POLYCYCLIC GROUPS: A NEW PLATFORM FOR CRYPTOLOGY?

BETTINA EICK AND DELARAM KAHROBAEI

Abstract. We propose a new cryptosystem based on polycyclic groups. The cryptosystem is based on the fact that the word problem can be solved effectively in polycyclic groups, while the known solutions to the conjugacy problem are far less efficient.

1. Introduction

Key exchange problems are of central interest in cryptology. The basic aim is that two people who can only communicate via an insecure channel want to find a common secret key. There are many approaches available which try to solve this problem. The most classical of these is perhaps the Diffie-Hellmann key exchange.

Key exchange methods are usually based on one-way functions; that is, functions which are easy to compute, while their inverses are difficult to determine. Here 'easy' and 'difficult' can mean that the complexities or the practicality of the methods are far away from each other; ideally, the one-way function has a polynomial complexity and its inverse has an exponential complexity.

Many of the known one-way functions have a common problem: it is often easy to find a one-way function with a polynomial complexity, but showing that there is no inverse function with similar complexity or practicality is usually the difficult part of the project, since the best inverse function might just not have been discovered yet. Hence it is of interest to investigate new one-way functions.

Here we propose a new one-way using similar ideas as in the Arithmetica key exchange [1]; that is, our one-way function is based on the word problem and the conjugacy problem in certain non-commutative groups.

The novelty in our approach is that we propose to use polycyclic groups as a basis for the protocols: These groups are a natural generalisation of cyclic groups, but they are much more complex in their structure than cyclic groups. Hence their algorithmic theory is more difficult and thus it seems promising to investigate classes of polycyclic groups as candidates to have a more substantial platform perhaps more secure.

2. The Diffie-Hellman key exchange

The Diffie-Hellman key exchange is both the original public-key idea and an important mechanism in current use. The practical point is that symmetric ciphers are generally

Date: 2004; submitted.

Key words and phrases. polycyclic groups, key exchange problems, word problem, conjugacy problem.

much faster than asymmetric ones (both hardware and software reasons), so the public-key cipher is merely used to set up a private key, a session key, intended to be used only for a single conversation. This protocol is based on the discrete logarithm, and it uses the relative difficulty of computing discrete logarithms. Originally, it was based on the $\mathbb{Z}/p$ discrete-logarithm, but it also generalizes to arbitrary finite fields, elliptic curves, or to any algebraic structure where logarithms make sense. We recall the classical idea of the Diffie-Hellman scheme as follows; see [8]: First, Alice, and Bob agree on a large prime m, and a primitive root g modulo m. These need not be kept secret and can be shared by a group of users. Alice chooses a large random integer x, privately computes $X = g^x \bmod m$, and sends X to Bob by the possible insecure channel. Meanwhile, Bob similarly chooses a large random integer y, privately computes $Y = g^y \bmod m$, and sends Y to Alice across the possible insecure channel. Then Alice privately computes $k = Y^x \bmod m$, and Bob symmetrically computes $k' = X^y \bmod m$. Then $k = k' \bmod m$, as

$$k' = X^y = (g^x)^y = g^{xy} = (g^y)^x = k \bmod m,$$

and thus $k = k'$ is the common secret of Alice and Bob. But no one else on the network can determine k, unless they can compute discrete logarithms.

3. Key exchange based on non-commutative groups

In 1999, the Arithmetica key exchange [1] was introduced by Anshel, Anshel and Goldfeld. It intends to achieve the same effect as the Diffie-Hellman key exchange; that is, it establishes a shared secret when the only communication possible is across an insecure channel. By contrast with Diffie-Hellman, it is based on combinatorial group theoretic properties and it uses non-commutative groups such as braid groups, making use of the difference of the complexity of the word and the conjugacy problem in such groups; see [11].

Since then, the area of 'group-theoretic cryptosystems' has been very active. For example, Bridson and Howie [4] have considered hyperbolic groups as a basis for a group-theoretic cryptosystem. Also, Gebhardt [9] describes a fast algorithm to check conjugacy in certain braid groups indicating that braid groups may not provide a secure basis for cryptosystems. A class of groups which provides a provably secure basis for the Arithmetica key exchange seems not to be known so far.

Below we recall two public key exchange methods: the Arithmetica key exchange [1] and a version of the Diffie-Hellman key exchange for non-commutative groups. We use these two key exchange methods as a basis for the remainder of this paper. We refer to [1] for their application to braid groups.

3.1. **Arithmetica key exchange.** Let G be a finitely generated group with solvable word problem. Let S and T be two finitely generated subgroups of G and let $\{s_1, \cdots, s_n\}$ and $\{t_1, \cdots, t_m\}$ be generating sets for S and T, respectively. Note that here x^y stands for $y^{-1}xy$. Now suppose two people, Alice and Bob, want to agree on a key. The group G, its subgroups S and T and their generators are public information. Then

a) Alice chooses a secret element $a \in S$ as a word in the generators $a = s_{i_1}^{a_1} \cdots s_{i_l}^{a_l}$ and publishes $t_1^a, \ldots, t_m^a$.
b) Bob chooses a secret element $b \in T$ as a word in the generators $b = t_{i_1}^{b_1} \cdots t_{i_h}^{b_h}$ and publishes $s_1^b, \ldots, s_n^b$.

Based on this setup, Alice and Bob can use $[a, b]$ as a shared secret. It is straightforward to observe that both, Alice and Bob, can compute $[a, b]$ readily, since

$$\begin{aligned} [a,b] &= (b^{-1})^a b = ((t_{i_1}^a)^{b_1} \cdots (t_{i_h}^a)^{b_h})^{-1} \cdot b \qquad \text{computable for Bob} \\ &= a^{-1}a^b = a^{-1} \cdot (s_{i_1}^b)^{a_1} \cdots (s_{i_l}^b)^{a_l} \qquad \text{computable for Alice} \end{aligned}$$

However, to determine $[a, b]$ based on the published data, we have to compute a and b, respectively. These can be determined by solving the conjugacy problem and finding an element which conjugates t_i on t_i^a for $1 \leq i \leq m$ and an element which conjugates s_j on s_j^b for $1 \leq j \leq n$. Thus the conjugacy problem can be used to break this cryptosystem.

3.2. **Non-commutative Diffie-Hellman key exchange.** Let G be a non-abelian group with solvable word problem. Let $u \in G$ and let S and T be two subgroups of G such that $[S, T] = \{1\}$. Suppose that two people, Alice and Bob, want to agree on a key. The group G, its element u and its subgroups S and T are public information. Then

a) Alice chooses a secret element $w \in S$ and publishes u^w.
b) Bob chooses a secret element $v \in T$ and publishes u^v.

If w and v commute, then Alice and Bob can use $u^{wv} = u^{vw}$ as a shared secret. This is straightforward to determine for Alice and Bob based on their secret data. The determination of the shared secret is less easy if only the public data is available. In this case the conjugacy problem can be used to determine w and v from u and u^w and u^v.

4. Polycyclic groups for cryptosystems

In this section we consider the use of polycyclic groups as a basis for the key exchange methods as described above. Recall that a group is called polycyclic if there exists a polycyclic series through the group; that is, a subnormal series of finite length with cyclic factors. There are two different natural representations for these groups which can be used for computations: polycyclic presentations and matrix groups over the integers. We consider these two representations in the following.

4.1. **Polycyclic presentations.** Every polycyclic group has a finite presentation which exhibits the polycyclic structure of the considered group: a polycyclic presentation of the form

$$\langle a_1, \ldots, a_n \mid a_j^{a_i} = w_{ij}, a_j^{a_i^{-1}} = v_{ij}, a_k^{r_k} = u_{kk} \text{ for } 1 \leq i < j \leq n \text{ and } k \in I \rangle$$

where $I \subseteq \{1, \ldots, n\}$ and $r_i \in \mathbb{N}$ if $i \in I$ and the right hand sides w_{ij}, v_{ij}, u_{jj} of the relations are words in the generators $a_{j+1}, \ldots, a_n$. Using induction, it is straightforward to show that every element in the group defined by this presentation can be written in the form $a_1^{e_1} \cdots a_n^{e_n}$ with $e_i \in \mathbb{Z}$ and $0 \leq e_i < r_i$ if $i \in I$.

A polycyclic presentation is called consistent if every element in the group defined by the presentation can be represented uniquely by a word of the form $a_1^{e_1} \cdots a_n^{e_n}$ with $e_i \in \mathbb{Z}$ and $0 \leq e_i < r_i$ if $i \in I$. In this case these words are called normal words. We note that every polycyclic group has a consistent polycyclic presentation and these presentations are frequently used as a basis for computations with polycyclic groups. We refer to [14] for background and a more detailed introduction to polycyclic presentations.

4.1.1. *The word problem.* The word problem in a consistent polycyclic presentation can be solved effectively using the so-called collection algorithm, see [14]. This algorithm computes the unique normal word for an arbitrary word in the generators. The basic idea of the collection algorithm is that it applies iteratedly the power and conjugate relations of the given presentation to subwords of a given word and thus it modifies the given word. The nature of the relations asserts that an iteration of this process will eventually produce a normal word. The efficiency of the collection algorithm depends critically on the sequence of chosen subwords which are processed. There are various strategies which have been investigated for this purpose. We refer to [12] and [10] for an analysis of strategies in finite polycyclic groups and in [9] for arbitrary polycyclic groups. The resulting complexities of the methods depend on the growth of the exponents e_j of generators g_{i_j} occurring in intermediate stages of the algorithm while processing the word $g_{i_1}^{e_1} \cdots g_{i_l}^{e_l}$. This growth can be bounded above if the considered group is finite. In infinite polycyclic groups there is the potential risk of an integer explosion inherent in the collection algorithm. In praxis, collection is known as an effective method to solve the word problem in consistent polycyclic presentations. The method is implemented in GAP [15] and MAGMA [3] and it has proved to be practical for finite and infinite polycyclic groups.

4.1.2. *The multiple conjugacy problem.* The Arithmetica key exchange can be broken if an effective algorithm to determine a multiple conjugating element a with $r_i^a = s_i$ for $1 \leq i \leq m$ can be found (knowing that such an element exists). We observe that this problem reduces to the single conjugacy problem as follows.

Let a_1 be an element with $r_1^{a_1} = s_1$ and let $G_1 = C_G(s_1)$. Then every element $a \in G$ with $r_1^a = s_1$ can be written as $a = a_1 c$ with $c \in G_1$.

By induction, suppose that an element a_j is given with $r_k^{a_j} = s_k$ for $1 \leq k \leq j$ and let $G_j = C_{G_{j-1}}(s_j) = C_G(s_1, \ldots, s_j)$. Then every element $a \in G$ with $r_i^a = s_i$ for all i is of the form $a = a_j c$ with $c \in G_j$. Compute $c \in G_j$ with $(r_{j+1}^{a_j})^c = s_{j+1}$ and, simultaneously, determine $G_{j+1} = C_{G_j}(s_{j+1})$. Now define $a_{j+1} = a_j c$ as the element for the next step. Iterating this process yields an element $a = a_m$ eventually.

As every centralizer G_j is a subgroup of G, an induced polycyclic presentation for G_j can be computed from a generating set. Thus the multiple conjugacy problem reduces to m applications of the single conjugacy problem with a simultaneous determination of the corresponding centralizers.

4.1.3. *The single conjugacy problem.* The non-commutative Diffie-Hellmann key exchange and the Arithmetica key exchange can be broken if an effective algorithm to determine a conjugating element $r^a = s$ for given $r, s \in G$ can be found. For the Arithmetica key exchange the additional computation of $C_G(s)$ is necessary.

In [2] has been proved that there exists an algorithm to compute such a conjugating element and such a centralizer in a polycyclic group. Later, in [7] another algorithm for this purpose has been described. The algorithm of [7] has been implemented in the Polycyclic package [6] based on GAP and Kant [5] and its performance has been investigated. It can be observed that the algorithm is practical on interesting examples, but its performance is far less good than the performance of the collection algorithm. In particular, in large polycyclic groups with a complex structure, the computation of conjugating elements is practically impossible while the word problem is still effectively solvable.

The algorithm described in [7] uses induction down along a normal series with elementary or free abelian factors of the considered group G. The steps corresponding to finite factors in the series can be solved by an application of an orbit-stabilizer algorithm as typical for finite groups.

The steps corresponding to infinite factors require:

a) Methods from representation theory such as the computation of submodule series,
b) Methods from algebraic number theory such as the computation of unit groups,
c) Methods from finite polycyclic groups such as the finite orbit-stabilizer algorithm.

A determination of the complexity of the conjugacy algorithm has not been attempted yet. However, the complexity of this algorithm could incorporate the complexity of the unit group computation in algebraic number fields and in this case it is going to be far away from the complexity of the word problem. This seems to suggest that polycyclic groups are a promising candidate for cryptosystems.

4.2. **Matrix representations.** Every polycyclic group can be described as a finitely generated subgroup of $GL(d, \mathbb{Z})$ for some $d \in \mathbb{N}$. In this setting, the word problem for a polycyclic group is solvable in polynomial time, as matrix multiplication for such groups is solvable in polynomial time.

The conjugacy problem has not been considered for polycyclic matrix groups so far. The only implemented method to solve the multiple or single conjugacy problem in such groups is by determining a polycyclic presentation and then applying the method of [7]. However, there is a possibility that linear methods can be used to compute conjugating elements or to improve the above approach, even though such methods have not been studied yet.

5. Examples

In this section we consider a few examples of polycyclic groups and we investigate the practicality of the collection algorithm and the conjugacy algorithm of [7]. This will yield the observation that not every class of polycyclic will be useful as a basis for a cryptosystem as outlined above, but there are classes which look promising.

5.1. **Some example computations.** Let $K = \mathbb{Q}[x]/(f)$ be an algebraic number field for a cyclotomic polynomial f_w, where w is a primitive r-th root of unity. Then $deg(f_w) = \varphi(n)$. The maximal order O of K is a ring whose additive group is isomorphic to $\mathbb{Z}^n$ and the unit group U of K is a finitely generated abelian group. The natural split extension $G(w) = O \rtimes U$ is a metabelian polycyclic group and we investigate a few examples of such groups and the practicality of their word and conjugacy problem in the following tables. For this purpose, we list the order r of w, the Hirsch length $h(G(w))$ of $G(w)$, the average time used for 100 applications of the collection algorithm on random words and the average time used for 100 applications of the conjugacy algorithm on random conjugates in such groups.

r	h(G(w))	coll	conj
3	2	0.00 sec	9.96 sec
4	2	0.00 sec	9.37 sec
7	6	0.01 sec	10.16 sec
11	14	0.05 sec	$>$ 100 hrs

For the prime 11, the result of a single conjugacy test could not be computed within one hour using the current methods. For primes larger than 11, the results of such timing are expected to be even worse.

5.2. **Nilpotent groups.** For finitely generated nilpotent groups there are various special methods known to compute with these groups. For example, the word problem in finitely generated nilpotent groups can be solved by evaluating polynomials as described in [13]. The conjugacy problem can be solved by an efficient methods as for example described in [14].

In this case the complexity and the practicality of the conjugacy problem is perhaps not as far away from the complexity of the word problem as in the case of an arbitrary polycyclic group.

References

[1] I. Anshel, M. Anshel, and D. Goldfeld. An algebraic method for public-key cryptography. *Math. Res. Lett.*, 6:287–291, 1999.

[2] G. Baumslag, F. B. Cannonito, D. J. S. Robinson, and D. Segal. The algorithmic theory of polycyclic-by-finite groups. *J. Alg.*, 142:118 – 149, 1991.

[3] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system I: The user language. *J. Symb. Comput.*, 24:235 – 265, 1997.

[4] M. R. Bridson and J. Howie. Conjugacy of finite subsets in hyperbolic groups. *submitted for publication*, pages 1–30, 2003.

[5] M. Daberkow, C.Fieker, J. Klüners, M. Pohst, K.Roegner, and K. Wildanger. Kant V4. *J. Symb. Comput.*, 24:267 – 283, 1997.

[6] B. Eick and W. Nickel. Polycyclic - computing with polycyclic groups, 2000. A GAP 4 package.

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

[8] P. Garrett. *Making, Breaking Codes: Introduction to Cryptology*. Pearson Education, 2000.

[9] V. Gebhardt. A new approach to the conjugacy problem in garside groups. *J. Symb. Comput.*, 2003.

[10] B. Höfling. Efficient multiplication algorithms for finite polycyclic groups. *Submitted*, 2004.
[11] J. L. Joan S. Birman, K.H. Ko. A new approach to the word and conjugacy problems in the braid groups. *http://xxx.lanl.gov/abs/math.GT/9712211*, pages 1–31, 1998.
[12] C. Leedham-Green and L. Soicher. Collection from the left and other strategies. *J. Symb. Comput.*, 9:665 – 675, 1990.
[13] C. Leedham-Green and L. Soicher. Symbolic collection using deep thought. *LMS J. Comput. Math.*, 1:9–24, 1998.
[14] C. C. Sims. *Computation with finitely presented groups*. Encyclopedia of Mathematics and its Applications, 48, Cambridge University Press, 1994.
[15] The GAP Group. GAP – Groups, Algorithms and Programming, 2000.

Bettina Eick, Institut Computational Mathematics, TU Braunschweig, Pockelsstr. 14, 38106 Braunschweig, Germany
E-mail address: beick@tu-bs.de

Delaram Kahrobaei, Mathematical Institute, University of St Andrews, North Haugh, St Andrews, Fife KY16 9SS Scotland, UK
E-mail address: delaram.kahrobaei@st-andrews.ac.uk
URL: http://www-groups.mcs.st-and.ac.uk/∼delaram/