Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

$\mu\Pi\lambda\forall$

Μεταπτυχιακό πρόγραμμα στη Λογική και Θεωρία Αλγορίθμων και Υπολογισμού

Διπλωματική εργασία

Artin groups for the Commuting Action Key Exchange platform

Τσομπλεκτζόγλου Βενέδικτος, 2011

Τριμελής επιτροπή:

Ευάγγελος Ράπτης, Καθηγητής τμ. Μαθηματικών (Επιβλέπων)
Δημήτριος Θηλυκός, Αναπ. Καθηγητής τμ. Μαθηματικών
Κωσταντίνος Δημητρακόπουλος, Καθηγητής, τμ. Μεθοδολογίας, Ιστορίας και Θεωρίας της Επιστήμης

Contents

Introduction

The following work is mainly based on a publication of Vladimir Shpilrain and Gabriel Zapata titled “Combinatorial Group theory and Public Key Cryptography” [1] (which will hence be abbreviated into CGT-PKC). In this publication the authors provide us with a review of the state of algebraic public key cryptography and propose a general protocol that allows two parties to securely compute a common private key over an insecure channel. The original idea of this dissertation was to develop a program that would implement the protocol using the Artin groups of extra large type as a platform, in order to compare it with other key exchange schemes in terms of efficiency and security. This specific platform was chosen as the authors of CGT-PKC used it themselves as an example. However this goal was abandoned since the first implementation was observably lacking real security. This lead to a change of direction; thus the new goal became to illustrate some of the weaknesses of the protocol that are common in algebraic cryptography and to investigate their causes.

At first, we will look into some necessary background, although a proper introduction to algebraic cryptography should be sought elsewhere. Then a presentation of the key exchange scheme will be given, with focus on the more specific case of Artin groups as platform. An analysis of the implementation will follow, and a general attack technique will be given. Also we will examine a subtle but crucial issue that concerns every algebraic key exchange protocol: how to use an established secret group element to secure subsequent communications. Looking at the literature one may conclude that this problem has not received the attention that it deserves.

An appendix with some simple examples is included. Though these are not actual test runs -that would require too much space- they will hopefully help to better illustrate the methods described.

I would like to thank my professors at MPLA, most notably prof. Evangelos Raptis and assoc. prof. Dimitrios Thilikos for their support and the very interesting courses they taught. Without them I would have neither the ability nor the motivation to author the present text, thus I feel obliged to dedicate it to them.

Commuting Action Key Exchange

Preliminaries in group theory

Let A be a finite set (which we will call the alphabet) and $A' = \{\overline{a} : a \in A\}$. Also let $S = (A \cup A')^*$ be the set of all words on $A \cup A'$, that is the set of all finite sequences of symbols in $A \cup A'$, including the empty sequence that will be hence referred to by the symbol e. Finally let $F(A) = (S, o)$, where o is the binary concatenation operator on S: $o(a,b) = ab$. Then it is easy to show that $F(A)$ is a group with e being its identity element. This group will be called the free group on A and A will be called a generating set for it. It is worth noting here that the specific alphabets used do not affect the essential structure of a free group; in fact $F(A)$ will be isomorphic to $F(B)$ if and only if A and B have the same cardinality. A useful notation needs to be introduced here: let A be an alphabet and R be a set of words in $F(A)$; then by $\langle A | R \rangle$ we will denote the quotient of the free group on A with the normal closure of R, $F(A) / <R>_{norm}$.

Free groups are of natural interest since they are the most "general" group type: every group can be the homomorphic image of a free group. In fact if R is the kernel of such a homomorphism, we can obtain a presentation for said group in the form of $\langle A | R \rangle$. Sometimes it is convenient to include in R not only words but also word equalities in the form $w_1 = w_2$. These are to be understood as another way of writing $w_1 w_2^{-1}$ (this is in accordance with the spirit of our definition as the former equality is equivalent to $w_1 w_2^{-1} = e$ and words in R are equal to e in the quotient group).

Artin groups, also known as generalized braid groups, are the groups that admit a presentation of the form $\langle x_1, \cdots, x_n \mid < x_1 x_2 >^{m_{1,2}} = < x_2 x_1 >^{m_{2,1}}, \cdots \rangle$, where $m_{i,j} = m_{j,i}$, $m_{i,j} \in \{2, 3, \cdots, \infty\}$ and $< xy >^m$ is the alternating product of x and y of length m with the convention that $< xy >^\infty$ represents the empty word, e. An Artin group can be represented by a weighted graph whose vertices correspond to the generators $x_1, \cdots, x_n$ and for each relation $< x_i x_j >^{m_{i,j}} = < x_j x_i >^{m_{j,i}}$ where $m_{i,j} < \infty$ there is an edge connecting the i-th and j-th vertex with weight equal to $m_{i,j}$. This correspondence can be more clearly illustrated if we consider that the Coxeter matrix of the group can be seen as the adjacency matrix of the graph; this of course also works and in the reverse direction: from any graph with a symmetric adjacency matrix we can find a corresponding Artin group. A more specific class of groups that we will be concerned with is the Artin groups of extra large type, which are Artin groups that have relators of length at least 8 (or equivalently whose Coxeter matrix contains weights only equal to or greater than 4).

Readers interested in learning more about Artin groups can find in [4] a broad overview of the subject. Also for a comprehensive introduction to presentations in group theory, see [3]. Note that the definitions given here follow the aforementioned publications.

Key exchange basics

The purpose of a key exchange protocol is to allow two parties to securely exchange -or agree on- a key over an insecure public channel. Usually it needs to be in the form of a binary sequence as that would provide us with the most versatility in applications. The key is not necessarily of their choosing as many protocols are designed so that both users contribute to a joint calculation in a way that makes it impossible for either one to manipulate the process into resulting in a predetermined value. Generally, in practical applications, the ultimate goal of this exchange is to allow the parties to use a private-key cryptosystem to secure subsequent communications. Using a key exchange protocol followed by symmetric encryption has some advantages over the alternative of public key cryptography as the latter is significantly more demanding in computational resources than the former. Moreover a new key may be generated as often as desired, whereas in a public key scheme each party usually owns one key pair at a time and changing it would require updating the public key repository. But it is not possible to secure communication that is not transmitted in real time, that is with someone who is not actively participating in the protocol at the time the key is exchanged (thus using such a protocol to secure e-mail for example is impossible). It is possible to exchange keys and store them for later use, however this creates new security issues that need to be addressed.

The first such system was proposed by Whitfield Diffie and Martin Hellman in their historic for the field of cryptography paper in 1976. Although they recognized the influence of Ralph Merkle (whom they cite in their original paper) and despite a recent proposal of Martin Hellman to the contrary, the protocol was named "Diffie-Hellman key exchange". It is based on the difficulty of solving the discrete logarithm problem[*] and can also be used with minor modifications as a public key cryptosystem. Now there is a variety of key-exchange protocols and public key cryptosystems that are based on a number of different hard mathematical problems. Their applications include the negotiation of wireless communication keys (802.11i and newer protocols), secure sockets (SSL / TLS) and session keys used to secure internet transactions.

As is the case with all public-key cryptographic primitives, a key exchange protocol relies for its security on a one-way function. That is, a function whose inverse is infeasible to compute within reasonable time limits, while at same time is itself computable. Specifically, cryptographic one-way functions are computable functions for which there is no polynomial time algorithm that can find pre-images for them. In the case of Diffie-Hellman and ElGamal this was the exponentiation in a finite cyclic group (and its inverse, the harder problem, is the discreet logarithm in the same group).

One of the simplest models in which it is possible to prove the security of a key exchange protocol is the Dolev-Yao model [5]. Their model supposes that an attacker has complete control over the public channel: she can intercept, alter and create messages at will. Other models have been proposed, such as [6] or [7], the later being considered one of the strongest, in the sense that protocols that are provably secure in it are also secure in most other models. The models provide a set of assumptions under which one tries to prove that a protocol has the *indistinguishability property*,

[*] Actually it is based on the difficulty of the "decision Diffie-Hellman problem" which in turn can be reduced to the discreet logarithm problem, but the two problems are not equivalent.

defined as the opponent's ability to win a certain game[*] with a greater probability than someone who would answer randomly.

For our purposes it will suffice to use a far more relaxed notion of security. Our adversary will only need to be able to read the messages that are exchanged and he will try to recover the whole key. Since our intention is to show that the protocol when implemented with Artin groups is insecure, it is enough to do that in this simpler model.

The protocol[†]

In CGT-PKC, the authors define a *public-key cryptographic system* as a tuple (S, T, f, H, h) such that:

- S and T are computable algebraic structures, for example groups or semigroups.
- $f : S \times T \to S$ is a well-defined one-way function, that is a function such that:

For $w = w'$ in S, $\forall t \in T\ f(w, t) = f(w', t)$ in S, and

Given $f(w, t)$ and w it is infeasible to calculate t.

- H is a set of computable algebraic structures.
- $h : X \to Y$ is an action where X and Y are any one of the S, T or an element of H.

H and h are auxiliary and their use depends entirely on the protocol. The security of the system will mostly depend on the difficulty of finding a pre-image for f.

As the authors note, their protocol is based on a generalization of the *Discreet Logarithm Problem*. As is common in cryptographic literature, we will assume that Alice and Bob want to exchange a key over a public channel. They first need to agree on a system (S, T, f, H) as defined above, where H contains two subsets of T, A and B, such that $\forall a \in A, \forall b \in B\ ab = ba$. Those elements are of course public and they need not necessarily be agreed upon by both parties (Alice might publish them for everyone that wants to send her a secure message for example). The protocol proceeds as follows:

- A word $w \in S$ is made public.
- Alice chooses a private word $a \in A$ such that $f(w, a) \neq e$ and transmits $f(w, a) = wa$ to Bob.
- Bob chooses a private word $b \in B$ such that $f(w, b) \neq e$ and transmits $f(w, b) = wb$ to Alice.
- Alice calculates $f(wb, a) = wba$ and Bob calculates $f(wa, b) = wab$. Since $ab = ba$, $wab = wba$ and this will be their common key.

The Diffie-Hellman key exchange can be seen as an instance of CAKE where S is the multiplicative group Z_p^*, p being a prime number, T is its automorphism group and f is the action that applies an automorphism t on an element w. For this particular protocol A, B and H are redundant as T is an abelian group, so we can set $A = B = T$.

[*] The game challenges the player to determine the value of a single bit and its setup depends on the type of protocol tested

[†] The material in this and the next section is entirely drawn from CGT-PKC [1]

CAKE using Artin groups

The example provided by Shpilrain and Zapata uses Artin groups of extra large type as a platform for CAKE. The authors justify their choice because it is a class of groups that has varying properties and consequently it is harder to find efficient algorithms that would solve a specific problem for any member of this class. Also these groups are known to be automatic*, which implies that the word problem can be solved efficiently (more specifically it has been shown to be solvable in quadratic time [10]). This is necessary as in the final step of the key exchange the two parties will share knowledge of an element but they are not guaranteed to have arrived at the same presentation for it; therefore they will need to have a means of producing some common binary sequence from it. This will be discussed at a later time though, in the section '*Obtaining a useable key*'. In the case of Artin groups as a platform, the CAKE tuple (S, T, f, H) will consist of an Artin group of extra large type ($S = G$), the set of endomorphisms of G ($T = EndG$), the action of applying an endomorphism to an element ($f(w, a) = a(w)$) and the auxiliary set that will be simply the union of two subsets of the endomorphisms of G, selected so that any two elements from different subsets commute.

Specifically, in the first step a tree, Γ, is generated with the following properties: its root (call it a_r) has a degree of 2 and all other vertices have a degree of at most m. Each edge is then assigned a weight greater than or equal to 4 and the vertices are labeled. Now the adjacency matrix of the tree can be viewed as the Coxeter matrix of the platform group. The tree without the root vertex is partitioned in two parts, Γ_A and Γ_B; from these we obtain the two corresponding subgroups G_A and G_B. Because they act on different generators, elements from $EndG_A$ and $EndG_B$ commute with one another freely. This gives us an obvious choice for H, it will be simply $\{EndG_A \cup EndG_B\}$. For an example, see appendix A.

The group G, a word w in G and a generating set for each member of H are made public (the latter is not really required by the protocol; all we need is a way to generate random members of $EndG_A$ and $EndG_B$, which also could be accomplished by having the original tree Γ be public). The authors note that w needs to contain generators from both G_A and G_B, otherwise the act of some of the endomorphisms will be trivial.

Implementation issues

One of the first algorithmic problems encountered was the generation of the monoid of endomorphisms for the platform group. As the authors of CGT-PKC note, these endomorphisms correspond to the endomorphisms of the representative tree of the platform group. But this observation does not significantly reduce the difficulty of the problem, as there is no known efficient algorithm† that computes a generating set

* Roughly, a group G is said to be automatic (or to have an automatic structure) if there exists a finite state machine that can solve the word problem for that group, that is it recognizes the language $\{(w_1, w_2) : w_1 \equiv w_2 \; in \; G\}$. For more information, see [9].

† Of course what is meant here is that a thorough search did not produce any results on this problem. Even if there were though it would still be impractical to generate and transmit such a large amount of data for a real-time protocol.

for the endomorphisms' monoid of a particular graph. However, the full monoid is not essentially necessary for the protocol; all that is required is for both parties to efficiently compute a random element of $EndG$. This is a much easier problem which can be approached by faster, probabilistic, methods. More specifically, any algorithm designed to explore a fitness landscape -such as simulated annealing or a genetic algorithm- would do.

These methods, though they cannot guarantee a result, are very practical so long as the fitness landscape is smooth [8]. The space we will be searching is $\Gamma \to \Gamma$, the group of functions from the generating tree to itself. Assuming an ordering of its vertices, $V = \{v_1, v_2, \ldots, v_n\}$, we will represent each element as a string of fixed length n in the form of $(f(v_1), f(v_2), \ldots, f(v_n))$. It is trivial to prove that there is a one to one correspondence between such strings and mappings in $\Gamma \to \Gamma$. The smoothest possible landscapes are those where the fitness function is inversely proportional to the Hamming distance of each element from the nearest (if more than one exist) global maximum. Even though we cannot know those optimal strings, it is still possible to calculate that distance, because it is equal to the number of vertices we need to map differently in order for the mapping to preserve edges. This can be easily calculated to be $dist = \left|\{v \in V : \exists v' \in V \left((v, v') \in E \wedge (f(v), f(v')) \notin E\right)\}\right| - \left|\{(v, v') \in E : (f(v), f(v')) \notin E\}\right|$. Let then $fit = \frac{1}{1 + dist}$; we observe that *fit* requires $O(|E|)$ time to be calculated and is a suitable fitness function that creates a landscape without any strictly local (that is sub-optimal) maxima.

Although as we saw it is not necessary to compute a generating set for $EndG$ to use the CAKE protocol, it might be of use for a variation of the protocol where the tree that generated the Artin group is not public. In this variation, one of the parties in the exchange generates the group and then transforms the defining relators before making them public. From what we saw, both the number of dimensions and the number of different possible values for each dimension of the search space are equal to the number of generators in the representation of the group (vertices in the corresponding graph). On the other hand, the fitness function requires time proportional to the number of the defining relations of the group (edges in the corresponding graph). In this scenario we observe that if the group presentation is altered to use relators of length at most 3 then the numbers of generators and relators are increased by $\sum_{r \in R} |r| - 3$. Thus the other party cannot efficiently compute an endomorphism through an exploration algorithm as the search space vastly increases in size and even the cost of the fitness function will be multiplied. Thus, in that case, the problem of effectively calculating and even transmitting a generating set of $EndG$ resurfaces.

Supposing we have solved this problem though, the rest of the implementation is very straightforward. The protocol will also require a means of selecting random words in G, which can be done by appending a set number of pseudo-randomly selected generators. Note that the word we get by this procedure will usually be shorter than that number as it may not be freely reduced (that is, it might contain consecutive inverses). Applying an endomorphism to an element is also a simple matter of scanning the element's presentation and substituting each generator by its image through the endomorphism.

Analysis

Perhaps the most crucial issue with the use of Artin groups in CAKE is deciding how to represent (and transmit; it is not internal representation that we examine) the elements of the platform group. This choice could potentially undermine the whole exchange by giving clues to the opponent about the endomorphisms used by the communicating parties. As we will see, it is impossible to avoid giving away some information and in many cases it will prove to be easy to obtain the entire private keys. In the following discussion, let $\Gamma = \Gamma_A \cup \Gamma_B \cup \{root\}$ be the representative tree of the Artin group G (similarly partitioned in G_A and G_B), $w = (w_1, w_2, \ldots, w_k)$ be the randomly chosen public word and let f_A, f_B be the endomorphisms selected by each party.

Preliminaries

We can assume that the eavesdropper will have full knowledge of the tree that generated the platform group. Even if the tree is not public, it is possible to reconstruct it from the defining relators, using the following algorithm: First find those generators that appear in at most one relator; they are the leaf nodes of the corresponding tree. Then for each relator that contains a generator already placed in the tree, add a node for the other generator in the relation, if it is not already present in the tree, as we construct it from the leaves. Then we connect them with an edge weighted at half of the relator's length. This of course supposes that the group presentation will be the one derived directly from the starting graph. Is it possible for the communicating parties to use other presentations that do not give away the tree structure?

The authors of CGT-PKC propose the use of Tietze transformations to alter the group's presentation, but the implementation details may not allow it in practical applications. If both parties need to know the full chain of transformations used to arrive at the resulting presentation, then the opponent knows them too. This is because it is only possible to communicate them through the insecure channel. Thus it is possible for everyone to reverse them in order to obtain the original presentation - and therefore the tree - without any noticeable computational overhead. If, on the other hand, the party that computes the transformations only makes public the resulting presentation and not the transformation chain, then it becomes very difficult for the other party to compute an endomorphism as required by the protocol (see section '*Implementation issues'*). Thus, unless the problem of generating the $EndG$ is first solved, the transformation chains used to arrive at the new group presentation need to be public.

Even in the cases where the opponent will not be able to reconstruct the original tree, she will still be able to discover which generator belongs to each of the two subtrees. This is because both parties need this information in order to be able to select an endomorphism that will only act on their corresponding set of generators, even if this information is implicit. That is, when the representative tree is not public then at least a generating set for the monoid of endomorphisms for one of its two partitions, suppose for G_B, must be public. The opponent could then calculate the union of the domains and images of the functions in $EndG_B$, giving her most if not

all the generators in G_B. As we will later see (section '*A generalized attack scheme*'), this will be enough to compromise the security of the protocol.

Simple cases

In the naïve implementation we transmit the elements "as-is". That means the parties first agree on a mapping of characters to generators; this may be implicit, as the relators and the public word will also use this presentation, and one can consider that the generators are the characters - provided they are of a fixed length or otherwise have clearly defined boundaries. Since there is no reason to use a different internal representation, the elements can be transmitted exactly as they are stored and processed. This approach however has a major drawback: the opponent can deduce the key by simple inspection of the public elements. By comparing w to $f_A(w)$ it is trivial to observe the action of f_A on the generators that appear in w (which is all we need to know of f_A). Both words will have the same length and the ith character in the transmitted $f_A(w)$ will be the result of applying f_A to the ith character of w. And f_B can be compromised in exactly the same way.

Therefore, it is obvious that we need to transmit elements equal to $f_A(w)$ and $f_B(w)$ in G, but with different presentations. An obvious choice would be some kind of normal form for the elements, for example the smallest word according to the "short-lex" ordering among the different presentations for the element could be used. However, there is no known algorithm for a normal form in the Artin groups of extra large type; in fact part of the reason why the authors selected this class of groups is this very characteristic.

We can foil such simple analysis by freely reducing the transmitted elements and use the defining relations to alter their presentation. But this will create small problems for our adversary. By comparing the freely reduced $f_A(w)$ to the freely reduced w we will discover few cases where a simplification took place. This would require that w contains a subword of the form xy^{-1}, so that x and y are mapped by f_A to the same generator, an occurrence that can be expected with probability at most ¼ in the extreme case that the image of f_A consists of a single element. In realistic cases we can expect that number to be significantly smaller. Since most of the necessary information is preserved, the opponent only needs to make a few educated guesses about the position of those simplifications. In fact an adaptation of an algorithm that calculates the Levenshtein distance* (in particular, one that would ignore additions and would only increase the distance the first time a specific substitution was found) will discover the most probable positions of those changes in quadratic time with respect to the length of w. This will provide us with the mapping of the generators in w that were not deleted in $f_A(w)$, which is at worst the ¾ of them. Again, in practical applications this can be expected to be a much larger part of them, enough to compromise the function. The use of relators can easily be reversed: in the overwhelming majority of the cases we would be forced to substitute short subwords for longer ones because the length of the defining relations is big enough to make it very improbable that more than half of their length is found in a part of

* Also known as the edit distance, this is defined as the number of primitive operations that are needed to edit one word into another. The operations allowed are insertion, deletion and substitution.

$f_A(w)$. Thus Dehn's algorithm could quickly provide the original word - or a very close match. From there we can continue as per the previous case and recover the endomorphism used - or at least a significant part of it. See appendix B for an example of this procedure.

It is worthwhile to note that both of these ideas failed due to the randomness of w and in particular because certain structures (that is a subword of alternating generators such as those present in the defining relations) are very rare in a randomly chosen word. One might think that since w is public anyway, it might be better to generate it so that the endomorphisms will not be detected so easily. Aside from the obvious demerits of departing from randomness in a cryptographic setting, such a word would be ultimately impossible to generate. This is because its form has to depend upon the particular endomorphisms used, but obviously the communicating parties cannot inform each other of their choice. At most one of them could select an element suitable to her choice, but an eavesdropper would need only discover one of the endomorphisms, say f_B (supposing that the word was selected by whoever chose f_A), and then apply it to the public element $f_A(w)$ to acquire the shared key.

A generalized attack scheme

Still, there are other ways to alter the presentation of elements that may provide some diffusion and mask the correspondence of positions in the common (source) word with positions in the mapped words. For example, Tietze transformations could be used in a way that would allow us to obtain a presentation of the group where the relators have a maximum length of three generators. This will increase the probability that some non-trivial transformation (that is apart from simple deletions or insertions of subwords that are equal to the empty word) is applicable on the elements of the group. And of course there may be other methods that we have not considered, but they will all have something in common: elements will be represented as finite sequences of characters, the characters being the generators on which the endomorphisms act. It is of no value to have the endomorphisms act on the original set of generators as then there would be no way to apply them on the last step of the protocol. This, combined with the fact that no relators contain elements belonging to both of the two different partitions of the representative tree of the platform group make possible a more general attack that could provide a large part of the endomorphisms used (making the total recovery feasible even by brute force).

The idea is that in a random word there will be many occurrences of sequences that effectively isolate generators, allowing an eavesdropper to observe how the endomorphisms act on particular generators and with enough such occurrences the clues will suffice to infer the entire function. Recall that the group is partitioned in two parts and that generators from one part are not related to generators from the other. That is, generators from different parts are not present in the same defining relation. Thus sequences of the form $(x_1\ y_1\ x_2)$ where x_1 and x_2 belong to G_A and y_1 belongs to G_B (the same of course hold for the converse case) will not allow y_1 to be masked by a defining relation, since those are very difficult to cross such boundaries. In fact, only the two relations that use the generator corresponding to the root node of the tree could do that, and the chance of their being applicable in this specific way in a random word is negligible. Moreover such sequences occur very often: all we require is that after a specific generator there will be a generator from the other partition, which (supposing that the tree is partitioned in a balanced way) has a

probability of ½, and that after that there will be a generator from the first group, again with probability ½. This means that on any position in the word we have a ¼ chance of finding such a sequence. On the other hand, such sequences with the same generators in the boundaries are rare, in fact they have a probability of $\frac{1}{2n^2}$, where n is the number of generators in the group, so we can expect that there will be many unique occurrences of the form $(x_1\ y_1\ x_2)$. When an endomorphism that acts on G_B, f_B, is applied, the boundaries will not be affected (ignoring the aforementioned negligible possibility). Therefore it will be quite straightforward for the opponent to scan the transmitted element for a sequence $(x_1\ y_2\ x_2)$, and if found she will have a very probable candidate for the image of y_1 through f_B, namely y_2. And there are more schemata that can provide useful information once located in both elements, for example one could look for sequences of type $(x\ y\ y\ x)$ or $(x\ y\ x\ y\ x)$, where by x we mean elements from G_A and by y we mean elements from G_B. A simple example of this method can be found in appendix C.

The question then is "what can we do to prevent this?". Unfortunately, not much: we can either try to make it more difficult for our opponent to recognize those opportunities, or we can try to prevent them from being there in the first place.

The first option consists of using relations to substitute indicatory parts of the transmitted elements or altering the group presentation. As we have also seen in the previous section, relations will not offer much protection as their application can easily be reversed by Dehn's algorithm. The case of Tietze transformations is only a little more complicated: If they are used to alter the presentation of the transmitted elements only then they could be reversed by simply substituting the new generators using their respective relations until the element contains only the original ones. This will be possible as the new presentation for the group (which needs to be public for the other party to be able to carry the same procedure on his element) would contain the relations that are needed. If on the other hand they are used before the first step of the protocol, then our opponent could ignore them and carry on with her analysis since she will be able to identify which partition they belong to, as if the group had that presentation to start with.

The second option would mean that the public word is not chosen randomly, but will be more carefully constructed, favoring consecutive selections of same-partition generators. Again, though any departure from randomness could potentially create many opportunities for exploitation, this particular solution would cause even more serious problems. The adversary would note the blocks in the public element that are formed by generators of the same partition and then observe how the selected endomorphisms act on those. In effect she would use the same technique as before but instead of looking at generators, the focus would be on bigger blocks. Again, defining relations would not cross those new boundaries and the information gained about the endomorphisms would be enough. To illustrate, suppose $w = (X_1\ Y_1\ X_2\ Y_2\ \ldots)$ is the public element. Then $f_A(w) = (f_A(X_1)\ Y_1\ f_A(X_2)\ Y_2\ \ldots)$ and by the same token $f_B(w) = (X_1\ f_B(Y_1)\ X_2\ f_B(Y_2)\ \ldots)$. Observe now that though the endomorphisms are not known, the secret common key, $f_A(f_B(w))$, can be obtained as it is $(f_A(X_1)\ f_B(Y_1)\ f_A(X_2)\ f_B(Y_2)\ \ldots)$ and this information is easily obtainable, just by

inspecting in which of the two partitions the generators composing the two transmitted words belong to.

It is noteworthy that the relative positions of both of those block types within the public word do not change by the application of the endomorphisms for the protocol. This fact makes it even easier for the opponent to avoid mistakes during her analysis, despite our efforts to confuse it.

Parameters

We should not omit to examine how the parameters of the protocol affect its security. The parameters to be considered are the size of the public element, which is directly related to the size of the key, and the size of the tree that the parties use to create the group.

It is interesting to see that increasing the element size does not significantly enhance the security of the protocol. That is because the methods employed by our adversary require algorithms that run in at most quadratic time on the word length (in fact they consist mainly of pattern matching, which can be even faster). Moreover, using longer words will result in proportionally more structures that can be used to obtain clues on the endomorphisms, making success even more likely for the opponent.

The size of the tree (and therefore the number of generators in the corresponding group) has a less simple effect. Smaller trees will result in groups that allow the creation of fewer indicative structures, but the endomorphisms will also be smaller (that is their domains will consist of fewer generators) and thus fewer such structures will be needed to obtain the endomorphisms. And of course an exhaustive search when the clues do not suffice will be much faster. Conversely, larger trees result in groups that do not provide much diffusion and as we saw it is not possible to remedy this with proportionately large elements.

Because there are different techniques available to our adversary, each appropriate for another choice of parameters, it will be very difficult to strike a good balance between them. In the tests that were conducted the most difficult problems, though still tractable by the methods presented this far, were encountered for the larger groups that were tested (up to approximately 120 generators, by increasing further the number of generators the computation of an endomorphism became significantly slower), with a large pool of endomorphisms to choose from (this requires a more homogenous tree structure and little variation in weights between connected vertices) and words that were at least 500 generators long (much shorter than that usually resulted in very easy problems). The presentation for the group was altered through Tietze transformations to contain relators of length at most 4 using the least number of generators possible to this effect, but the public element was selected from the original presentation. Unfortunately, even in these cases it was possible to infer the endomorphisms; after a series of random sequence pairs were generated by the computer it was a matter of minutes to examine them and with a few educated guesses arrive at the functions.

Obtaining a useable key

One more consideration concerning the practical application not only of this particular key exchange protocol but of every public key exchange using combinatorial group theory is that once both parties have knowledge of the same

element in some group, that knowledge alone cannot be considered the key. That key invariably needs to be a binary sequence. Unless there is some well defined function that can produce such a sequence (and of course the same sequence for elements equal in the platform group) it cannot be used to secure subsequent communication. In cases where a normal form algorithm is feasible for the group in question, that function could be to convert the element in its normal form and then (in order to diffuse the information contained in it) use a hash algorithm on the resulting presentation. But for our particular platform no such algorithm is available, and if we put some effort at disguising the elements to foil simple attacks then the two parties are guaranteed to terminate the protocol with different presentations for the common element.

Therefore, for this reason, the authors of CGT-PKC proposed an extra step for their protocol following [2] in their approach: After having calculated the common element, let it be u, one of the parties selects a binary sequence $(b_1, b_2, \ldots, b_n)$ and transmits another sequence of elements $(u_1, u_2, \ldots, u_n)$ such that $u_i = u$ (in G) if and only if $b_i = 1$. Now if the word problem is solvable in our platform group, the other party can recover the secret sequence selected by the first by testing whether each u_i equals to her own presentation of u.

There are a few issues with this idea, such as how the sequence $(u_1, u_2, \ldots, u_n)$ is to be generated and of course how taxing will this generation and the subsequent recovery of the binary sequence be on our computational resources. Moreover the volume of transmitted information is increased by a factor of n, the length of the binary sequence. In real-time cryptography, where the need of key exchange usually occurs, these are very important issues that need to be addressed as the protocol would compete with others based on the way it solves them.

However we will not be concerned by these issues as there is a surprisingly simple attack that would yield this new secret key, provided that the attacker can solve the word problem without significantly more effort than the communicating parties. In this case (as well as in most cases where group theory is used in cryptographic settings) the group presentations are public. Therefore there is indeed no difference in an opponent's ability to solve the word problem compared to the ability of a legitimate user.

Our adversary will start comparing the transmitted elements amongst themselves until she finds a large set of equivalents; these elements will obviously be the ones that correspond to the bits that are equal to '1' in the sequence (b_i). The number of comparisons she will need to make is almost the same as the legitimate user: since the probability that some u_i equals another element, u_j, is negligible unless the corresponding b_i and b_j both equal '1', after an opponent has found a match she will proceed to test the other elements against either of them. Assuming the binary sequence is random, such a match is obtained after on average less than ten comparisons.

It should also be mentioned here that it is always risky to transmit elements equal to the established secret one, because this would usually place an attacker on equal footing with the communicating parties. Using a different presentation makes no difference as that is exactly the knowledge that a legitimate user of the protocol has.

Conclusion

As the authors of CGT-PKC themselves note, one of the difficult problems that algebraic cryptography faces is that of diffusion: in our case the lack of it was what gave away evidence that could be used to construct the private key of the protocol. As we saw 'generic' methods (in the sense that they can be used regardless of the particular group) such as Tietze transformations and the -usually random- use of defining relations to change the presentations of transferred words are insufficient. An opponent can either reverse them or ignore them. That last choice might seem counter-intuitive; after all the reason why we would use these transformations is to diffuse the information in the elements and that seems to be accomplished easier when the defining relations are short. We must remember here though that while between group presentations that utilize approximately the same number of generators the one with the shorter relations offers the most diffusion, the comparison is not as simple when in order to obtain the shorter relations we need to increase the number of generators. And these transformations add one new generator for every new relation they introduce. In short we have no reason to expect that a word chosen randomly from the group with an altered presentation would be easier to transform in a one-way manner (that is, without someone else being able to deduce our original choice) just because the relators are short.

The other major problem discussed here is that the secure transmission of elements of a group by themselves is not very useful in practical applications of cryptography as what we ultimately need is a shared secret in the form of a binary sequence. Despite a lot of effort to seek solutions to this issue in the bibliography, only one scheme was found and that was susceptible to an attack that would easily compromise it. This problem is not confined to the class of protocols discussed here, but perhaps may not be addressable in a uniform way.

What is interesting is that both of these problems can be avoided if our platform groups have a known normal form algorithm. The latter would be easily solved by requiring as the final step in the protocol that both parties calculate the normal form of the shared element and then use a hashing algorithm on the resulting form. The former is not as simple to overcome as there is no guarantee that the normal form will sufficiently diffuse a randomly generated element, but at least it will provide us with a clearer indication of the security of the protocol if it does. And because the normal form of an element might differ quite a lot from the representative we calculated it, the "fingerprinting" attack that we described earlier could be foiled.

References

1. Vladimir Shpilrain and Gabriel Zapata. "Combinatorial Group Theory and Public Key Cryptography." *Applicable Algebra in Engineering, Communication and Computing 17.3-4* (2006): 291-302.
2. Iris Anshel, Michael Anshel, and Dorian Goldfeld. "An Algebraic Method for Public Key Cryptography." *Mathematical Research Letters 6* (1999): 1-5.
3. Wilhelm Magnus, Abraham Karrass, and Donald Solitar. *Combinatorial Group Theory: Presentations of Groups in Terms of Generators and Relations*. Mineola, NY: Dover Publications, 2004.
4. K. I. Appel and P. E. Schupp. "Artin Groups and Infinite Coxeter Groups." *Inventiones Mathematicae 72.2* (1983): 201-20.
5. D. Dolev and A. Yao. "On the Security of Public Key Protocols." *IEEE Transactions on Information Theory 29.2* (1983): 198-208.
6. Anupam Datta, Ante Derek, John C. Mitchell, and Bogdan Warinschi. "Key Exchange Protocols: Security Definition, Proof Method and Applications." *19th IEEE Computer Security Foundations Workshop (CSFW 19)* (2006).
7. Ran Canetti and Hugo Krawczyk. "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels." *Eurocrypt (LNCS 2045)* (2001).
8. Terry Jones. *Evolutionary Algorithms, Fitness Landscapes and Search*. Diss. University of New Mexico, 1995.
9. S. M. Gersten and H. B. Short. "Small Cancellation Theory and Automatic Groups."*Inventiones Mathematicae* 102.1 (1990): 305-34.
10. D. Peifer, "Artin Groups of Extra-large Type Are Biautomatic." *Journal of Pure and Applied Algebra* 110.1 (1996): 15-56.

Bibliography

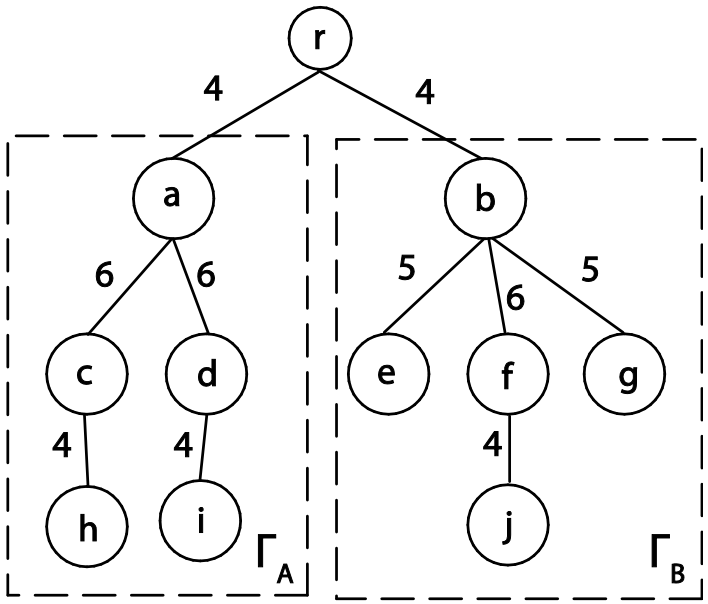
The books and publications listed here are not specifically referenced in the text, however they deserve mention as they have influenced and inspired the ideas proposed.

- J. Menezes, Oorschot Paul C. Van, Scott A. Vanstone, and R. L. Rivest. *Handbook of Applied Cryptography*. CRC, 2001.
- P. E. Schupp "On Dehn's Algorithm and the Conjugacy Problem." *Mathematische Annalen* 178.2 (1968): 119-30.
- J. M. Alonso, T. Brady, D. Cooper, V. Ferlini, M. Lustig, M. Mihalik, and M. Shapiro. *Notes on Word Hyperbolic Groups*. Ed. H. Short. M.S.R.I., 1990.
- Ilya Kapovich, Alexei Myasnikov, Paul Schupp, and Vladimir Shpilrain. "Generic-case Complexity, Decision Problems in Group Theory, and Random Walks." *Journal of Algebra* 264 (2003): 665-94.

Appendices

Appendix A: An example of platform generation

Let us suppose that the tree Γ has the following structure:



Then the corresponding Artin group will admit the following presentation:

$$G = \langle \{r,a,b,c,d,e,f,g,h,i,j\} \mid \begin{array}{l} rara = arar, \\ acacac = cacaca, \\ chch = hchc, \\ adadad = dadada, \\ didi = idid, \\ rbrb = brbr, \\ bebeb = ebebe, \\ bfbfbf = fbfbfb, \\ fjfj = jfjf, \\ bgbgb = gbgbg \end{array} \rangle$$

Some example endomorphisms for the partitions of G are:

$\{d \to c, i \to h\} \in EndG_A$

$\{c \to d,\ d \to c,\ h \to i,\ i \to h\} \in EndG_A$

$\{e \to g\} \in EndG_B$

Appendix B: An example of simple analysis

For this example we will use the tree from appendix A as the platform. The public element will be the randomly generated (and then freely reduced) word

$w = AFIhcRRRBIEdcRDHgFAfccJAJHBhbGEIBGEACjbGrD$
where the capital letters stand for the inverse of the corresponding generator. We will also need the private keys selected by the two parties, let them be
$f_A = \{d \to c, i \to h\}$ and $f_B = \{e \to g\}$. Then the elements transmitted would be:
$f_A(w) = AFcRRRBHEccRCHgFAfccJAJHBhbGEHBGEACjbGrC$ and
$f_B(w) = AFIhcRRRBIGdcRdHgFAfccJAJHBhbGGIBGGACjbGrD$. The common key will be $f_A(f_B(w)) = AFcRRRBHGccRCHgFAfccJAJHBhbGGHBGGACjbGrC$.

When calculating the edit distance (as defined in the present text, that is no additions are allowed and replacements of a letter with another only count once) between w and $f_A(w)$ our opponent will see that the following operations are needed: delete "Ih", replace "I" with "H", replace "D" with "C", replace "d" with "c". The simpler function that satisfies these constraints is indeed f_A and thus the secrecy of the key is compromised.

Note that as the public element does not contain any subword that is also a subword of a defining relator that has at least half its (the relator's) length, any simple substitution using defining relations will be reversible by Dehn's algorithm.

Appendix C: A simple case of using fingerprints

Suppose that the setting is as in the previous example and our opponent has obtained $w, f_A(w), f_B(w)$ and now tries to retrieve one of the keys. She will search the public word for a sequence in the form of $x_1\ y\ x_2$ where the middle generator belongs to a different tree segment than the others. Let us suppose that the JAJ segment is chosen. Since A belongs to the left tree, she will scan $f_A(w)$ for a sequence $J\ y\ J$, but the sequence she finds is again JAJ. This means that f_A probably maps the generator 'a' to itself. Then she may select another sequence, let it be IEd. As E belongs to the right tree, she searches in $f_B(w)$ for $I\ y\ d$ and finds IGd. Then she assumes that f_B maps the generator 'e' to 'g'. Continuing this way will reveal enough clues for the private keys that she could reconstruct them or at least apply them on either transmitted element to acquire the common secret key.