Novel Protocols in Group-based Cryptography

Serge Horbach
IMAPP – Radboud University
serge.horbach@student.ru.nl

ABSTRACT

In this research we propose new protocols in group-based cryptography contributing to the research of finding novel cryptographic systems that are secure against quantum computers. The protocols we introduce either employ different one-way functions or different groups then the currently available protocols in group-based cryptography. Thereby, we avoid the successful attacks that threaten the contemporary existing group-based cryptosystems.

Keywords: conjugacy, cryptography, groups, post-quantum, protocol, subgroups.

INTRODUCTION

Traditionally, cryptography is the science of writing in secret code. Its main objective is to enable communication between two or more parties without eavesdroppers being able to intercept this communication. Presently, there is an increasing demand for secure cryptosystems due to the enormous increase of applications like internet shopping or electronic financial transfers.

The basic idea of cryptography is the use of a so-called one-way-function or trapdoor-function: an action which is easy to perform but hard to invert, except if one is in the possession of some 'extra' knowledge. In this case it should again be easy to invert to process.

Over the past decades, attempts have been made to create secure cryptographic protocols, based on several mathematical structures. From these, the protocols based on number theoretic issues have been most widely used and form the core of nearly all contemporarily used cryptographic protocols.

In our research we have attempted to set up protocols based on the structure of non-abelian groups and thereby contribute to the research called *group-based cryptography* ([7]).

A group is a mathematical structure consisting of a set of objects and an operation on these objects, which is invertible. Examples of groups used in our research are groups of matrices, and permutation groups.

We have designed two new protocols based on group theoretic issues. In one of the protocols, the novelty lies in the introduction of a new type of one-way-function, conjugacy of subgroups, while in the other protocol we use a standard one-way-function but apply it in a class of groups which has not been used for cryptographic purposes before. For the sake of clarity and brevity we will only describe the first of these protocols in this paper, thereby still covering most of the new ideas.

CRYPTOGRAPHY

As mentioned, the basic idea of cryptography is the use of a one-way-function: some action we can easily perform but that we cannot invert unless we have some extra knowledge. Classical examples of this include the permutation of letters of the alphabet to encrypt some written text. Indeed if we have a message and we are given a permutation of the alphabet (for example we write 'b' instead of 'a', 'c' instead of 'b' etc.) then we can easily perform the permutation on the message. However, unless we know what permutation is used, it is difficult (at least it was in former times) to find the original message when we are only given the encrypted piece of text.

Contemporary cryptography consists mainly of so-called public-key (or asymmetric) cryptography. This is a form of cryptography in which two distinct keys are required: a secret (or private) key and a public key. Although different, these two keys are linked to each other in such a way that it is possible to use the public key to encrypt a message or to verify a digital signature, whereas the private key is used to decrypt a message or to create a digital signature. This form of cryptography was introduced in 1976 by Diffie and Hellman ([7]).

The use of public-key cryptography was initiated by a new cryptographic protocol called a key-exchange protocol. As its name suggests, this protocol is used to exchange secret keys between two or more parties. Those keys can subsequently be used in encryption protocols or signature verifying protocols. The protocol suggested in this paper is a novel example of a key-exchange protocol.

Over the past decades, cryptography has turned its attention to abstract mathematics in order to provide security for its protocols. Most notably, the difficulty to factorize large numbers into their prime factors has been at the core of many cryptographic applications for over fifty years. However, there is an ever increasing demand for cryptographic applications and most notably the possible foundation of a *quantum computer*, from which it is known that it breaks most of our current cryptographic protocols, forms a serious threat to contemporary cryptographic protocols ([9]). Therefore, the introduction of novel cryptographic protocols, based on different mathematical structures and one-way-functions, is highly desirable ([7]).

GROUP-BASED CRYPTOGRAPHY

As a response to this call for novel cryptographic protocols, several papers have been published, starting with [1] and [4], that introduce the concept of group-based cryptography. These protocols are based on the algebraic structure of a group and use operations in a group as their one-way-function.

A *group* in mathematics is defined as a set of objects in which two objects can be composed to obtain a third one. In addition, for every element there exists an *inverse element*, meaning that we can 'undo' any operation that we have performed. An example of a group is the integer numbers with the operation of addition. Indeed, if we take the set of all integers: $\{\dots, -2, -1, 0, 1, 2, 3, \dots\}$ and we add two of them, we find another integer. Moreover, for any integer k there exists the integer $-k$ which has the property that first adding k and then $-k$ results in no change at all. This shows that $-k$ is the inverse element of k. Hence the integers form a group.

Important to note here is that the above group is said to be *commutative,* since for any integers a and b we have that $a + b = b + a$, loosely speaking: it does not matter in which order we compose the group elements. However, in many groups this order does matter. Groups in which this is the case will be called *non-abelian groups* and it is precisely those groups that group-based cryptography is interested in.

Most commonly the operation in groups is written as multiplication (instead of addition like in the above example). Hence, for group elements g and h we write $g \cdot h = gh$ for applying the composition of these elements. We stress that in non-abelian groups, in general we have $gh \neq hg$. This will mainly be used in a process called *conjugation* of elements. We say that the product ghg^{-1} is the conjugate of h by g, where g^{-1} denotes the inverse of the element g. Note that conjugate elements coincide precisely when $gh = hg$, since in this case $ghg^{-1} = hgg^{-1} = h$, but otherwise they are different elements.

In the research on group-based cryptography, the so-called *conjugacy search problem* has played a major role ([1], [4]). In an instance of the conjugacy search problem one is given two elements h, k of a given group G, which are conjugate by some unknown group element g, i.e. we have $h = gkg^{-1}$. The problem then is to find an element x such that $h = xkx^{-1}$. Note that the unknown element g will do the job, but there may be different group elements that also conjugate k to h.

In the first articles on group-based cryptography, conjugacy was used as a one-way function, i.e. the security of the protocols relied on the difficulty of the conjugacy search problem ([1], [4]). However, powerful attacks based on some notion of the length of group elements (the length is defined as the minimum number of generators needed to write the element as a product of generators) have shown that the conjugacy search problem provides a much lower level of security than required ([7]). Therefore, we suggest a protocol using a novel one-way function that uses conjugacy in a different way.

A NEW CRYPTOGRAPHIC PROTOCOL

In this section we will introduce a key-exchange protocol based on the computational difficulty of the subgroup conjugacy search problem in permutation groups. In order to do so, we will discuss the mathematical background of the protocol.

We will use a group of permutations as our platform group. Each element of this group is a function permuting the numbers 1 to n for some chosen integer n. The operation in this group is the composition of mappings, the inverse of any element is the reverse permutation moving every number back to its original position. We use the notation S_n for the group of permutations of the numbers 1 to n and call n the *degree* of S_n.

Furthermore, our protocol will make essential use of the notion of a *subgroup*: Given some group G we will call a subset H of G a subgroup of G if H is a group in its own right. For example the set of even numbers forms a subgroup of the integers. If H is a subgroup of G we write $H \leq G$.

The protocol is based on the subgroup conjugacy search problem, which extends the notion of conjugacy from group elements to (sub-)groups. We denote by $\sigma G \sigma^{-1}$ the group of elements of the form $\sigma g \sigma^{-1}$ where g runs over all elements of G and call this the conjugate group of G by σ.

We now give a description of the protocol in which two parties, usually called Alice and Bob, create a common secret key:

- **Public information**: Positive integers n and m, with n even and $m < 2^{n/2}$. Furthermore, a subgroup $G \leq S_{2n}$ (given by generators); and two commuting subgroups $A \leq S_{2n}$ and $B \leq S_{2n}$ (i.e. for every element a in A and every b in B we demand that $ab = ba$).
- **Private information**:
  - Alice: an element π in A.
  - Bob: an element σ in B
- **The protocol**:
  - Alice computes $H = \pi G \pi^{-1}$ and sends it to Bob.
  - Bob computes $K = \sigma G \sigma^{-1}$ and sends it to Alice.
  - Alice and Bob both compute $L = \pi \sigma G \sigma^{-1} \pi^{-1}$. The common secret key is the m-th element in L with respect to the lexicographic order of permutations.

Note that Alice and Bob can both compute L because π and σ commute, Alice computes it as $L = \pi \sigma G \sigma^{-1} \pi^{-1} = \pi K \pi^{-1}$ and likewise Bob computes it as $L = \sigma \pi G \pi^{-1} \sigma^{-1} = \sigma H \sigma^{-1}$.

ANALYSIS

For the above protocol we make some remarks on its complexity and on possible ways of attacking the protocol. Before that, we comment on the choices that have to be made in the protocol:

- For the public key G we advise to choose G as an elementary abelian 2-subgroup, meaning that any element has order 2 (performing twice the same permutation yields the identity map) and elements within G commute with each other. We recommend this choice because any knowledge on the structure of elements in G which is invariant under conjugation will also show up in the elements of any conjugate subgroup, thus also in L. One of these structural properties are the *orbits* of G on the numbers between 1 and $2n$. We say that any two numbers x and y are in the same orbit under G if there exists a permutation g in G such that g maps x to y. Under conjugation this property is preserved in the sense that the number of orbits and their sizes are equal under G and under H. For an elementary abelian 2-subgroup all orbits have size 2, therefore no additional information is leaked to an attacker. Moreover, in this way we maximize the number of orbits under G which, among others, makes it difficult to construct a so-called Base and Strong Generating Set (BSGS) for G which is a standard tool for making computations in permutation groups efficient and would almost certainly be required for an attack on the protocol ([6]). Lastly, we recommend that G has roughly $\sqrt{|S_{2n}|}$ elements (the square root of the number of elements in S_{2n}) to ensure that both G and its coset space (a partition of the elements of S_{2n} in parts of the same size as G) are large and thereby averting brute force and quotient attacks ([7]).
- For the public keys A and B we recommend on choosing them as large as possible. Indeed, the secret keys are chosen from the subgroups A and B. Hence, choosing small subgroups would enable brute force attacks. One possibility is to choose A to be the set of all permutations that fix the numbers $n+1$ up to $2n$ while permuting the numbers 1 to n amongst themselves. Similarly B is the set of permutations fixing the first n numbers while permuting the last n ones. In this way elements from A commute with those of B and both A and B have $n!$ elements. Hence the key space grows very fast with increasing n.

Secondly, we comment on the method that Alice and Bob employ to find the m-th element in lexicographic order from L. In order to do this we transform the abelian 2-subgroup L into a linear code over the field with two elements (i.e. a binary code). For the technical construction to do this we refer to [3]. Important in this construction is that it can be done very efficiently and that it preserves the lexicographic order. The m-th element in the linear code can be computed very quickly (without actually enumerating all the elements) by using the binary expansion of m. Finally, the m-th element of the linear code is translated back to the corresponding element of L. ([3]). Note that the assumption $m < 2^{n/2}$ is used in the fact that if G is constructed as above, it contains precisely $2^{n/2}$ elements.

Attacks and complexity

In this section we will comment on the general difficulty of breaking the protocol. The above protocol is based on the difficulty of finding a conjugating element between two subgroups. Indeed, if one would succeed in finding the (secret) element π from the public information G and the transmitted information H then one would have knowledge of π and H and therefore would be able to compute L in the same way as Alice does, and thus find the secret key g.

However, it was shown in [6] that the problem of finding a conjugating element between two permutation subgroups in general is hard to solve, meaning that the time to find a solution grows exponentially with the parameter n.

To execute the protocol we saw that essential use has to be made of the possibility to transform the subgroup L into a binary linear code. However, this construction also facilitates a possible attack on the protocol. In the same way as L is transformed into a linear code, one could also transform the subgroups G and H to binary linear codes. By construction, these linear codes will be isomorphic ([3]). Finding an isomorphism between these linear codes will directly provide a conjugating element between the corresponding subgroups ([3]), hence the problem of finding a conjugating element between the subgroups G and H translates into the problem of finding an isomorphism between binary linear codes. This problem has been studied under the heading *code-equivalence problem* and has been part of the research on post-quantum cryptography - the research on cryptographic protocols that are resistant against attacks by quantum computers. It is believed that the code-equivalence problem is hard and in fact may even provide a platform for post-quantum cryptography ([8], [9]). Therefore, we conclude that this attack does not pose an actual threat to our protocol and in fact provides some decent ground for security assumptions.

As a final, but crucial point we note that the length-based attacks that were used to attack protocols based on the conjugacy search problem cannot be employed to attack our protocol. Indeed, the use of subgroups rather than elements was motivated by the fact that it makes these attacks infeasible.

Recommendation on parameters

In order to get some insight in the complexity of the key-exchange protocol and the complexity of the above mentioned attack on the protocol via binary linear codes, we ran some computer experiments. Results of these experiments are presented in table 1, '*Experimental data'*, below. The first column of the table indicates the value of n that has been used (and hence determines in what group S_{2n} we are working). The second and third columns present the time it takes to respectively generate a secret key, i.e. to perform all steps of the protocol, and the time it takes to break to protocol by an attack using linear codes. All timings are given in seconds and have been averaged over 100 runs for the first four rows and over ten runs for the last row. The last column provides the interval from the shortest to the longest time it took to break the protocol. Experiments were run using a 3.10 GHz computer and the V2.19-2 version of the Computer Algebra system MAGMA ([2]).

n	**Key generation (s)**	**Breaking (s)**	**Interval (s)**
80	0.005	0.250	[0.200 , 0.300]
100	0.010	14.16	[12.11 , 21.06]
110	0.010	72.97	[64.18 , 123.5]
120	0.015	303.46	[268.90 , 596.8]
140	0.020	8683.2	[8092.1 , 9316.8]

Table 1: Experimental data

From the table we conclude that key generation can be done very rapidly using the above mentioned protocol. Even for values of $n = 500$ key exchange can take place in less than a second.

Some additional experiments were run in order to be able to extrapolate the duration of an attack for larger values of n ([3]). From these results we find the same trend as can be spotted in the table: the duration of an attack grows exponentially with n, increasing n by 1 increases the breaking time by a factor of $k \geq 1.2$. Using these results we recommend values about 200 for n. For these values key exchange can be done in less than 0.1 seconds, while using the above attack will cost over a million years of computation time (on a single computer). We note that these parameter sizes are well chosen according to the standards in [5].

Another interesting result that is shown in the table is that the variation of the duration of the breaking algorithm is actually quite small. This is an important requirement for using the above protocol in practical applications, since it indicates that breaking the protocol by a 'lucky strike' is very unlikely.

CONCLUSION

In our research project we have suggested a novel protocol in group-based cryptography that employs the computational difficulty of the subgroup conjugacy search problem. With this protocol we intended to avoid the length-based attacks that were very effective in breaking the existing protocols based on the conjugacy search problem.

For this protocol we have thoroughly investigated the computational difficulty in order to find hard instances of the subgroup conjugacy search problem. By means of this analysis we recommend on employing permutation groups of degree about 400 as platform group and to use elementary abelian 2-subgroups as public key.

We showed that one of the attacks against our protocol deals with the problem of finding an equivalence between binary codes, a problem that has been extensively studied in the field of post-quantum cryptography and has the potential of being secure against quantum-computer attacks ([8], [9]). Therefore, our protocol may even be relevant for the post-quantum era. We note that a rigorous mathematical proof of the security of the system is (as with most cryptographic systems) very challenging. However, the results of the above experiments using state-of-the-art attack methods and the recommendations in ([8], [9]) serve as good indications for the security.

In our research we moreover took a closer look at employing different groups as platform group for our protocol and suggest that polycyclic groups are worth considering ([3]). More importantly, we introduced another novel protocol based on the computational difficulty of the decomposition search problem. This problem is related to the conjugacy search problem, for given group elements g and $h = agb$ it asks to reconstruct a and b from g and h. In groups for which no useful length function exists this is a hard problem and we suggest to use matrix groups like $SL(4, p)$ for prime numbers p as platform groups ([3]). This second protocol also turned out to be very robust against (standard) attacks.

ROLE OF THE STUDENT

For this research I have been working under the supervision of Dr. Bernd Souvignier (Radboud University) and I had some valuable discussion with Prof. Derek Holt (University of Warwick). Valuable insights have been gained during discussions with both supervisors, though the initiatives and ideas for the results of the project, as well as working them out and writing the report came from the side of the student.

REFERENCES

1. Anshel I., Anshel M., Goldfeld D., '*An Algebraic Method for Public Key Cryptography'*. Math.Res.Lett, vol. 6, Springer Verlag, pp 287-291, 1999.
2. Bosma W. and Cannon J., '*MAGMA Handbook'*, Sydney, 1993.
3. Horbach S., *'Group-based Cryptography',* Bachelor Thesis, Radboud University, Nijmegen, 2014.
4. Ko K., Lee J. et al.: '*New Public-key Cryptography using BraidGroups'*. Advances in Cryptology - CRYPTO 2000, Lecture Notes in Computer Science, Springer, vol. 1880, pp 166 - 183, 2000.
5. Lenstra A. and Verheul E., *'Selecting Cryptographic Key Sizes'*. Public Key cryptography, Lecture Notes in Computer Science, vol. 1751, Springer-Verlag, pp 446 - 465, 2000.
6. Luks E.M., *'Permutation Groups and Polynomial-Time Computation'*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 11, pp 139 - 175, 1993.
7. Myasnikov A., Shpilrain V., Ushakov A., *'Group-based cryptography'*, Advanced Courses in Mathematics, CRM Barcelona, 2007.
8. Overbeck R. and Sendrier N., *'Code-based cryptography'*. Post-Quantum Cryptography 2009, Springer, pp 95 - 145, 2009.
9. Sendrier N. and Simos D.E., *'The hardness of code equivalence over Fq and its applications to Code-Based cryptography'*. Post-Quantum Cryptography, Lecture Notes in computer Science, vol. 7932, Springer-Verlag, pp. 203 - 216, 2013