Y. Lindell show that the hardness of DLog is based on the hardness of CDH and the hardness of CDH is based on the hardness of DDH [KL15, Chapter 8]. In other words, we can use a PPT algorithm which computes DLog (resp. CDH) to construct a PPT algorithm that computes CDH (resp. DDH).

5.2. Computational security

In modern cryptography the concept of computational security is used. In this section we introduce its idea whereas we use the notations of J. Katz and Y. Lindell [KL15, Chapter 3 and 10]. Compared to the information-theoretic security concept, which we will not discuss in this work, it is a weaker concept as it only requires that security is preserved against *efficient* adversaries and these adversaries are allowed to succeed with some *very small probability*, in a way to be defined soon.

If we talk about an efficient adversary, we mean an algorithm which can be represented by a PPT algorithm, i.e., a PTM which works in polynomial-time as presented in the previous section.

In the following, we define what a very small probability of success exactly means.

Definition 5.5. *A function $f\colon \mathbb{N} \to \mathbb{R}$ is called* negligible *if for every positive polynomial p a natural number $N \in \mathbb{N}$ exists such that for all integers $\kappa > N$ it holds*

$$|f(\kappa)| < \frac{1}{p(\kappa)}.$$

In other words, the absolute values of a negligible function tend faster to 0 than the inverse of every polynomial. The functions $2^{-\kappa}, 2^{-\sqrt{\kappa}}, n^{-\log \kappa}$ are, for instance, all negligible.

In order to present the AAG protocol in Chapter 7, we define what it means for a key exchange (KE) protocol to be correct and one-way secure against eavesdropping. Two parties called Alice and Bob want to communicate over an insecure channel and therefore agree on a shared secret key, which they afterwards use for encryption. To calculate this shared key both parties, which can be seen as ITMs, run a probabilistic protocol Π.

We denote by r_A and r_B the random coins used by Alice and respectively by Bob. Furthermore, the output of Alice and Bob are marked as $\mathsf{output}_{\Pi,A}(1^\kappa, r_A, r_B)$ and $\mathsf{output}_{\Pi,B}(1^\kappa, r_A, r_B)$, where κ is the security parameter. The security parameter is a variable that determines the input size of the computational problem.

More precisely, Alice and Bob presented via some PPT algorithms not only get the random coins r_A and r_B as input but also the security parameter κ, expressed in unary representation $1^\kappa := 1 \dots 1$. Therefore, we can guarantee that the time complexity of both PPT algorithms are polynomial in the size of the input. By $\mathsf{transcript}_\Pi(1^\kappa, r_A, r_B)$ we denote the transcript of all messages sent by Alice and Bob while running Π.

Definition 5.6. *A protocol Π for key exchange is called* correct *if for every $\kappa \in \mathbb{N}$ it holds*

$$\mathbb{P}[\mathsf{output}_{\Pi,A}(1^\kappa, r_A, r_B) \neq \mathsf{output}_{\Pi,B}(1^\kappa, r_A, r_B)] = 0,$$

where the probabilities are taken over the choices of r_A and r_B.

In other words, we call a KE protocol correct if, by an honest execution of Π, both parties Alice and Bob always agree on the same key.

To define what a one-way secure KE protocol is, we introduce the one-way eavesdropping key exchange experiment.

Experiment 5.7. *The one-way eavesdropping key exchange experiment $\mathsf{KE}^{eav}_{Eve,\Pi}(\kappa)$ for a correct KE protocol Π consists of the following steps:*

1. *Alice and Bob choose random strings r_A and r_B as defined in the protocol Π.*
2. *They run the protocol Π and set $K := \mathsf{output}_{\Pi,A}(1^\kappa, r_A, r_B)$.*
3. *The adversary Eve gets the security parameter κ and the transcript as input $\mathsf{transcript}_\Pi(1^\kappa, r_A, r_B)$. Based on these informations she returns a string K'.*
4. *The output of this experiment is 1 if $K = K'$ and 0 if not.*

Definition 5.8. *A protocol* Π *for key exchange is called* one-way secure in the presence of eavesdropping adversaries *if for every PPT adversary Eve a negligible function* negl *exists such that*

$$\mathbb{P}[\mathsf{KE}^{eav}_{Eve,\Pi}(\kappa) = 1] \leq \mathsf{negl}(\kappa).$$

Informally, this means that no eavesdropping PPT adversary Eve is able to compute the shared key K of Alice and Bob, except for a negligible probability.

We want to point out that several different notions of secure key exchange protocols exist. One which is stronger than the one presented above is defined by J. Katz and Y. Lindell [KL15, Chapter 10]. It presumes that Eve can't even compute a single bit of the key K. One of the most common KE establishments is the Diffie-Hellman KE protocol which can be proven correct and secure in this stronger sense under the DDH assumption [KL15, Chapter 10].

6. Group-based problems

In order to construct a concrete PKC scheme which uses a polycyclic group as platform group, we have to find a problem related to polycyclic groups which is assumed to be hard to compute. Therefore, we first recap the three important group-based problems introduced by M. Dehn, namely the word, the conjugacy and the isomorphism decision problem. In the following, we show that these problems are computable in the more general class of polycyclic-by-finite groups. To show this, we use the so-called local-global-principle and the properties that polycyclic-by-finite groups are residually finite and conjugacy separable.

Thereafter, we state a faster working solution to the word problem for polycyclic groups via collection. One possibility to attack the AAG protocol introduced in the next chapter is to solve the subgroup-restricted multiple conjugacy search problem, which will be presented in detail. Whereas this problem is assumed to be hard for polycyclic groups in general, there are better solutions for some subclasses. These solutions are presented for finite polycyclic groups, for f.g. nilpotent groups and for virtually f.g. nilpotent groups. We also look at the CSP in some semidirect products of polycyclic groups.

6.1. Three decision problems

We recap the three decision problems on finitely presented groups formulated in 1911 by M. Dehn [Deh11].

1) *Word decision problem*: For any $g \in G$, decide if $g = 1$,

2) *Single conjugacy decision problem*: Decide for any $u, v \in G$ if u is conjugate to v, i.e., if a $w \in G$ exists such that $w^{-1}uw = v$. If this is the case, we write $u \sim v$ and $u^w = v$,

3) *Isomorphism decision problem*: Given groups G and G', decide if they are isomorphic.

The word decision problem is a special case of the conjugacy decision problem with $u = 1$ and $v = g$. There is also a generalized version of the word decision problem.

4) *Generalized word decision problem*: For any element $g, g_1, \ldots, g_n \in G$, determine if $g \in \langle g_1, \ldots, g_n \rangle$.

In arbitrary groups all those problems are uncomputable, see the article by J. Stillwell [Sti82] for a survey about it. However, in the case of polycyclic groups all of them are computable in the sense of Definition 5.1.

Theorem 6.1. *Polycyclic groups have a computable word decision problem.*

Theorem 6.2. *Polycyclic groups have a computable single conjugacy decision problem.*

Theorem 6.3. *Polycyclic groups have a computable isomorphism decision problem.*

The first result is due to A. Mal'cev [Mal83], who showed that every residually finite and finitely presented group has a computable word problem.

Theorem 6.4. *There exists an algorithm which, given a polycyclic-by-finite group G and an element $g \in G$, decides if $g = 1$.*

Proof. The main idea of the proof is to define two procedures. The first will detect any word in G which is non-trivial and the other will list all words in G which equal 1. Taking the results of both, we can solve the word problem. Lemma 3.3 forms the basis for the first algorithm. One may call this the *local-global-principle*. An element of G is trivial if and only if it is trivial in every finite quotient of G. So we list all finite groups in increasing order by enumerating all possible multiplication tables. Then we enumerate for each finite group F the finite number of set maps defined by the generators of G to F. If all relators from G are mapped to the trivial element in F we can extend this map to a homomorphism of groups. Hence, we get a list of all group homomorphisms from G to finite groups. Given a word g in G we go down the list and check if g is mapped to a non-trivial element. As G is residually finite, we can be sure, that there is a presentation of G for which an element $g \neq 1$ will be mapped to such an element. The second procedure finds all elements in G representing the trivial element by enumerating all consequences of the defining relators by calculating all products of conjugates of relators. If we cannot find g in the first list, we will find it in the second one and hence $g = 1$. $\square$

As every polycyclic group is a polycyclic-by-finite group, we also proved Theorem 6.1. G. Baumslag et al. [BCRS91] show, using a similar argument, that the generalized word problem of polycyclic-by-finite groups is also computable.

Definition 6.5. *A group G is called* conjugacy separable *if for any two elements u, v of G,*

$$uN \sim vN \text{ in } G/N \text{ for all } N \trianglelefteq G \text{ with } G/N \text{ finite } \Rightarrow u \sim v \text{ in } G.$$

More precisely, two elements which are not conjugate in G have non-conjugate images in some finite image of G. If we can show that polycyclic groups are conjugacy separable, we can use the *local-global-principle* from above to construct an algorithm for the conjugacy decision problem. This has been proven by E. Formanek [For76] and by V.N. Remeslennikov [Rem69].

In order to present the proof of E. Formanek, we use the two following results, given without proof. The first one is due to A. Mal'cev, shown for example in [Rob72, Theorem 3.25] or in [Seg83, Section 2, Theorem 4], and states that every polycyclic-by-finite group is nilpotent-by-(abelian-by-finite). The second was shown by E. Formanek [For76, Proposition 12]. Recall that a group G is called *abelian-by-finite* if it exhibits an abelian normal subgroup N such that G/N is finite.

Theorem 6.6. *Let G be a polycyclic-by-finite group, then G has a normal nilpotent subgroup N such that G/N is abelian-by-finite. If G is infinite, then N may be chosen so that its center $Z(N) = \{z \in N \mid zn = nz \text{ for all } n \in N\}$ is infinite.*

Proposition 6.7. *Let A be a f.g. abelian group and let H be an abelian-by-finite group acting on A. Suppose $x, y \in A$ and $y \notin H \cdot x$. Then there is an integer $m > 0$ such that $y \notin H \cdot x + mA$.*

Theorem 6.8. *Let G be a polycyclic-by-finite group. Then G is conjugacy separable.*

Proof. We assume that G is not conjugacy separable. The proof is composed of the following steps:

(1) Consider all pairs (M, H) of subgroups of G such that $M \trianglelefteq H$ and H/M is not conjugacy separable. Since G satisfies the maximal condition, there is a pair (M_0, H_0) with M_0 maximal. By replacing G by H_0/M_0 we may assume that H/M is conjugacy separable whenever $1 \neq M \trianglelefteq H \subset G$.

(2) As G is not conjugacy separable it has to be infinite. By Theorem 6.6 G has a normal nilpotent subgroup N such that G/N is abelian-by-finite with infinite center $Z(N)$.

(3) Choose $x, y \in G$ such that x and y are not conjugate in G but their images are conjugate in every finite quotient of G. Then they are conjugate in every proper quotient of G by (1). In particular, they are conjugate in $G/Z(N)$. By replacing y by a conjugate we can assume that $x = y \bmod Z(N)$.

(4) Let $K := \{g \in G \mid [x, g] \in Z(N)\}$, hence $K/Z(N)$ is just the centralizer of $xZ(N)$ in $G/Z(N)$. We claim that x and y are conjugate in every finite quotient of K. Therefore, let M be a normal subgroup of finite index in K. As $Z(N) \leq K$ and $M \trianglelefteq_f K$ it yields $M \cap Z(N)$ is of some finite index m in $Z(N)$. As the center $Z(N)$ is a characteristic subgroup of N and as N is normal in G we can deduce that $Z(N)$ is normal in G and therefore also $Z(N)^m$. Using the fact that the center of N is infinite, it follows that $Z(N)^m$ is a non-trivial normal subgroup of G contained in M. By (3), x and y are conjugate in $G/Z(N)^m$. Choose $g \in G$ such that $g^{-1}xg = y \bmod Z(N)^m$. Then

$$[x, g] = x^{-1}g^{-1}xg = x^{-1}y = 1 \bmod Z(N)$$

by (3), so $g \in K$, which shows that they are also conjugate in K/M and the claim is proven. We can now replace G by K and N by $N \cap K$ which contains $Z(N)$ and thus assume that $[x, G] \subset Z(N)$. Notice that the assumptions made in (1)-(3) remain valid.

(5) Let $K_1 := \{[x, k] \mid k \in N\}$. We claim that K_1 is normal in G. Let g be an arbitrary element in G and $\tilde{k} = [x, k]$ be an element in K_1. We have to show that $g^{-1}\tilde{k}g$ lies in K_1. Using the fact that $[x, G] \subset Z(N)$ and that N is normal in G, it holds that

$$gxg^{-1}x^{-1}k^{-1} = k^{-1}gxg^{-1}x^{-1},$$

which is equal to

$$g^{-1}x^{-1}k^{-1} = x^{-1}g^{-1}k^{-1}gxg^{-1}x^{-1}$$

and so it yields

$$g^{-1}\tilde{k}g = g^{-1}(x^{-1}k^{-1}xk)g = (g^{-1}x^{-1}k^{-1})xkg = x^{-1}(g^{-1}k^{-1}g)x(g^{-1}kg) \in K_1.$$

Since x and y are not conjugate in G, we can deduce

$$g^{-1}yg \neq k^{-1}xk = x[x,k],$$

for all $g \in G$ and $k \in N$. Hence, x and y are not conjugate in G/K_1, so $K_1 = 1$ by (3). This means that $x \in Z(N)$.

(6) $Z(N)$ is a G/N-module with the action given by conjugation and it holds

$$\{\text{conjugates of } x \text{ in } G\} = \text{ orbit of } x \text{ under } G/N.$$

Since G/N is abelian-by-finite by (2) and $Z(N)$ is f.g. and abelian, we are now in the situation of Proposition 6.7, with $A = Z(N)$ and $H = G/N$. If we write $Z(N)$ multiplicatively, it yields for some integer $m > 0$

$$y \neq g^{-1}xg \bmod Z(N)^m$$

for all $g \in G$. More precisely x and y are not conjugate in $G/Z(N)^m$. Since $Z(N)$ is infinite, $Z(N)^m$ is non-trivial and we obtain a contradiction to (3).

□

Hence, we also proved Theorem 6.2. Notice that we only showed the existence of a solution to the conjugacy decision problem. Its complexity is not polynomial in the general case of an arbitrary polycyclic group.

The isomorphism decision problem was the last to be proven computable in polycyclic, and even in polycyclic-by-finite groups, by D. Segal [Seg90]. As he showed a more general result in his paper, it goes beyond the scope of this work to give the proof. To the interested reader we recommend to read the original article.

6.2. Collection algorithm

Whereas we showed in Theorem 6.4 that the word problem is computable for every polycyclic-by-finite group, the given algorithm is not polynomial-time in the sense of Definition 5.3. For the use of polycyclic groups in cryptography we need the word problem to be computed very quickly. Therefore, we will introduce another, faster working algorithm.

Collection is a method that can be used to determine the unique normal form of an element in a group given by a consistent polycyclic presentation, i.e., $R(X) = S$ using the notation from Section 1.3. Thus, collection can be used to solve the word problem in such groups. The notations follow D. Holt [HEO05, Chapter 8].

Definition 6.9. *Let G be a polycyclic group given in a consistent polycyclic presentation $PC\langle X|R\rangle$ with relative orders $R(X)$.*

1. *A word w is called* collected *if $w = x_{i_1}^{a_1} \cdots x_{i_r}^{a_r}$ with $i_1 < i_2 < \cdots < i_r$ and integers $1 \leq a_j \leq r_j - 1$ if $r_j \neq \infty$. Otherwise it is called* uncollected.
2. *A word v in X is a* minimal uncollected subword *of the word w if v is a subword of w and has one of the following forms:*
   a) $v = x_{i_j}^{a_j} \cdot x_{i_{j+1}}$ *for* $i_j > i_{j+1}$,
   b) $v = x_{i_j}^{a_j} \cdot x_{i_{j+1}}^{-1}$ *for* $i_j > i_{j+1}$,
   c) $v = x_{i_j}^{a_j}$ *for* $r_{i_j} \neq \infty$ *and* $a_j \geq r_{i_j}$.

A reduced word, i.e., a word which does not contain redundant pairs of the form xx^{-1} or $x^{-1}x$, is collected if and only if it does not contain a minimal uncollected subword. The main idea of the collection algorithm is to use the polycyclic relations to successively eliminate minimal uncollected subwords by replacing them with equivalent words. For a formal description of the collection procedure see Algorithm 1. We will now prove that this process always terminates.

Algorithm 1: Collection

```
Data: A polycyclic presentation $PC\langle X|R\rangle$ with $S$ the set of power exponents
      and a word $w$ in $X$.
Result: A collected word equivalent to $w$.
while there exists a minimal uncollected subword of $w$ do
    choose a minimal uncollected subword $v$ of $w$;
    if $v = x_{i_j}^{a_j} \cdot x_{i_{j+1}}$ for $i_j > i_{j+1}$ then
        replace $v$ by $x_{i_{j+1}}(R_{i_j,i_{j+1}})^{a_j}$ in $w$;
    if $v = x_{i_j}^{a_j} \cdot x_{i_{j+1}}^{-1}$ for $i_j > i_{j+1}$ then
        replace $v$ by $x_{i_{j+1}}^{-1}(R'_{i_j,i_{j+1}})^{a_j}$ in $w$;
    if $v = x_{i_j}^{a_j}$ for $s_{i_j} \neq \infty$ and $a_j \geq s_{i_j}$ then
        let $a_j = qs_{i_j} + r$ with $r \in \{0,\dots,s_{i_j}-1\}$,
        replace $v$ by $x_{i_j}^{r}(R_{i_j,i_j})^{q}$ in $w$;
return $w$;
```

Theorem 6.10. *The collection algorithm terminates.*

Proof. Let G be a group given in a consistent polycyclic presentation $PC\langle X|R\rangle$ with relative orders $R(X) = S$. The main idea is to prove the theorem by induction on n, the number of generators in X. For the base case assume that $n = 1$, where G is cyclic. It is obvious that the algorithm terminates. Let $PC_2 := PC\langle x_2,\dots,x_n|R_2\rangle$ be the polycyclic subpresentation given by deleting the generator x_1 and all relations containing it. For the induction step assume that the collection algorithm terminates for every word in the presentation PC_2. Let w be a word in X. The generator x_1 arises only finitely many times in w, thus we can write the word as $w = x_1^{a_1} v_1 x_1^{a_2} v_2 \cdots x_1^{a_r} v_r$, where $v_1, \ldots, v_r$ are words in $X_2 := \{x_2, \ldots, x_n\}$. If the algorithm finds a minimal uncollected subword containing x_1, it replaces it by a word in X_2. Since no new occurrences of x_1 appear, we get an equivalent word $x_1^{a_1+\cdots+a_r \bmod s_1} v$ after finitely many steps, with v a word in X_2. Due to the induction assumption the collection algorithm terminates on v and hence it also terminates on w. □

There have been various improvements of this basic algorithm, for instance in trying to find a good choice for the minimal uncollected word v in the collection procedure. This refinement has been studied in detail by C.R. Leedham-Green and L.H. Soicher [LGS90, Theorem 3]. They show that collection from the left, where one chooses the first occurrence of a minimal uncollected subword by parsing through w from the left, needs polynomial-time for finite p-groups, i.e., groups in which each element is of order p^n for a $n \in \mathbb{N}$. As every finite p-group is nilpotent, it is also a finite polycyclic group.

In order to determine the complexity of collection, the authors use the exponent sum of a word as measure. Following Definition 5.3, the word decision problem of finite p-groups lies in the complexity class **P**.

For infinite groups the complexity of the algorithm and a modified version are considered by V. Gebhardt [Geb02]. He shows that in the case of infinite polycyclic groups the complexity depends on the exponents occurring during collection, so it has no bound. Following J. Gryak and D. Kahrobaei [GK16], a determination of the precise computational complexity of the collection algorithm has not been attempted yet, but it is quite fast in practice. This means, although at the moment no proof exists that the word decision problem of infinite polycyclic groups lies in the complexity class **P**, experimental results indicate it.

6.3. Conjugacy search problem (CSP)

We showed that the conjugacy decision problem is computable in polycyclic groups. Subsequently, one may ask whether it is possible to determine a conjugating element if two conjugate elements are given. This problem is called the conjugacy search problem.

1) *Single conjugacy search problem*: Given two conjugate elements u, v, find a conjugating element w.

A slight variation of the single conjugacy search problem is the multiple conjugacy search problem (MCSP), sometimes also called simultaneous conjugacy search problem (SCSP). Given pairs of elements which are all conjugate, one has to find a common conjugating element.

2) *Multiple conjugacy search problem*: Given n pairs of conjugate elements (u_i, v_i), find a conjugating element w such that $u_i^w = v_i$ for all $1 \leq i \leq n$.

For the AAG protocol, which we define in Chapter 7, we need another variation of the CSP, the so-called subgroup-restricted multiple conjugacy search problem (SR-MCSP). Given pairs of elements which are all conjugate, one has to find a common conjugating element with the additional restriction that it has to be an element of a given subgroup.

3) *Subgroup-restricted multiple conjugacy search problem*: Given a subgroup H of a group G and n pairs of conjugate elements (u_i, v_i) in G, find a conjugating element w such that $u_i^w = v_i$ for all $1 \leq i \leq n$ and $w \in H$.

Due to B. Eick and D. Kahrobaei [EK04], we get the following Cook reduction in the sense of Definition 5.4:

Lemma 6.11. *The MCSP in polycyclic groups for n pairs of elements is Cook reducible to the CSP with a simultaneous determination of the corresponding centralizers.*

Proof. We prove the lemma by induction on n, the number of conjugate pairs. For the base case we assume to have one conjugate pair (u_1, v_1). This is already a version of the single CSP. Furthermore, let w_1 be an element with $u_1^{w_1} = v_1$ and let $G_1 = C_G(u_1) = \{g \in G \mid gu_1 = u_1 g\}$ be the centralizer of u_1 in G. Then every element $w \in G$ with $u_1^w = v_1$ can be written as $w = cw_1$ with $c \in G_1$ as $u_1^w = v_1 = u_1^{w_1}$ implies $u_1^{ww_1^{-1}} = u_1$ and hence $ww_1^{-1} \in G_1$.

By induction, we suppose to have $n+1$ conjugate pairs and an element w_n with $u_k^{w_n} = v_k$ for all $k \in \{1, \ldots, n\}$. Furthermore, we define $G_n := C_{G_{n-1}}(u_n) = C_G(u_1, \ldots, u_n)$. Then every element $w \in G$ with $u_i^w = v_i$ for every $i \in \{1, \ldots, n+1\}$ is of the form $w = cw_n$ with $c \in G_n$. Now we can solve the single CSP and compute $c \in G_n$ for which it holds $(u_{n+1})^c = (v_{n+1})^{w_n^{-1}}$ and at the same time calculate $G_{n+1} = C_{G_n}(u_{n+1})$. Now we can set $w_{n+1} := cw_n$ which is an element such that $u_k^{w_{n+1}} = v_k$ for all $k \in \{1, \ldots, n+1\}$.

Furthermore, every centralizer G_n is a subgroup of G and hence polycyclic itself. Thus, the MCSP reduces to n single CSP with a simultaneous determination of the corresponding centralizers. □

We want to point out that in general no polynomial-time algorithm exists to determine these centralizers. As we will see, in the case of f.g. nilpotent and virtually f.g. nilpotent groups, the situation is much better.

In the general case, one uses orbit-stabilizer methods to solve the CSP in polycyclic groups. More precisely, B. Eick and G. Ostheimer [EO03] state an algorithm to solve the orbit-stabilizer problem for polycyclic groups acting as subgroups of $GL(n, \mathbb{Z})$ on the elements of $\mathbb{Q}^n$. Recall, that every polycyclic group is isomorphic to a subgroup of $GL(n, \mathbb{Z})$ for some $n \in N$, see Section 1.4.

As an application, one can use this method to solve the CSP for elements within the group. It combines many different ideas, like using algorithms for the finite orbit-stabilizer problem, computations in number fields and linear methods for polycyclic groups. The algorithm was implemented in GAP [Gro15] and seems to work quite good for polycyclic groups of small Hirsch length. Unfortunately, it is difficult to give a precise analysis of its complexity, as there is no bound for the orbit lengths arising in the application of the finite orbit-stabilizer algorithm.

The results conform to the experimental results by B. Eick and D. Kahrobaei [EK04]. They tested the Eick-Ostheimer algorithm to solve the CSP with polycyclic groups of the semidirect product form from Chapter 4. When using the prototype polycyclic groups with high Hirsch length, computing the CSP required a much longer time whilst using those with a small Hirsch length.

6.4. Conjugacy in finite polycyclic groups

As we have seen in Section 1.2, a polycyclic group G with polycyclic sequence X is finite if and only if each entry in the sequence of relative orders $R(X)$ for X is finite. In this case, it is possible to determine a polycyclic sequence X' of G such that all entries in $R(X')$ are prime. If an entry r_i of $R(X)$ is not prime, i.e., $r_i = ab$ with $a, b \in \mathbb{N}$ and $a, b \neq 1$, then we just add the generator $x_i' = x_i^a$ to X. By iterating this process, we obtain a new sequence X' such that all entries in $R(X')$ are prime.

R. Laue et al. [LNS84] describe some basic algorithms applicable for finite solvable groups, hence for finite polycyclic groups. This includes an algorithm for computing orbits of a group G acting on some finite set Ω. In the special case of G acting on itself $\Omega = G$ via conjugation, the orbits of the action are just the conjugacy classes and elements of the same conjugacy class are conjugate. Hence, as an application we have a solution to the conjugacy problem in finite polycyclic groups.

Meanwhile, there have been improvements of this calculation, see for example the work by M. Mecky and J. Neubüser [MN89]. It is possible to decide in polynomial-time if two elements of a given finite polycyclic group G are conjugate, hence the conjugacy decision problem in finite polycyclic groups lies in the complexity class **P**, see Definition 5.3. Therefore, finite polycyclic groups are not of interest for use in cryptography and we will skip the details and recommend to read the original papers.

6.5. Conjugacy in finitely generated nilpotent groups

As we already mentioned, the CSP in polycyclic groups has no polynomial-time solution in general. However, for the class of f.g. nilpotent groups a much faster working solution using centralizers exists as shown by C. Sims [Sim94, Section 9.7].

Let G be a f.g. nilpotent group. We can assume that G is given in a nilpotent polycyclic presentation $PC\langle X|R\rangle$ as outlined in Lemma 2.7, for instance by refining the lower central series. Firstly, we need to compute the centralizer

$$C_G(g) = \{h \in G \mid hg = gh\}$$

of g in G.

Proposition 6.12. *There is an algorithm which computes the centralizer $C_G(g)$ of g in G for a given f.g. nilpotent group G.*

Proof. Let
$$G = G_1 \trianglerighteq G_2 \trianglerighteq \cdots \trianglerighteq G_n \trianglerighteq G_{n+1} = 1$$
be a polycyclic series of G given by a consistent nilpotent polycyclic presentation with generating set $X = \{x_1, \dots, x_n\}$, for instance the refined lower central series of G. We prove the proposition by induction on n, the length of this series. For the base case assume that $n = 1$, thus G is cyclic. Cyclic groups are abelian, hence for every $g \in G$ it yields $C_G(g) = G$. Now let $n > 1$ be the length of the nilpotent polycyclic series of G. Let $N := G_n = \langle x_n \rangle$ be the last factor of this series generated by x_n and let g be an element of G. As this series is determined by a nilpotent polycyclic presentation of G, every term G_i is normal in G, in particular $N \trianglelefteq G$. We assume by induction that we can compute the centralizer K of gN in G/N, i.e.,
$$K = C_{G/N}(gN) = \{h \in G/N \mid hgN = ghN\}$$
and find its preimage L under the map $G \to G/N$ given by
$$L = \left\{u \in G \mid u^{-1}gu \in gN\right\}.$$
We define the following map
$$f_g \colon L \to N \text{ with } u \mapsto [g, u].$$
Using Lemma 1.7, f_g defines a homomorphism into N: Let $u_1, u_2 \in L$, then
$$f_g(u_1u_2) = [g, u_1u_2] = [g, u_2][g, u_1][[g, u_1], u_2].$$
As $u_1, u_2 \in L$ the terms $[g, u_2], [g, u_1]$ lie in N which is cyclic and so they commute. Additionally, we have $[[g, u_1], u_2] = 1$ as $[g, u_1] \in N$ and $[G, N] = 1$. Thus,
$$f_g(u_1u_2) = [g, u_1][g, u_2] = f_g(u_1)f_g(u_2).$$
The centralizer $C_G(g)$ is just the kernel of f_g. There are methods to determine the kernel of a homomorphism of polycyclic groups [Sim94, Section 9.6]. Since N is a cyclic group, the computation of the kernel of f_g is just an easy application of these methods, but it goes beyond the scope of this work to present them in detail. □

Proposition 6.13. *There is an algorithm which decides for a given f.g. nilpotent group G if two elements $g, h \in G$ are conjugate in G. If they are, the conjugating element can be determined.*

Proof. Suppose we have two elements $g, h \in G$. Let
$$G = G_1 \trianglerighteq G_2 \trianglerighteq \cdots \trianglerighteq G_n \trianglerighteq G_{n+1} = 1$$

be a polycyclic series of G given by a consistent nilpotent polycyclic presentation, for instance the refined lower central series of G. We will prove the proposition by induction on the length of this series. For the base case assume that $n = 1$, thus G is cyclic. Cyclic groups are abelian, hence g and h are conjugate in G, if and only if $g = h$.

Now let $n > 1$ be the length of the nilpotent polycyclic series of G. Let $N := G_n = \langle x_n \rangle$ be the last factor of this series generated by x_n. We consider hN and gN in G/N. By induction we can decide if hN and gN are conjugate in G/N. If hN and gN are not conjugate in G/N, then they are not conjugate in G. If they are conjugate, then we can find an element $u \in G$ such that $u^{-1}(hN)u = u^{-1}huN = gN$ using the induction assumption. Replacing h by $u^{-1}hu$ we can assume that $hN = gN$ in G/N.

In this case, the conjugating element lies in the preimage L of the centralizer K of gN in G/N. Let f_g again be the homomorphism from L to N mapping u to $[g,u] = g^{-1}u^{-1}gu$. The conjugates of g by elements of L are the elements of $g \cdot f_g(L)$. Let $w = g^{-1}h$. Then h is conjugate to g if and only if $w \in f_g(L)$. If $w \in f_g(L)$, then $h = gw$ is a conjugate of g. Conversely, if h is conjugate to g, also hN and gN are conjugate in G/N. Hence we can assume that the conjugating element lies in L and that is why h has the form $h = gf_g(u)$ for a $u \in L$ and therefore $w = g^{-1}h$ is in $f_g(L)$. There are also methods to determine the image of a homomorphism of polycyclic groups [Sim94, Section 9.6]. If w lies in the image $f_g(L)$, we can find an element v such that $f_g(v) = w$ and hence $h = v^{-1}gv$. $\square$

As we are able to compute centralizers and the CSP relatively fast, we get a better solution to the MCSP in f.g. nilpotent groups following Lemma 6.11.

6.6. Conjugacy in virtually finitely generated nilpotent groups

One may ask if it is possible to use the solution to the MCSP in f.g. nilpotent groups of Section 6.5 to solve the MCSP in virtually f.g. nilpotent groups, i.e., groups which exhibit a f.g. nilpotent subgroup of finite index. C. Monetta shows in his slides [Mon17] the subsequent result, originally stated for supersolvable groups, but which is also correct for the more general case of virtually f.g. nilpotent groups.

As pointed out at the beginning of Section 1.1, being virtually f.g. nilpotent is the same as being (f.g. nilpotent)-by-finite, that is to say, having a f.g. nilpotent *normal* subgroup of finite index. Therefore, we can assume that a virtually f.g. nilpotent group G with H being a normal f.g. nilpotent subgroup of finite index r are given. Given $T = \{t_1, \dots, t_r\} \subset G$ a finite transversal to H in G, i.e., $G = \bigcup_{i=1}^{r} t_i H$.

Proposition 6.14. *Two elements g, h of G are conjugate in G if and only if g and h^{t_i} are conjugate in H for some $t_i \in T$.*

Proof. Suppose g and h^{t_i} are conjugate in H for some $t_i \in T$, so a $u \in H$ exists such that

$$g = (h^{t_i})^u = u^{-1}(h^{t_i})u = (u^{-1}t_i^{-1})h(t_i u)$$

and with $v := t_i u \in G$ it follows that g and h are conjugate in G. Conversely, suppose that g and h are conjugate in $G = \bigcup_{i=1}^{r} t_i H$. A $u \in H$ and a $t_i \in T$ exist such that $g = h^{t_i u} = (h^{t_i})^u$ and hence g and h^{t_i} are conjugate in H for some $t_i \in T$. □

In order to compute the CSP in G one has to compute the CSP at most r-times in the f.g. nilpotent group H. Using this result together with Proposition 6.13, we get a better solution to the CSP in virtually f.g. nilpotent groups compared to the general case.

As pointed out in Lemma 6.11, the problem of computing the SR-MCSP reduces to multiple versions of the CSP and the computation of centralizers $C_U(v)$ for any subgroup $U \subset G$ and any element $v \in G$. Using the fact that it holds $C_U(v) = C_G(v) \cap U$, we can focus on computing the centralizer of v in G for any $v \in G$. The idea proposed by C. Monetta [Mon17] is to calculate the centralizer in G via the centralizer in a f.g. nilpotent normal subgroup H of finite index.

Proposition 6.15. *Let $T = \{t_1, \ldots, t_r\} \subset G$ be a finite transversal to H in G and set $S = \{i \in \{1, \ldots, r\} \mid \exists h \in H$ such that $(v^{t_i})^h = v\}$. The centralizer $C_G(v)$ of an element v in G is given by*

$$C_G(v) = \left\langle C_H(v), t_i h_i \mid i \in S \text{ with } h_i \in H \text{ such that } (v^{t_i})^{h_i} = v \right\rangle .$$

Proof. „$\supset$“ Each element of the right side lies in $C_G(v)$ as $C_H(v) \subset C_G(v)$ and as every $t_i h_i$ fulfills $v^{t_i h_i} = v$ for $i \in S$.

„$\subset$“ Conversely, let g be an element of $C_G(v)$, hence $v^g = v$. As T is a transversal to H in G an element $h \in H$ and an index $i \in \{1, \ldots, r\}$ exist such that $g = t_i h$ and hence

$$(v^{t_i})^h = v^{t_i h} = v^g = v,$$

thus $i \in S$. Now we show that the right side is well defined, i.e., that for every t_i it is sufficient to take only one $h \in H$ such that $v^{t_i h} = v$. Let h' be another element of H such that $g' = t_i h' \in C_G(v)$. We have to prove that $gg'^{-1} \in C_H(v)$. It holds that

$$v^g = v^{t_i h} = v = v^{t_i h'} = v^{g'}$$

and hence

$$gg'^{-1} = (t_i h)(t_i h')^{-1} = t_i h h'^{-1} t_i^{-1} = \tilde{h} \in H$$

as $H \trianglelefteq G$, with $v^{\tilde{h}} = v$ and thus $gg'^{-1} = \tilde{h} \in C_H(v)$. □

In Section 7.3 we will show that we are searching for polycyclic groups which are non-virtually nilpotent, because they have exponential growth rate. Therefore, it is not alarming for cryptography that virtually f.g. nilpotent groups possess a better solution to the MCSP than the general case. Nevertheless, it could also be seen as an indication that the MCSP in polycyclic groups is not hard enough to use it for cryptographic protocols.

6.7. Conjugacy in supersolvable groups

Suppose G is a supersolvable group and a series $G = G_1 \trianglerighteq G_2 \trianglerighteq \cdots \trianglerighteq G_{n+1} = 1$ in which each G_i is a normal subgroup of G and each quotient group G_i/G_{i+1} is cyclic is given.

We claim that every supersolvable group is nilpotent-by-(finite and abelian).

Proposition 6.16. *If G is a supersolvable group, then a normal nilpotent subgroup H exists such that G/H is finite and abelian.*

Proof. Let G be as defined above. For every $i = 1, \dots, n$ we can consider the centralizer

$$C_G(G_i/G_{i+1}) = \{g \in G \mid [g, x] \in G_{i+1} \text{ for every } x \in G_i\}.$$

This subgroup is normal in G as for every $g \in C_G(G_i/G_{i+1})$, $h \in G$ and $x \in G_i$ it yields

$$\begin{aligned}[h^{-1}gh, x] &= (h^{-1}gh)^{-1}x^{-1}(h^{-1}gh)x \\ &= h^{-1}g^{-1}(hx^{-1}h^{-1})g(hxh^{-1})h \\ &= h^{-1}g^{-1}\tilde{x}^{-1}g\tilde{x}h \\ &= h^{-1}\tilde{g}h \in G_{i+1}\end{aligned}$$

for a $\tilde{x} \in G_i$ and a $\tilde{g} \in G_{i+1}$ using that G_i and G_{i+1} are normal in G. We define the intersection of all those centralizers as

$$H := \bigcap_{i=1}^{n} C_G(G_i/G_{i+1}).$$

This subgroup is normal in G as it is the intersection of finitely many normal subgroups of G. It is also a nilpotent group as $[H, G_i] \leq G_{i+1}$ for every $i = 1, \dots, n$ and hence $[H, G_n] = 1$ and therefore the $(n+1)$-th factor of the lower central series $\gamma_{n+1}(H) = 1$. Furthermore, it yields that the automorphism group $\operatorname{Aut}(G_i/G_{i+1})$ is finite and abelian for every index i as all factors are cyclic. The kernel of the homomorphism

$$\phi_i \colon G \to \operatorname{Aut}(G_i/G_{i+1}) \text{ defined by } g \mapsto \left\{n \mapsto g^{-1}ng\right\}$$

is just the centralizer $C_G(G_i/G_{i+1})$. Hence, the quotient $G/C_G(G_i/G_{i+1})$ is finite and abelian and accordingly G/H is also finite and abelian using that $H = \ker(\oplus_{i=1}^{n}\phi_i)$. □

Hence, supersolvable groups are virtually f.g. nilpotent and therefore we can use the result of Section 6.6 to get a better solution to the MCSP in the class of supersolvable groups.

6.8. Conjugacy in polycyclic groups with high Hirsch length

In this section, we question the observation made by J. Gryak and D. Kahrobaei [GK16]. The authors write referring to the CSP that *"[p]resently, there are no known [practical] algorithms for infinite polycyclic groups of high Hirsch length."* [GK16, p. 6] It is very important to underline that this statement does not hold for every infinite polycyclic group of high Hirsch length. We can assume that they only meant the infinite polycyclic groups of high Hirsch length as constructed in Chapter 4.

To illustrate our claim, we look at the following infinite polycyclic groups of high Hirsch length: With the help of the method ExamplesOfSomePcpGroups(n) defined in the GAP package *polycyclic* by B. Eick et al. [ENH04] we determine a polycyclic group G. This function gets as input $n \in \{1, \ldots, 16\}$ and returns an example of a polycyclic group. We focus on the examples 1, 2, 4, 6 and 7 as they are all non-virtually nilpotent, which is important for their use in cryptography, as we will explain later in Section 7.3.

To expand the group G to an infinite polycyclic group with higher Hirsch length, we simply add new generators, all having an infinite relative order and all commuting with the other generators. As they commute with the other generators, the CSP does not become harder, but the Hirsch length increases with every generator we add. The expanded group $Ex(G)$ is just the direct product of an example group and a free abelian group of finite rank.

The program code we use to test the CSP in the expanded groups in GAP is listed in Appendix A. Firstly, we present the code for calculating the average time needed to solve the CSP n times in a general polycyclic group G in Appendix A.1. Secondly, the program to construct the expanded group $Ex(G)$ by adding m generators to the example polycyclic group ExamplesOfSomePcpGroups(n) is listed in Appendix A.2.

Table 1 shows our results. Our tests were run on a Intel Core i5 2.60 GHz computer with 4 GB of RAM, Ubuntu 16.04 and GAP 4.7.9. With G we denote the underlying polycyclic group, $h(G)$ lists the Hirsch length of G, in *Average time* we store the average time in *ms* needed to solve the CSP for G in 100 cases. Furthermore, we list the Hirsch length of our expanded polycyclic group $Ex(G)$ and the average time in *ms* needed to solve the CSP in 100 cases. The Hirsch length of $Ex(G)$ is just the sum of the Hirsch length of the example group and the number of trivial generators added.

G	$h(G)$	Average time	$h(Ex(G))$	Average time
ExamplesOfSomePcpGroups(1)	4	131.16 ms	17	148.54 ms
ExamplesOfSomePcpGroups(2)	6	1132.92 ms	17	1196.16 ms
ExamplesOfSomePcpGroups(4)	3	56.2 ms	17	56.24 ms
ExamplesOfSomePcpGroups(6)	3	41.96 ms	17	46.68 ms
ExamplesOfSomePcpGroups(7)	4	53,58 ms	17	82.16 ms

Table 1: Average time needed to solve the CSP in some expanded examples

We can clearly observe that the CSP does not become harder if we trivially add further generators and hence increase the Hirsch length. In other words, the Hirsch length is no indicator for the hardness of the CSP.

6.9. Conjugacy in products of polycyclic groups

Similar to the procedure of the section above, we can construct general direct products of polycyclic groups. More precisely, we take two polycyclic groups G and H with polycyclic presentations $G = PC\langle x_1, \dots, x_n \mid R_G \rangle$ and $H = PC\langle y_1, \dots, y_m \mid R_H \rangle$. Their sequences of power exponents are given by S_G and S_H. The direct product is defined as

$$G \times H = PC\langle x_1, \dots, x_n, y_1, \dots, y_m \mid R_G \cup R_H \rangle,$$

where all generators x_i of G commute with all generators y_j of H. As usual we omit the trivial polycyclic relations.

Theorem 6.17. *The CSP in $G \times H$ is Cook equivalent to the CSP in G and H.*

Proof. Suppose x is an element of $G \times H$. Its normal form

$$x = x_1^{e_1} \cdots x_n^{e_n} y_1^{d_1} \cdots y_m^{d_m},$$

with $e_i, d_j \in \mathbb{Z}$ and $0 \le e_i < s_i$ for $s_i \in S_G$ and $0 \le d_j < \tilde{s_j}$ for $\tilde{s_j} \in S_H$ for every index $0 \le i \le n$ and $0 \le j \le m$, can be presented as a product $x = x_G x_H$ where $x_G = x_1^{e_1} \cdots x_n^{e_n}$ is an element in its normal form in G and $x_H = y_1^{d_1} \cdots y_m^{d_m}$ an element in its normal form in H. To solve the CSP in $G \times H$ one has to solve one CSP in G and one CSP in H as every x_i commutes with every y_j. It yields for any two elements $x, y \in G \times H$

$$x^{-1}yx = (x_G x_H)^{-1}(y_G y_H)(x_G x_H) = (x_G^{-1} y_G x_G)(x_H^{-1} y_H x_H).$$

Conversely, to solve the CSP for two conjugate elements x_G, y_G in G one has to solve the CSP for the two conjugate elements $x = x_G \cdot 1_H$ and $y = y_G \cdot 1_H$ in $G \times H$. Analogously, for two conjugate elements x_H, y_H in H one has to solve the CSP for the two elements $x = 1_G \cdot x_H$ and $y = 1_G \cdot y_H$ in $G \times H$. $\square$

Inspired by the idea of using semidirect products to obtain new polycyclic groups as in Chapter 4, we examine another possibility to construct semidirect products. Let G be a polycyclic group and g an element in G. We recall that the inner automorphism of g is given by $\phi_g \colon G \to G$ with $h \mapsto g^{-1}hg$ and thus we get a group homomorphism $\phi \colon G \to \mathrm{Aut}(G)$ defined by $g \mapsto \phi_g$. Subsequently, we can define the semidirect product $G \ltimes_\phi G$.

The first obstacle we meet with is that there is no known algorithm to compute the full automorphism group $\mathrm{Aut}(G)$ of a general polycyclic group G. One idea is to only consider the image of ϕ, which is the group of all inner automorphisms $\mathrm{Inn}(G)$. The kernel of ϕ is just the center $Z(G) = \{z \in G \mid gz = zg \text{ for all } g \in G\}$ of G. Thus, by the First Isomorphism Theorem it yields

$$\mathrm{Inn}(G) \cong G/Z(G).$$

As the center $Z(G)$ is just the intersection of the centralizer of all elements in G and as there is no known algorithm to compute centralizers in arbitrary polycyclic groups, this idea does not help us to overcome the obstacle.

Nevertheless, the following result holds:

Theorem 6.18. *The CSP in $G \ltimes_\phi G$ is Cook equivalent to the CSP in G.*

Proof. Let (h,g) and $(\tilde{h},\tilde{g})$ be two elements of $G \ltimes_\phi G$. Their product is defined as

$$(h,g)(\tilde{h},\tilde{g}) = (h\tilde{h}, \phi_h(g)\tilde{g}) = (h\tilde{h}, h^{-1}gh\tilde{g}). \tag{6.1}$$

The inverse of (h,g) is given by $(h,g)^{-1} = (h^{-1}, h^{-1}g^{-1}h)$ as it yields

$$(h,g)(h,g)^{-1} = (h,g)(h^{-1},h^{-1}g^{-1}h) = (hh^{-1}, (h^{-1}gh)\cdot(h^{-1}g^{-1}h)) = (1_G, 1_G). \tag{6.2}$$

Using Equation 6.1 and Equation 6.2 we can deduce for the conjugate

$$\begin{aligned}(h,g)^{-1}(\tilde{h},\tilde{g})(h,g) &= (h^{-1}, h^{-1}g^{-1}h)(\tilde{h}h, \tilde{h}^{-1}\tilde{g}\tilde{h}\cdot g)\\ &= (h^{-1}\tilde{h}h, \phi_{h^{-1}}(h^{-1}g^{-1}h)\tilde{h}^{-1}\tilde{g}\tilde{h}g)\\ &= (h^{-1}\tilde{h}h, g^{-1}\tilde{h}^{-1}\tilde{g}\tilde{h}g)\\ &= (\tilde{h}^h, \tilde{g}^{(\tilde{h}g)}).\end{aligned} \tag{6.3}$$

Hence, to solve the CSP in $G \ltimes_\phi G$ one has to solve two CSPs in G independently.

Conversely, to solve the CSP for two conjugate elements $h, \tilde{h}$ in G one has to solve the CSP for the two elements $(h, 1_G)$ and $(\tilde{h}, 1_G)$ in $G \ltimes_\phi G$. □

Additionally, one may ask if it is possible to also take other products than (semi)direct products. However, the class of polycyclic groups is not closed under taking free products or the more general amalgamated free products as the presentation does not need to stay polycyclic.

For example, the free product of two groups $G_1 = \langle X_1 | R_1 \rangle$ and $G_2 = \langle X_2 | R_2 \rangle$ with finite presentations is defined by

$$G_1 * G_2 := \langle X_1 \cup X_2 | R_1 \cup R_2 \rangle .$$

Hence, the generators of the first group are not related to the ones of the second group. This is not compatible with Definition 1.21 of polycyclic presentations.

With the same reasoning, it is easy to see that HNN-extensions of polycyclic groups are in general not polycyclic. For the precise definitions of amalgamated products and HNN-extensions, see [Löh15, Section 2.3.2]. J. Dyer [Dye80] shows a related, but weaker result, namely that the amalgamated free product $G_1 *_A G_2$ of two conjugacy separable groups G_1, G_2 over a finite group A is also conjugacy separable and that the HNN-extension $G_{*\phi}$ of a conjugacy separable group G with respect to an isomorphism ϕ of finite groups is also conjugacy separable. Hence, the conjugacy decision problem is still computable in those groups.

7. The Anshel-Anshel-Goldfeld key exchange protocol

One of the first and most studied cryptosystems based on a variation of the CSP is the Anshel-Anshel-Goldfeld key exchange protocol [AAG99], also called AAG protocol for short, which was introduced in 1999. In this chapter, we will present the protocol, examine its correctness and its security in terms of Definition 5.6 and 5.8. We will also investigate the impact of the growth rate on the security of the AAG protocol.

The notation follows A. Myasnikov and A. Ushakov [MU07], where the AAG protocol is studied over braid groups. In our case, the braid group is replaced by a polycyclic group. Let $G = PC\langle x_1, \dots, x_n | R\rangle$ be a polycyclic group given in a consistent polycyclic presentation with $X = \{x_1, \dots, x_n\}$ being a generating set. We choose the parameters $N_1, N_2, L_1, L_2, L \in \mathbb{N}$ with $L_1 \le L_2$ where all parameters and the number of generators n are polynomial in the security parameter $\kappa \in \mathbb{N}$. As introduced in Section 5.2 we have two parties, called Alice and Bob, who want to communicate over an insecure channel and therefore agree on a common secret key. A third party, Eve, tries to recover the shared key while eavesdropping on their communication.

Protocol 7.1. *The AAG protocol is composed of the following steps:*

A1) Alice randomly generates a N_1-tuple $\overline{a} = (a_1, \dots, a_{N_1})$ of words a_i in G, whereas each word is of length[1] *between L_1 and L_2 and sends it to Bob;*

B1) Bob randomly generates a N_2-tuple $\overline{b} = (b_1, \dots, b_{N_2})$ of words b_i in G, whereas each word is of length[1] *between L_1 and L_2 and sends it to Alice;*

A2) Alice randomly generates $A = a_{s_1}^{e_1} \cdots a_{s_L}^{e_L}$, where $a_{s_i} \leftarrow_\$ \overline{a}$ and $e_i \leftarrow_\$ \{-1, 1\}$;[2]

B2) Bob randomly generates $B = b_{t_1}^{d_1} \cdots b_{t_L}^{d_L}$, where $b_{t_i} \leftarrow_\$ \overline{b}$ and $d_i \leftarrow_\$ \{-1, 1\}$;[2]

A3) Alice computes the conjugates $b_i^A = A^{-1} b_i A$ for every index $1 \le i \le N_2$ and sends the tuple $\overline{b}^A = (b_1^A, \dots, b_{N_2}^A)$ to Bob;

B3) Bob computes the conjugates $a_i^B = B^{-1} a_i B$ for every index $1 \le i \le N_1$ and sends the tuple $\overline{a}^B = (a_1^B, \dots, a_{N_1}^B)$ to Alice.

A4) Alice defines the key K as the normal form of $A^{-1}(a_{s_1}^B)^{e_1} \cdots (a_{s_L}^B)^{e_L}$.

B4) Bob defines the key K as the normal form of $\left(B^{-1}(b_{t_1}^A)^{d_1} \cdots (b_{t_L}^A)^{d_L}\right)^{-1}$.

At this point we want to recall that $a_{s_i}^{e_i}$ with a_{s_i} an element of G and $e_i \in \{-1, 1\}$ denotes exponentiation, whereas a_i^B with a_i and B both elements of G denotes conjugation.

[1]Here, we mean by length the *natural word length* $l_X(g)$ of an element g of G generated by X induced by the *word metric* of Section 7.3. It will be discussed in more detail in Section 8.1.1.

[2]With $\leftarrow_\$$ we denote that the element is chosen uniformly randomly from a set. In the case of a tuple we choose elements uniformly randomly from the set determined by the elements of the tuple.

The following diagram illustrates the separate steps of the AAG protocol. Given a polycyclic group in a consistent polycyclic presentation $G = PC\langle x_1, \dots, x_n | R\rangle$ with $X = \{x_1, \dots, x_n\}$ being a generating set.

The AAG key exchange protocol

Alice		**Bob**
for $i \in \{1, \dots, N_1\}$		for $i \in \{1, \dots, N_2\}$
$\quad a_i \leftarrow_{\$} G$ s.t. $l_X(a_i) \in [L_1, L_2]$		$\quad b_i \leftarrow_{\$} G$ s.t. $l_X(b_i) \in [L_1, L_2]$
$\overline{a} = (a_1, \dots, a_{N_1})$		$\overline{b} = (b_1, \dots, b_{N_2})$
	$\xrightarrow{\quad \overline{a} \quad}$	
	$\xleftarrow{\quad \overline{b} \quad}$	
for $j \in \{1, \dots, L\}$		for $j \in \{1, \dots, L\}$
$\quad a_{s_j} \leftarrow_{\$} \overline{a}$,		$\quad b_{t_j} \leftarrow_{\$} \overline{b}$,
$\quad e_j \leftarrow_{\$} \{-1, 1\}$		$\quad d_j \leftarrow_{\$} \{-1, 1\}$
$A = a_{s_1}^{e_1} \cdots a_{s_L}^{e_L}$		$B = b_{t_1}^{d_1} \cdots b_{t_L}^{d_L}$
set $\overline{b}^A = (b_1^A, \dots, b_{N_2}^A)$		set $\overline{a}^B = (a_1^B, \dots, a_{N_1}^B)$
	$\xrightarrow{\quad \overline{b}^A \quad}$	
	$\xleftarrow{\quad \overline{a}^B \quad}$	
$K = A^{-1}(a_{s_1}^B)^{e_1} \cdots (a_{s_L}^B)^{e_L}$		$K^{-1} = B^{-1}(b_{t_1}^A)^{d_1} \cdots (b_{t_L}^A)^{d_L}$

Figure 4: The AAG key exchange protocol

7.1. Correctness of the AAG protocol

We now examine the correctness of the AAG protocol with respect to Definition 5.6.

Theorem 7.2. *Assuming that collection works in polynomial-time and that the AAG protocol is executed according to its rules, the AAG key exchange protocol is correct and the shared key is given by the commutator $K = [A, B]$.*

Proof. Alice can compute $K = [A, B]$ with the help of the conjugated tuple $\overline{a}^B$ she received from Bob:

$$\begin{aligned} A^{-1}(a_{s_1}^B)^{e_1} \cdots (a_{s_L}^B)^{e_L} &= A^{-1}B^{-1}(a_{s_1}^{e_1})B \cdots B^{-1}(a_{s_L}^{e_L})B \\ &= A^{-1}B^{-1}(a_{s_1}^{e_1} \cdots a_{s_L}^{e_L})B \\ &= A^{-1}B^{-1}AB \\ &= [A, B], \end{aligned}$$

knowing the factorization of A and by cancellation of BB^{-1}. Equally, Bob can compute with the help of the conjugated tuple $\overline{b}^A$ he received from Alice:

$$\begin{aligned} B^{-1}(b_{t_1}^A)^{d_1} \cdots (b_{t_L}^A)^{d_L} &= B^{-1}A^{-1}(b_{t_1}^{d_1})A \cdots A^{-1}(b_{t_L}^{d_L})A \\ &= B^{-1}A^{-1}(b_{t_1}^{d_1} \cdots b_{t_L}^{d_L})A \\ &= B^{-1}A^{-1}BA, \end{aligned}$$

knowing the factorization of B and by cancellation of AA^{-1}. Inverting this term, he also gets $K = [A, B]$.

In general, the element K obtained by Alice is not presented in the same way as the one calculated by Bob. By applying the collection algorithm from Section 6.2, both get K in its unique normal form. Hence, the AAG protocol is correct, whereupon the difference of the probabilities that Alice and Bob compute different keys is always zero. □

At this point it is important to underline that we assumed that Alice and Bob, who want to determine a common key, are represented by two ITMs which work in polynomial-time. As we saw in Section 6.2 in the case of infinite polycyclic groups the complexity of the collection algorithm to compute the normal form of an element could not be determined yet. However, in practice the algorithm works quite fast. That is why we can reasonably assume that collection works in polynomial-time. We still have to keep in mind that this is a weaker concept of correctness.

7.2. Security of the AAG protocol

In this section we analyze the security of the AAG protocol regarding Definition 5.8.

Proposition 7.3. *The hardness of the SR-MCSP is necessary for the security of the AAG protocol. In other words, if there is a PPT algorithm which computes the SR-MCSP, then a PPT algorithm which successfully attacks the AAG protocol exists.*

Proof. Suppose a PPT algorithm, presenting the eavesdropper adversary Eve, captures the tuple $\overline{b}$ and $\overline{b}^A$ and is able to compute the SR-MCSP. More precisely, it finds an element $A' \in \langle a_1, \ldots, a_{N_1} \rangle$ such that

$$\overline{b}^A = A^{-1}\overline{b}A = (A')^{-1}\overline{b}(A') = \overline{b}^{A'}$$

in G. (Equivalently, we could assume that Eve finds with the help of $\overline{a}$ and $\overline{a}^B$ an element $B' \in \langle b_1, \ldots, b_{N_2} \rangle$ such that $\overline{a}^B = \overline{a}^{B'}$ in G.) With the same reasoning as in Lemma 6.11 this A' is of the form $A' = cA$ for an element $c \in C_G(b_1, \ldots, b_{N_2}) \subset C_G(B)$. If Eve writes this A' as a word in the elements of $\overline{a} = (a_1, \ldots, a_{N_1})$, i.e., $A' = a_{u_1}^{f_1} \cdots a_{u_M}^{f_M}$ with $a_{u_i} \in \overline{a}$ and $f_i \in \{-1, 1\}$, she can compute

$$(A')^{-1}(a_{u_1}^B)^{f_1} \cdots (a_{u_M}^B)^{f_M} = (A')^{-1}B^{-1}a_{u_1}^{f_1}BB^{-1} \cdots BB^{-1}a_{u_M}^{f_M}B = (A')^{-1}B^{-1}(A')B,$$

knowing the factorization of A' and by cancellation of BB^{-1}. As A' is of the form $A' = cA$ for a $c \in C_G(B)$ she can deduce

$$(A')^{-1}B^{-1}(A')B = (cA)^{-1}B^{-1}(cA)B = A^{-1}(c^{-1}B^{-1}c)AB = A^{-1}B^{-1}AB = K.$$

Hence, Eve can compute the shared key K and is thus able to attack successfully the AAG protocol. $\square$

It is important to see that this does not imply that the AAG key exchange protocol is one-way secure in the presence of eavesdropping adversaries regarding Definition 5.8, based on the assumption that the SR-MCSP is hard. To be more precise, this statement would be a Cook reduction and would require that every PPT algorithm which successfully attacks the AAG protocol can compute the SR-MCSP. In Proposition 7.3 we showed an implication in the other direction.

Unfortunately, in several papers this has been mixed up. D. Garber et al. write that *"the security of the AAG protocol is based on the assumption that the subgroup-restricted simultaneous conjugacy search problem is hard."* [GKL15, p. 3] Likewise it is pointed by A. Myasnikov and A. Ushakov that *"the security of AAG protocol is partially based (but not equivalent) on the assumption that SR-SCSP is hard."* [MU07, p. 78] In the paper it is not defined what *partially based* exactly means. J. Gryak and D. Kahrobaei refer to [SU06] and claim that *"the security of AAG is based on both the simultaneous conjugacy search problem and the subgroup membership search problem"*. [GK16, p. 8] Combining both group problems we get the SR-MCSP.

As K. Blaney and A. Nikolaev [BN16] point out, the difficulty of Eve's task to calculate the shared key K also depends on the choice of the public tuples $\overline{a}$ and $\overline{b}$ from Alice and Bob. For instance, if every $a_i \in \overline{a}$ lies in the centralizer $C_G(b_1, \dots, b_{N_2})$ of Bob's tuple $\overline{b}$, then $K = 1$, independent of the secret keys A and B as $\overline{a}' = \overline{a}$. As it yields

$$C_G(b_1, \dots, b_{N_2}) = \bigcap_{i=1}^{N_2} C_G(b_i),$$

the probability that every a_i lies in the centralizer of every b_i becomes very small if we increase the security parameter κ and therefore increase the size of Bob's public set N_2.

7.3. Growth rate

We now investigate the impact of the growth rate on the security of the AAG protocol. In the AAG protocol, the elements A and B from Alice and Bob are words in the set of the tuples $\overline{a}$ and $\overline{b}$ of length less than or equal to L. Hence, the size of the key space of this key exchange protocol is determined by the growth rate of the underlying group. Informally, the growth rate gives the number of elements that can be written as a product of a given length.

To avoid that the eavesdropping adversary Eve can calculate the shared key K simply by doing an exhaustive search, namely testing every possible element, it is necessary to have a very large key space. This is ensured by a very high growth rate, which can be seen as a heuristic for the security of the AAG protocol, but does not guarantee it. The notation of this section follows C. Löh's book *Geometric group theory, an introduction* [Löh15].

Let G be a f.g. group with generating set X. The *word metric* d_X on G with respect to X is given by

$$d_X(g,h) := \min\left\{n \in \mathbb{N} \mid \exists x_1,\dots,x_n \in X \cup X^{-1} \text{ such that } g^{-1}h = x_1 \cdots x_n\right\}$$

for all $g, h \in G$. The word metric on $\mathbb{Z}$, for example, corresponding to the generating set $\{1\}$ coincides with the metric on $\mathbb{Z}$ induced by the standard metric on $\mathbb{R}$. On the other hand, if we take $\mathbb{Z}$ as the generating set, all group elements have distance 1 from every other group element.

The *growth rate* of G is specified by its *growth function* given by $\beta_{G,X} \colon \mathbb{N} \to \mathbb{N}$ with

$$\beta_{G,X}(n) := \#\{g \in G \mid d_X(g, 1_G) \leq n\}\,.$$

In other words, the growth function is defined as the number of elements which lie in the closed ball of radius n around the identity element 1_G with respect to the word metric d_X of G. A group has an *exponential growth rate* if there are constants $u > 0$ and $v > 1$ such that $u \cdot v^m \leq \beta_{G,X}(m)$ for every integer $m \geq 1$. A group has a *polynomial growth rate of degree* $\leq E$ if there is a constant $c > 0$ such that $\beta_{G,X}(m) \leq c \cdot m^E$ for every integer $m \geq 1$. Finite groups have, for example, a constant growth rate and free groups of rank greater 1 have an exponential growth rate.

Recap from Section 1.1 that a group is said to be virtually nilpotent if it has a subgroup of finite index which is nilpotent. J. Wolf proves that polycyclic groups have a polynomial growth rate if they have a nilpotent subgroup of finite index and an exponential growth rate if they have no such subgroup, i.e., if they are non-virtually nilpotent [Wol68]. That is why we are looking for non-virtually nilpotent polycyclic groups.

To decide whether a given polycyclic group G is virtually nilpotent or not, one can determine the *Fitting Subgroup* $\mathrm{Fitt}(G)$. It is defined as

$$\mathrm{Fitt}(G) := \langle N \mid N \trianglelefteq G \text{ and } N \text{ nilpotent } \rangle\,.$$

Fitting's Theorem states that the product of finitely many nilpotent normal subgroups is still nilpotent. Hence, in the case of groups satisfying the maximal condition, $\mathrm{Fitt}(G)$ is nilpotent. Moreover, it is the unique largest normal nilpotent subgroup of G. If the index $[G : \mathrm{Fitt}(G)]$ is finite, then G is virtually nilpotent. The GAP package *polycyclic* by B. Eick et al. [ENH04] provides a method to determine this index.

For f.g. rational matrix groups another solution is given by B. Assmann and B. Eick [AE07]. The idea is to first decide whether a given f.g. rational matrix group G is virtually solvable. Then they give an algorithm to decide if G is polycyclic or virtually polycyclic. As a last step it is possible to test if G is nilpotent or virtually nilpotent. The algorithm for testing polycyclicity is implemented in the package *polenta* by B. Assmann [Ass05] in GAP.

Given a virtually nilpotent group G, we can look at the product $G \times \mathbb{Z}$. As we added one more infinite cyclic factor, one may think of this as a canonical way to construct non-virtually nilpotent groups out of virtually nilpotent ones. However, this is not the case. As G is virtually nilpotent, it has a normal nilpotent subgroup N of finite index. Furthermore, $\mathbb{Z}$ is abelian and therefore nilpotent. Using Fitting's Theorem we can deduce that $N \times \mathbb{Z}$ is a nilpotent, normal subgroup of $G \times \mathbb{Z}$ with finite index. Hence $G \times \mathbb{Z}$ is also virtually nilpotent.

8. Cryptanalysis

With the development of cryptographic protocols based on non-commutative groups, as the AAG protocol introduced in the previous chapter, also new attacks against them have been developed. In this chapter we review two of the known attacks against the AAG protocol, namely the Length-Based Attack (LBA) and the Field-Based Attack (FBA). After presenting the main idea of the LBA, we discuss the role of its length function when testing it on different types of polycyclic groups. Afterwards, we present two variants of the LBA, the LBA with backtracking and the Memory LBA, two out of many versions which have been developed in the last decade. The chapter closes with a presentation of the FBA, an attack against the prototype polycyclic groups.

8.1. Length-Based Attack (LBA)

The LBA was originally introduced for cryptanalysis against the AAG key exchange protocol over braid groups by J. Hughes and A. Tannenbaum [HT03]. It is a heuristic attack against PKC systems based on the conjugacy search problem for finitely presented groups by using the length of a word as a heuristic. It rests upon the idea that the conjugation of the correct element should decrease the length of the captured tuple. In the case of attacking the AAG protocol, using the notation of Chapter 7, the captured tuple can be $\overline{b}^A = (b_1^A, \dots, b_{N_2}^A)$. Each b_i^A is obtained by a sequence of conjugations of b_i by the factors of the private key $A = a_{s_1}^{e_1} \cdots a_{s_L}^{e_L}$.

$$b_i \longrightarrow (a_{s_1}^{-e_1}) b_i (a_{s_1}^{e_1}) \longrightarrow \dots \longrightarrow (a_{s_L}^{-e_L} \cdots a_{s_1}^{-e_1}) b_i (a_{s_1}^{e_1} \cdots a_{s_L}^{e_L}) = b_i^A \tag{8.1}$$

If we conjugate $\overline{b}^A$ with elements from the subgroup $\langle a_1, \dots, a_{N_1} \rangle$ and the length of the resulting tuple has been decreased, then we have found a candidate for the conjugating factor. We repeat the process of conjugation with the decreased-length tuple until a longer candidate for the conjugation factor appears. The process ends when the conjugated captured package is the same as $\overline{b} = (b_1, \dots, b_{N_2})$, which is also captured while eavesdropping. The conjugating element we are looking for is just the inverse of the sequence of conjugating factors we determined. The basic idea of the LBA can be illustrated by going from right to left in Equation 8.1.

8.1.1. The length function

The crux of the LBA is to find a length function l such that $l(a^{-1}ba) \gg l(b)$ for the majority of elements $a, b \in G$. D. Garber et al. [GKL15] propose the sum of the absolute values of the exponents in its normal form as length of an element. More precisely, let $g \in G = PC\langle x_1, \dots, x_n | R\rangle$ be given in its normal form $g = x_1^{e_1} \cdots x_n^{e_n}$. Then we define the *length* $l(g)$ of g as

$$l(g) := \sum_{i=1}^{n} |e_i|. \tag{8.2}$$

Note that if G is given in a consistent polycyclic presentation, one can compute the unique normal form using collection from Section 6.2. Furthermore, we define the *total length* of a captured package $\overline{c} = (c_1, \ldots, c_k)$ as

$$|\overline{c}| = \sum l(c_i).$$

At this point, we want to emphasize that the length function l for a polycyclic group G does not coincide with the *natural word length function* l_X induced by the word metric of Section 7.3 and defined as $l_X(g) := d_X(g, 1)$, where X is the set of generators of G.

Example 8.1. *Given the polycyclic group G defined by*

$$G = PC \left\langle x_1, x_2, x_3 \mid x_2^{x_1} = x_3, x_2^{x_1^{-1}} = x_2^{-7} x_3, x_3^{x_1} = x_2 x_3^7, x_3^{x_1^{-1}} = x_2 \right\rangle$$

with $X = \{x_1, x_2, x_3\}$ and the element $g = x_3 x_2$. It has length $l_X(g) = 2$ in terms of the natural word length function, but as its normal form is given by $g = x_1 x_2 x_3^7$ it has length $l(g) = 9$ as defined in Equation 8.2.

D. Garber et al. [GKL15] show that the defined length function ensures $l(a^{-1}ba) \gg l(b)$ by the use of experimental results on the polycyclic groups arising from a number field of the form $G_F = U_F \ltimes \mathcal{O}_F$, as defined in Chapter 4. For this purpose, they randomly choose two elements a and b of length between 10 and 13, for consistency with the LBA parameters they later use, and compute the difference $l(b^a) - l(b)$. They perform 100 tests for each group and the average difference is recorded. The results are listed in Table 2. The polynomial $f(x) \in \mathbb{Z}[x]$ determines the number field $F = \mathbb{Q}[x]/(f)$, which specifies the polycyclic group G_F of Hirsch length $h(G_F)$.

Polynomial $f(x)$	$h(G_F)$	Average difference
$x^2 - x - 1$	3	79.92
$x^5 - x^3 - 1$	7	80.17
$x^{11} - x^3 - 1$	16	44.93

Table 2: Length-difference for the semidirect product polycyclic group G_F [GKL15]

To check if these experimental results also hold for other types of polycyclic groups than the prototype of the form $G_F = U_F \ltimes \mathcal{O}_F$, we additionally test it for Heisenberg groups, infinite metacyclic groups and some examples of non-virtually nilpotent polycyclic groups.

The *Heisenberg group* H^{2n+1} with $n \in \mathbb{N}$ is given as the group of $(n+2) \times (n+2)$ matrices of the form

$$\begin{pmatrix} 1 & x_1 & \cdots & x_n & c \\ 0 & 1 & 0 & \cdots & y_n \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & y_1 \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix},$$

where $c, x_i, y_i \in \mathbb{R}$. The Hirsch length of H^{2n+1} is $2n+1$. As pointed out by D. Kahrobaei and H.T. Lam [KL14], the group exhibits the finite presentation

$$\langle a_1, \ldots, a_n, b_1, \ldots, b_n, c \mid [a_i, b_i] = c, [a_i, c] = [b_i, c] = 1, [a_i, a_j] = [b_i, b_j] = 1 \text{ for } i \neq j \rangle .$$

Being f.g. nilpotent, clearly every Heisenberg group is polycyclic. With Example 2.8 we already saw an example of a Heisenberg group.

Infinite metacyclic groups are classified by J.R. Beuerle and L.-C. Kappe [BK00]. In Theorem 3.2 they show that every infinite metacyclic group is isomorphic to

$$G'(m,n,r) := \left\langle a, b \mid a^n = b^m = 1, [a,b] = a^{1-r} \right\rangle ,$$

where $m, n, r \in \mathbb{Z}$ with $mn = 0$ and r relatively prime to m. By mapping $b \mapsto x^{-1}$ and $a \mapsto y$ this presentation is easily transformed into the polycyclic presentation

$$G(m,n,r) := \langle x, y \mid x^m = y^n = 1, y^x = y^r \rangle .$$

Using these presentations one can implement the Heisenberg groups and the infinite metacyclic groups in the GAP system [Gro15]. The implementations are part of the GAP package *polycyclic* by B. Eick et al. [ENH04]. Unfortunately, the results for both types are significantly inferior to those of D. Garber et al. [GKL15], see Table 3. The program code we used for the calculations in GAP is listed in Appendix A.3. Our tests were run on a Intel Core i5 2.60 GHz computer with 4 GB of RAM, Ubuntu 16.04 and GAP 4.7.9.

G	$h(G)$	Average difference
H^5	5	3.53
H^{11}	11	2.93
H^{15}	15	1.16
$G(0,15,7)$	1	1.26
$G(0,31,9)$	1	10.49

Table 3: Length-difference for Heisenberg groups and infinite metacyclic groups

Notice that every Heisenberg group is a f.g. nilpotent group and hence we can use the solution of Section 6.5 for solving the CSP. K. Blaney and A. Nikolaev show that the SR-MCSP in Heisenberg groups is solvable in quasi-quintic time with respect to the number of generators of the subgroup or the rank of the elements in the platform group and linear with respect to the bit size of the elements [BN16]. Hence, they are not suitable as platform groups for the AAG protocol.

Furthermore, infinite metacyclic groups have at most Hirsch length 1, which is very small. As pointed out in Section 6.3, the Eick-Ostheimer algorithm can be used to break the conjugacy problem in these groups. Hence, they are also not suitable as platform groups for the AAG protocol.

If we look at the polycyclic groups given by the method ExamplesOfSomePcpGroups(n) with $n \in \{1, 2, 4, 6, 7\}$ which are non-virtually nilpotent, one gets much better results listed in Table 4. Taking these results into account, the length function defined by D. Garber et al. [GKL15] seems to be a suitable candidate for testing the LBA against polycyclic groups.

G	$h(G)$	Average difference
ExamplesOfSomePcpGroups(1)	4	26.23
ExamplesOfSomePcpGroups(2)	6	15.54
ExamplesOfSomePcpGroups(4)	3	38500.57
ExamplesOfSomePcpGroups(6)	3	413.92
ExamplesOfSomePcpGroups(7)	4	3374.67

Table 4: Length-difference for examples of non-virtually nilpotent polycyclic groups

8.1.2. LBA with backtracking

The first algorithm for LBA coming to mind is the *LBA with backtracking*, presented by A. Myasnikov and A. Ushakov [MU07], which enumerates all possible conjugations and adds those whose length has been decreased to an array S, see Algorithm 2.

```
Algorithm 2: LBA with backtracking
    Data: $\overline{a} = (a_1, \ldots, a_{N_1}), \overline{b} = (b_1, \ldots, b_{N_2})$ and $\overline{b}^A = (b_1^A, \ldots, b_{N_2}^A)$ for an element
          $A \in G$.
    Result: An element $A' \in \langle a_1, \ldots, a_{N_1} \rangle$ such that $b_i^A = b_i^{A'}$ for all $i = 1, \ldots, N_2$ or
            FAIL if the algorithm cannot find such $A'$.
    Initialize $S = \left\{(\overline{b}^A, id_G)\right\}$;
    while $S \neq \emptyset$ do
        Choose $(\overline{c}, x) \in S$ such that $|\overline{c}|$ is minimal and remove it from $S$;
        for $i = 1, \ldots, N_1$ and $e = \pm 1$ do
            Compute $d_{i,e} = |\overline{c}| - |\overline{c}^{a_i^e}|$;
            if $\overline{c}^{a_i^e} = \overline{b}$ then
                Output inverse of $xa_i^e$ and stop;
            if $d_{i,e} > 0$ then Add $(\overline{c}^{a_i^e}, xa_i^e)$ to $S$;
            ;
    Otherwise, output FAIL;
```

D. Garber et al. consider the prototype polycyclic groups of Chapter 4 and run their experiment with the following parameters: The polycyclic group G_F is constructed by the polynomial $f(x) = x^9 - 7x^3 - 1$, having a Hirsch length of $h(G_F) = 14$, the length of random elements is between $L_1 = 10$ and $L_2 = 13$, the number of factors in the private key is $L = 10$ and the length of both tuples are $N_1 = N_2 = 20$. Then the LBA with backtracking ran for 48.68 hours with a success rate of 0%.

8.1.3. Memory LBA

Following D. Garber et al. [GKL15], the algorithm which works best in computing the CSP, considering the success rate and the running time, is the so-called *Memory LBA*, firstly presented in [GKT+05]. This variant uses an array S of fixed size M. In every iteration, all the elements of S are conjugated, but only the M smallest are inserted back into S. Furthermore, we define a time-out as halting condition. For a description of it, see Algorithm 3.

Algorithm 3: Memory LBA

```
Data: $\overline{a} = (a_1, \dots, a_{N_1}), \overline{b} = (b_1, \dots, b_{N_2})$ and $\overline{b}^A = (b_1^A, \dots, b_{N_2}^A)$ for an element
      $A \in G$.
Result: An element $A' \in \langle a_1, \dots, a_{N_1} \rangle$ such that $b_i^A = b_i^{A'}$ for all $i = 1, \dots, N_2$ or
        FAIL if the algorithm cannot find such $A'$.
Initialize $S = \left\{ (|\overline{b}^A|, \overline{b}^A, id_G) \right\}$;
while not time-out do
    for $(|\overline{c}|, \overline{c}, x) \in S$ do
        Remove $(|\overline{c}|, \overline{c}, x)$ from $S$;
        Compute $\overline{c}^{a_i^e}$ for all $i \in \{1, \dots, N_1\}$ and $e = \pm 1$;
        if $\overline{c}^{a_i^e} = \overline{b}$ then
            Output inverse of $xa_i^e$ and stop;
        Save $(|\overline{c}^{a_i^e}|, \overline{c}^{a_i^e}, xa_i^e)$ in $S'$;
    Sort $S'$ by the first element of every tuple;
    Copy the first $M$ elements into $S$ and delete the rest of $S'$;
Otherwise, output FAIL;
```

D. Garber et al. consider the prototype polycyclic groups of Chapter 4 and suggest, based on their experimental results, the following parameters: The polycyclic group G should have a Hirsch length of $h(G) = 16$, the length of random elements should be between $L_1 = 20$ and $L_2 = 23$, the number of factors in the private key should be $L = 20$ and the memory size should be $M = 1000$. Then, all known variants of the LBA, in particular the Memory LBA, have a success rate of 0%.

They conclude that the LBA is insufficient for cryptanalyzing the polycyclic groups of high enough Hirsch length. As in Section 6.3 we want to point out that this statement does not hold if we just expand a given polycyclic group trivially to one with a much higher Hirsch length.

8.2. Field-Based Attack

Whereas D. Garber et al. [GKL15] show that the LBA has no success using the prototype polycyclic groups from Chapter 4 with Hirsch length higher than 15, M. Kotov and A. Ushakov found a way to attack the AAG protocol based on this special class of polycyclic groups, called the Field-Based Attack (FBA) [KU15].

A polycyclic group of the semidirect product form $G_F = U_F \ltimes \mathcal{O}_F$ as introduced in Chapter 4 is given. Furthermore, the number field F may be represented as a set of matrices as in Equation 4.1:

$$F = \left\{ a_0 \mathbf{1} + a_1 M + a_2 M^2 + \cdots + a_{n-1} M^{n-1} \mid a_0, \dots, a_{n-1} \in \mathbb{Q} \right\}.$$

The FBA is based on the following idea: Extend the group G_F and work in the extended group $G_F^* := F^* \ltimes F$ instead. In general, this group is not f.g. and hence not polycyclic, but its elements can be effectively represented by pairs of matrices.

Using the notation of Chapter 7, one way to attack the AAG protocol is to find an element $A' \in \langle a_1, \dots, a_{N_1} \rangle$ such that it holds that $\overline{b}^A = (A')^{-1} \overline{b} (A')$ in G_F or an element $B' \in \langle b_1, \dots, b_{N_2} \rangle$ such that $\overline{a}^B = (B')^{-1} \overline{a} (B')$ in G_F. We suppose that the attacker receives the following system of N_2 conjugacy equations

$$\begin{cases} (A')^{-1} b_1 (A') & = b_1^A \\ & \dots \\ (A')^{-1} b_{N_2} (A') & = b_{N_2}^A \end{cases},$$

with unknown $A' \in G_F = U_F \ltimes \mathcal{O}_F$. We consider it now as a system over G_F^* with

$$A' = (C, S), \quad b_i = (B_i, T_i), \quad b_i^A = (B_i^A, T_i^A) \text{ in } G_F^*.$$

We use Equation 4.2 of Chapter 4

$$(D, T)^{(C,S)} = (C, S)^{-1} \cdot (D, T) \cdot (C, S) = (D, S(\mathbf{1} - D) + TC)$$

and focus on the second entry of the tuple to transform it into a system of N_2 linear equations over the field F with two unknowns C and S

$$\begin{cases} S(\mathbf{1} - B_1) + T_1 \cdot C & = T_1^A \\ & \dots \\ S(\mathbf{1} - B_{N_2}) + T_{N_2} \cdot C & = T_{N_2}^A \end{cases}.$$

Hence, we can find a unique solution if the coefficient matrix of this system has rank two over F by using a deterministic, polynomial-time algorithm. In this case, the attacker finds a solution A' which is equal to the original private key A of Alice.

M. Kotov and A. Ushakov implemented an attack in GAP [KU15]. They show that the FBA is able to find first Alice's private key A and subsequently the shared key K regardless of the group's Hirsch length.

They also present an attack for G_F given by a polycyclic presentation as in Section 4.2. For this purpose, they transform it into a semidirect product of an abelian matrix group and $\mathbb{Z}^n$. The interested reader is referred to the original paper for further details.

In view of the FBA the polycyclic groups of Chapter 4 should not be used for cryptographic purposes. It also reminds us that a low success rate of an attack against a cryptographic protocol does not prove its security - as it may be possible to find other attacks against it.

9. Further work

We complete this thesis presenting possible further work within this field of research. There are many unanswered questions related to the class of polycyclic groups and their application in cryptography. Within the scope of this work we studied in detail the conjugacy search problem in polycyclic groups for use in cryptography. Having one more look at Figure 2, we could refine it to

$$\{ \text{ f.g. nilpotent } \} \subsetneq \{ \text{ supersolvable } \} \subsetneq \{ \text{ virtually f.g. nilpotent } \} \subsetneq \{ \text{ polycyclic } \}$$

Figure 5: Relation between (virtually) f.g. nilpotent, supersolvable and polycyclic groups

As we showed in Section 6.5, 6.6 and 6.7, there are better solutions to the MCSP in f.g. nilpotent, in supersolvable and in virtually f.g. nilpotent groups than in a general polycyclic group. In order to compute the MCSP we used the Cook reduction of Lemma 6.11 and therefore only needed to compute the CSP with a simultaneous determination of centralizers. One may use these results to find a better solution to the MCSP in general polycyclic groups.

Precise complexity estimations for the different group-based problems presented in Section 6.1 are very important to ensure correctness and security of the presented cryptographic scheme in Chapter 7. It is not sufficient to use experimental results as a base for its presumed correctness and security. For instance, we need the collection algorithm to work in polynomial-time so that the AAG protocol is correct. Right now, we have no proof that this is the case for infinite polycyclic groups.

To gain confidence in the security of the AAG protocol we need a hardness reduction to a well studied group problem which is assumed to be hard to compute. More precisely, we have to find a group-based problem such that a successful PPT-attacker against the AAG protocol can also compute the group-based problem efficiently.

At present, we only have a reduction in the other direction: a PTM which computes the SR-MCSP can be used to break the AAG protocol in polynomial-time. However, it could be possible that there are other ways to attack the AAG protocol. This thesis made a contribution to the process of this point becoming apparent.

Furthermore, it is important to find a class of polycyclic groups which is suitable for cryptography and which, in addition, possesses a precise construction for implementation purposes. To be more precise, we are looking for polycyclic groups which at least fulfill the following properties:

1. The group is non-virtually nilpotent, thus has an exponential growth rate, as we explained in Section 7.3;
2. The group is of high Hirsch length, thus the LBA is not successful, as shown in Section 8.1;

3. The group is not of the semidirect product form of Chapter 4, thus the FBA is not successful, as outlined in Section 8.2;

4. The group exhibits a precise construction to implement it in a programming language, for instance in GAP.

Even if we are able to find such a class of polycyclic groups and reliable hardness reductions with provable complexity estimations it is an unanswered question if the proposed cryptosystems with polycyclic groups as platform groups are resistant to quantum algorithms.

Cryptographic schemes which are assumed to be secure against attacks by a quantum computer, i.e., a computer which is based on quantum physics principles, are referred to as *post-quantum cryptography.* Currently, quantum computers with a large computing capacity have not been constructed yet. Nevertheless, experts estimate that in 15 years from today the situation will be different. At present, there are several classes of cryptographic schemes which are believed to be quantum resistant, for example lattice-based or code-based cryptographic protocols. For further details see D.J. Bernstein's *Introduction to post-quantum cryptography* [Ber09].

It will be exciting to see, which candidates will stand the test once large quantum computers will be available. In the case of polycyclic group-based cryptography, we should at the very first find concrete complexity estimations for the collection algorithm and also problems related to them on which the AAG protocol could be based on. On the side of cryptanalysis it could be interesting to find better solutions to the CSP in arbitrary polycyclic groups.

A. GAP program code

A.1. Average time for the CSP in GAP

```
AverageTestCSP:= function(G,n)

        local i,result_conjugates,x,y,sum_time,start_time,
            end_time,total_time,conjugate,average_time;

        #sum_time stores the total time needed for all
            calculations
        sum_time:=0;
        #result_conjugates lists all calculated conjugating
            elements
        result_conjugates:=[0];
        #run CSP n times
        for i in [1..n] do
                #choose two random elements of G
                x:= Random(G);
                y:= Random(G);
                start_time:=Runtime();
                conjugate:= IsConjugate(G,y,x^(-1)*y*x);
                end_time:=Runtime();
                total_time:=end_time-start_time;
                #store the calculated conjugating elements
                result_conjugates[i]:=conjugate;
                sum_time:=sum_time+total_time;
        od;

        average_time:=sum_time/n;

        return [average_time,result_conjugates];
end;
```

A.2. Expanding the example polycyclic groups in GAP

```
#m=number of generators to add
#n=number of example from the method ExamplesOfSomePcpGroups(m)
    with n in 1 to 16
ExpandingExample:=function(n,m)
        local G,coll,collG,l,i,j,list,H,temp;
        G:=ExamplesOfSomePcpGroups(n);
        collG:=Collector(G);
        l:=NumberOfGenerators(collG);
        #new collector coll has the original l plus m new
            generators
        coll:=FromTheLeftCollector(m+l);
        #store the structure of G
        #first, copy the set of relative orders and the power
            rules
        list:=RelativeOrders(collG);
        for i in [1..l] do
                #only if the relative order is not infinity (
                    stored as 0)
                if list[i] <> 0 then
                        SetRelativeOrder(coll,i,list[i]);
                        temp:=GetPower(collG,i);
                        SetPower(coll,i,temp);
                fi;
        od;
        #second, copy the conjugate relations
        for i in [1..l] do
                for j in [1..l] do
                        #conjugate relations are only defined
                            for j<i
                        if j < i then
                                temp:=GetConjugate(collG,i,j);
                                SetConjugate(coll,i,j,temp);
                        fi;
                od;
        od;
        #update the collector and build the polycyclic group
            out of it
        UpdatePolycyclicCollector(coll);
        H:=PcpGroupByCollector(coll);
        return H;
end;
```

A.3. Average length difference in GAP

In Section 8.1.1 we defined for a given polycyclic group G with generating set $\{x_1, ..., x_l\}$ the length of g presented in its normal form $g = x_1^{e_1}...x_l^{e_l}$ as

$$l(g) := \sum_{i=1}^{l} |e_i|.$$

As a first step this program implements the function *LengthOfPcpElement(g)* which calculates the length of an element g, given in its normal form. For the Length-Based Attack it is crucial that the conjugation of an element increases its length, hence $l(g^{-1}hg) \gg l(h)$ for most g, h in G.

Furthermore, the function *AverageTestDiff(G, m)* is implemented. It gets as input a polycyclic group G and the number of tests m to calculate the average difference of $l(g^{-1}hg)$ and $l(h)$. To do so, we also implement the method *RandomElement(coll)* which calculates a random element from a collector of a polycyclic group. A collector is an object used for calculations in the GAP program that stores the structure of the underlying polycyclic group.

To stick with D. Garber et al. [GKL15] we construct elements of length between 10 and 13. Therefore, we randomly choose exponents e_i in the range $\{0, ..., 13/l + 1\}$ where l is the number of generators of G.

```
#==========first subroutine==========

#calculates the length of a PcpElement g, i.e., an element of a
    polycyclic group given in its normal form.
#the length is just the sum of the absolute values of the
    exponents of its normal form
LengthOfPcpElement:=function(g)
        local exp,l,length,i;
        exp:=Exponents(g);
        #l number of generators of the polycyclic group
        l:=NumberOfGenerators(Collector(g));
        length:=0;
        for i in [1..l] do
                length:=AbsoluteValue(exp[i])+length;
        od;
return length;
end;

#==========second subroutine==========

#calculates a random element with length in [10,13]
#coll is a Collector for a polycyclic group
```

```
RandomElement:=function(coll)
          local n,list,Randomlist,i,j,current,currentlength,l;
          #l number of generators of the polycyclic group
          l:=NumberOfGenerators(coll);
          #n number of maximal number of attempts to get one with
               length in [10,13]
          n:=100;
          #list out of which we draw the random exponents
          #therefore, round 13/l and add 1 (choice by experience)
          list:=[0..(Int(13/l)+1)];
                    for i in [1..n] do
                    Randomlist:=[];
                              for j in [1..l] do
                              #construct list of random exponents
                              Append(Randomlist,[Random(list)]);
                              od;
                    #construct the random element out of this
                         exponent list
                    current:=PcpElementByExponents(coll,Randomlist)
                         ;
                    od;
end;

#==========third subroutine==========

#calculates the difference of length(x^{-1}*y*x) and length(y)
    where x and y are constructed randomly
DiffLength:=function(coll)
          local x,y,lengthb,lengthconj,result;
          x:=RandomElement(coll);
          y:=RandomElement(coll);
          lengthb:=LengthOfPcpElement(y);
          lengthconj:=LengthOfPcpElement(x^-1*y*x);
          result:=[lengthb,lengthconj];
          return result;
end;

#==========main program==========

#calculates the average difference of length(x^{-1}*y*x) and
    length(y)
#m number of tests
AverageTestDiff:=function(G,m)
          local coll, l, SumLengthb, SumLengthconj, i, result,
            Averageb, Averageconj, result2;
          coll:=Collector(G);
          l:=NumberOfGenerators(coll);
          SumLengthb:=0;
          SumLengthconj:=0;
          for i in [1..m] do
                    result:=DiffLength(coll);
                    SumLengthb:=SumLengthb+result[1];
                    SumLengthconj:=SumLengthconj+result[2];
          od;
          Averageb:=SumLengthb/m;
          Averageconj:=SumLengthconj/m;
          result2:=[Averageb, Averageconj];
          return result2;
end;
```

References

[AAG99] Iris Anshel, Michael Anshel, and Dorian Goldfeld. An algebraic method for public-key cryptography. *Math. Res. Lett.*, 6(3-4):287–291, 1999.

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity.* Cambridge University Press, Cambridge, 2009. A modern approach.

[AE07] Björn Assmann and Bettina Eick. Testing polycyclicity of finitely generated rational matrix groups. *Math. Comp.*, 76(259):1669–1682, 2007.

[Ass05] Björn Assmann. Gap package polenta - polycyclic presentations for matrix groups. `https://www.gap-system.org/Packages/polenta.html`, 2005.

[BCRS91] Gilbert Baumslag, Frank B. Cannonito, Derek J. S. Robinson, and Dan Segal. The algorithmic theory of polycyclic-by-finite groups. *J. Algebra*, 142(1):118–149, 1991.

[Ber09] Daniel J. Bernstein. Introduction to post-quantum cryptography. In *Post-quantum cryptography*, pages 1–14. Springer, Berlin, 2009.

[BK00] James R. Beuerle and Luise-Charlotte Kappe. Infinite metacyclic groups and their non-abelian tensor squares. *Proc. Edinburgh Math. Soc. (2)*, 43(3):651–662, 2000.

[BN16] Kenneth R. Blaney and Andrey Nikolaev. A PTIME solution to the restricted conjugacy problem in generalized Heisenberg groups. *Groups Complex. Cryptol.*, 8(1):69–74, 2016.

[Can00] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000.

[Deh11] Max Dehn. Über unendliche diskontinuierliche Gruppen. *Math. Ann.*, 71(1):116–144, 1911.

[Dye80] Joan L. Dyer. Separating conjugates in amalgamated free products and HNN extensions. *J. Austral. Math. Soc. Ser. A*, 29(1):35–51, 1980.

[EK04] Bettina Eick and Delaram Kahrobaei. Polycyclic groups: a new platform for cryptology? *arXiv preprint math/0411077*, 2004.

[ENH04] Bettina Eick, Werner Nickel, and Max Horn. Gap package polycyclic - computation with polycyclic groups. `https://www.gap-system.org/Packages/polycyclic.html`, 2004.

[EO03] Bettina Eick and Gretchen Ostheimer. On the orbit-stabilizer problem for integral matrix actions of polycyclic groups. *Math. Comp.*, 72(243):1511–1529, 2003.

[For76] Edward Formanek. Conjugate separability in polycyclic groups. *J. Algebra*, 42(1):1–10, 1976.

[Geb02] Volker Gebhardt. Efficient collection in infinite polycyclic groups. *J. Symbolic Comput.*, 34(3):213–228, 2002.

[GK16] Jonathan Gryak and Delaram Kahrobaei. The status of polycyclic group-based cryptography: a survey and open problems. *Groups Complex. Cryptol.*, 8(2):171–186, 2016.

[GKL15] David Garber, Delaram Kahrobaei, and Ha T. Lam. Length-based attacks in polycyclic groups. *J. Math. Cryptol.*, 9(1):33–43, 2015.

[GKT$^+$05] David Garber, Shmuel Kaplan, Mina Teicher, Boaz Tsaban, and Uzi Vishne. Probabilistic solutions of equations in the braid group. *Adv. in Appl. Math.*, 35(3):323–334, 2005.

[GKT$^+$06] David Garber, Shmuel Kaplan, Mina Teicher, Boaz Tsaban, and Uzi Vishne. Length-based conjugacy search in the braid group. In *Algebraic methods in cryptography*, volume 418 of *Contemp. Math.*, pages 75–87. Amer. Math. Soc., Providence, RI, 2006.

[Gol01] Oded Goldreich. *Foundations of cryptography.* Cambridge University Press, Cambridge, 2001. Basic tools.

[Gro15] The GAP Group. Gap – groups, algorithms, and programming, version 4.7.9. `https://www.gap-system.org/Packages/polycyclic.html`, 2015.

[HEO05] Derek F. Holt, Bettina Eick, and Eamonn A. O'Brien. *Handbook of computational group theory.* Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, 2005.

[Hir38] Kurt A. Hirsch. On infinite soluble groups. I. *Proc. London Math. Soc. (2)*, 44(1):53–60, 1938.

[Hir52] Kurt A. Hirsch. On infinite soluble groups. IV. *J. London Math. Soc.*, 27:81–85, 1952.

[HJ13] Roger A. Horn and Charles R. Johnson. *Matrix analysis.* Cambridge University Press, Cambridge, second edition, 2013.

[HT03] James Hughes and Allen Tannenbaum. Length-based attacks for certain group based encryption rewriting systems. Cryptology ePrint Archive, Report 2003/102, 2003.

[KK06] Delaram Kahrobaei and Bilal Khan. Nis05-6: A non-commutative generalization of elgamal key exchange using polycyclic groups. In *Global Telecommunications Conference, 2006. GLOBECOM'06. IEEE*, pages 1–5. IEEE, 2006.

[KK12] Delaram Kahrobaei and Charalambos Koupparis. Non-commutative digital signatures. *Groups Complex. Cryptol.*, 4(2):377–384, 2012.

[KL14] Delaram Kahrobaei and Ha T. Lam. Heisenberg groups as platform for the aag key-exchange protocol. In *Network Protocols (ICNP), 2014 IEEE 22nd International Conference on Geoinformatics*, pages 660–664. IEEE, 2014.

[KL15] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography.* Chapman & Hall/CRC Cryptography and Network Security. CRC Press, Boca Raton, FL, second edition, 2015.

[KLC$^+$00] 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. In *Advances in cryptology—CRYPTO 2000 (Santa Barbara, CA)*, volume 1880 of *Lecture Notes in Comput. Sci.*, pages 166–183. Springer, Berlin, 2000.

[KU15] Matvei Kotov and Alexander Ushakov. Analysis of a certain polycyclic-group-based cryptosystem. *J. Math. Cryptol.*, 9(3):161–167, 2015.

[LGS90] Charles R. Leedham-Green and Leonard H. Soicher. Collection from the left and other strategies. *J. Symbolic Comput.*, 9(5-6):665–675, 1990. Computational group theory, Part 1.

[LNS84] R. Laue, J. Neubüser, and U. Schoenwaelder. Algorithms for finite soluble groups and the SOGOS system. In *Computational group theory (Durham, 1982)*, pages 105–135. Academic Press, London, 1984.

[Löh15] Clara Löh. Geometric group theory, an introduction. *Published online, mathematik.uniregensberg.de*, 2015.

[Mal40] Anatoliĭ I. Mal'cev. On isomorphic matrix representations of infinite groups. *Rec. Math. [Mat. Sbornik] N.S.*, 8 (50):405–422, 1940.

[Mal83] Anatoliĭ I. Mal'cev. On homomorphisms onto finite groups. *Trans. Amer. Math. Soc*, 119:67–79, 1983.

[MKS66] Wilhelm Magnus, Abraham Karrass, and Donald Solitar. *Combinatorial group theory: Presentations of groups in terms of generators and relations.* Interscience Publishers [John Wiley & Sons, Inc.], New York-London-Sydney, 1966.

[MN89] M. Mecky and J. Neubüser. Some remarks on the computation of conjugacy classes of soluble groups. *Bull. Austral. Math. Soc.*, 40(2):281–292, 1989.

[Mon17] Carmine Monetta. The conjugacy search problem for supersoluble groups. `http://www.groupsstandrews.org/2017/slides/Monetta-C.pdf`, 2017.

[MU07] Alex D. Myasnikov and Alexander Ushakov. Length based attack and braid groups: cryptanalysis of Anshel-Anshel-Goldfeld key exchange protocol. In *Public key cryptography—PKC 2007*, volume 4450 of *Lecture Notes in Comput. Sci.*, pages 76–88. Springer, Berlin, 2007.

[Ost99] Gretchen Ostheimer. Practical algorithms for polycyclic matrix groups. *J. Symbolic Comput.*, 28(3):361–379, 1999.

[Rem69] Vladimir N. Remeslennikov. Conjugacy in polycyclic groups. *Algebra i Logika*, 8:712–725, 1969.

[Rob72] Derek J. S. Robinson. *Finiteness conditions and generalized soluble groups. Part 1.* Springer-Verlag, New York-Berlin, 1972. Ergebnisse der Mathematik und ihrer Grenzgebiete, Band 62.

[Rob82] Derek J. S. Robinson. *A course in the theory of groups*, volume 80 of *Graduate Texts in Mathematics.* Springer-Verlag, New York-Berlin, 1982.

[Seg83] Dan Segal. *Polycyclic groups*, volume 82 of *Cambridge Tracts in Mathematics.* Cambridge University Press, Cambridge, 1983.

[Seg90] Dan Segal. Decidable properties of polycyclic groups. *Proc. London Math. Soc. (3)*, 61(3):497–528, 1990.

[Sho94] Peter W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science (Santa Fe, NM, 1994)*, pages 124–134. IEEE Comput. Soc. Press, Los Alamitos, CA, 1994.

[Sim94] Charles C. Sims. *Computation with finitely presented groups*, volume 48 of *Encyclopedia of Mathematics and its Applications.* Cambridge University Press, Cambridge, 1994.

[Sip13] Michael Sipser. *Introduction to the theory of computation.* Cengage Learning, [Andover], 3. ed., internat. ed. edition, 2013. Includes bibliographical references and index.

[Sti82] John Stillwell. The word problem and the isomorphism problem for groups. *Bull. Amer. Math. Soc. (N.S.)*, 6(1):33–56, 1982.

[SU06] Vladimir Shpilrain and Alexander Ushakov. The conjugacy search problem in public key cryptography: unnecessary and insufficient. *Appl. Algebra Engrg. Comm. Comput.*, 17(3-4):285–289, 2006.

[Tur37] Alan M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London mathematical society*, 2(1):230–265, 1937.

[Wol68] Joseph A. Wolf. Growth of finitely generated solvable groups and curvature of Riemanniann manifolds. *J. Differential Geometry*, 2:421–446, 1968.

Declaration

I hereby confirm that the present thesis is solely my own work and that if any text passages or diagrams from books, papers, the Web or other sources have been copied or in any other way used, all references have been acknowledged and fully cited.

Karlsruhe, 26th March 2018