Zero Knowledge with Rubik's Cubes and Non-Abelian Groups

Emmanuel Volte[1], Jacques Patarin[2], and Valérie Nachef[1]

[1] Department of Mathematics, University of Cergy-Pontoise, CNRS UMR 8088
2 avenue Adolphe Chauvin, 95011 Cergy-Pontoise Cedex, France
[2] PRISM, University of Versailles
45 avenue des Etats-Unis, 78035 Versailles Cedex, France
emmanuel.volte@u-cergy.fr

Abstract. The factorization problem in non-abelian groups is still an open and a difficult problem [12]. The hardness of the problem is illustrated by the moves of the Rubik's cube. We will define a public key identification scheme based on this problem, in the case of the Rubik's cube, when the number of moves is fixed to a given value. Our scheme consists of an interactive protocol which is zero-knowledge argument of knowledge under the assumption of the existence of a commitment scheme. We will see that our scheme works with any non-abelian groups with a set of authorized moves that has a specific property. Then we will generalize the scheme for larger Rubik's cubes and for any groups.

Key words: zero-knowledge, Rubik's cube, authentication, symmetric group, cryptographic protocol, factorization

1 Introduction

The puzzles based on the Rubik's cube meet a great success. Generally speaking, Rubik's cube's owners try to solve the following problem: how to recover the initial position of the cube from a random position. At first sight, this problem seems very difficult, but there exist efficient algorithms to solve it [2]. Nevertheless, several other problems with the cube and its neighboring puzzles seem to be really difficult from a computing point of view. For example if we impose that the number of moves is equal (or inferior) to a fixed value d that makes **unique** or **almost unique** the moves that must be done to recover the cube, then we obtain a difficult problem. In [2] it is showed that finding an optimal solution (i.e. the minimum factorization) is NP-hard if we ignore some of the facets of the cube $n \times n \times n$. Moreover the size of the Rubik's group grows exponentially with the number of facets. In appendix A, we will also discuss some connections between these problems and NP-complete or NP space problems.
Consequently, we can try to build some public key zero knowledge argument of knowledge protocols with a proven security linked to these difficult problems (and also on the existence of a commitment scheme). Then we can use this protocol to do identification. It is well known that

there exist cryptographic algorithms transforming every NP problem into a zero knowledge authentication protocol [5]. The theoretical way to do this is polynomial but nevertheless generally not efficient at all. This is why we will present and study some specific algorithms in this article, which can be used for practical cryptography and with a proven security based on some difficult well-known problems of the Rubik's cube.

Our algorithm is the first serious attempt to make zero knowledge argument of knowledge with the Rubik's cube. There were obvious ways to do zero-knowledge with this toy. For example, we can scramble the cube and memorize all the moves, then we can prove that we can recover the initial position under a scarf. Nevertheless, since some people can recover the cube even without seeing it, this is not a sure way to authenticate oneself. As Colmez says in [1], the Rubik's cube is one of the rare groups we can walk with in the street.

Organization of the paper. In **Section** 1 we introduce all the notations and the definition of the repositioning group that is crucial to write all our schemes. In **Section** 2 we define the problem we will use for our zero knowledge argument of knowledge, this problem is equivalent to the factorization with a fixed number of elements from a given set. We also see the generic attack for this problem. In **Section** 3, we show how to construct a scheme that is zero knowledge argument of knowledge, in the case of the Rubik's cube $3 \times 3 \times 3$, and we prove that this still works for any group and any set of generators that has a repositioning group. The scheme we will propose is an interactive one with 3 pass. We use a standard cut-and-choose technique.

- First, the Prover hides each move of the solution thanks to a rotation of the cube. By doing this, we still can see that she makes a basic move but without knowing which one.
  Then she masks all the turned moves with a unique random permutation that preserves the composition. Finally, the Prover only sends commitments of the rotation, of the random permutation used for masking and of all the masked permutations.
- The verifier asks for some verification. She has the choice of verifying the entire composition or only one of the turned moves. She can not check 2 moves simultaneously because she will have the information of the equality or not of the two initial moves.
- The prover reveals some of the permutations and the Verifier checks the answer, then accepts or not.

In **Section** 4, we try to do this with the Rubik's cube $5 \times 5 \times 5$. The difficulty is that the set of generators has no repositioning group. By working in larger groups, we manage to construct a scheme that is suitable for cryptographic applications, and that is quite efficient. In **Section** 5, we first generalize the scheme for any group and any set of generators, and then for a number of moves that is not constant but inferior to a given value, for example the diameter of a group. There are recent papers [14] that help us give an approximation of this value for some groups and for some set of generators. These works try to answer Babai's conjecture on the diameter of simple groups. In the case of the Rubik's cube, in [2] it is shown that "God's number", i.e. the minimal number of moves to solve

a Rubik's cube $n \times n \times n$ is $\Theta(n^2/log(n))$. At last, in **Section** 7, we will briefly discuss the efficiency of our schemes.

2 Notations and definitions

2.1 Mathematical standard notations and definitions

Most of the following notations can be found in [8], which is an original way to learn algebra with toys such as the Rubik's cube and the Merlin's machine.
For a finite set X, S_X is the symmetric group of X. In the particular case $X = \{1; 2; \ldots; n\}$ where $n \in \mathbb{N}^*$, we call this group S_n. For $\sigma, \sigma' \in S_X$, we use the classic notation $\sigma\sigma'$ to design the composition $\sigma' \circ \sigma$.
When G is a group, and $(g_1, g_2, \ldots, g_\alpha) \in G^\alpha$, then $\langle g_1, g_2, \ldots, g_\alpha \rangle$ is the subgroup generated by $g_1, g_2, \ldots g_\alpha$.
We say that $\mathcal{F} = \{g_1, \ldots, g_\alpha\}$ is a **set of generators** of G when $\langle g_1, g_2, \ldots, g_\alpha \rangle = G$. This set is **symmetric** when for all $\sigma \in \mathcal{F}$ we have $\sigma^{-1} \in \mathcal{F}$.
When we have a symmetric set of generators $\mathcal{F}$ of a group G, we set that two elements g and g' are in relation if and only if $g^{-1}g' \in \mathcal{F}$. The corresponding graph is called the **Cayley graph** of the group.
Let G be a group, the **conjugation** on G is defined by

$$\forall(\sigma, \tau) \in G^2, \quad \sigma^\tau = \tau^{-1}\sigma\tau$$

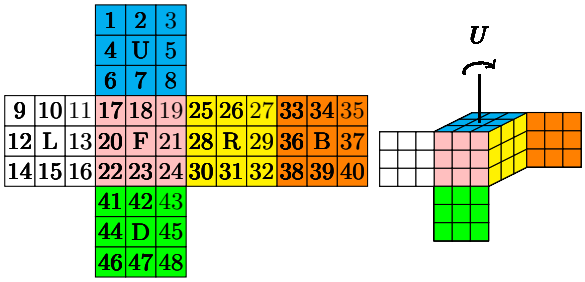
Moreover we have:

$$\forall(\sigma, \sigma', \tau, \tau') \in G^4, \quad (\sigma^\tau)^{\tau'} = \sigma^{\tau\tau'}, \quad \sigma^\tau\sigma'^\tau = (\sigma\sigma')^\tau$$

We can also write $\boxed{\sigma^G = \{\sigma^g | g \in G\}}$.

2.2 Mathematical representation of the Rubik's cube.

For the Rubik's cube, we can write a number on each facet except the centers. In this paper we consider that all the centers are white or void (there exists in fact a physical cube that has no centers). Not taking in consideration the centers will not really change the complexity of all the problems



then we define 6 permutations of $G = S_{48}$ which are the basic clockwise quarter turns of the faces:

$F = (17,19,24,22)(18,21,23,20)(6,25,43,16)(7,28,42,13)(8,30,41,11)$
$B = (33,35,40,38)(34,37,39,36)(3,9,46,32)(2,12,47,29)(1,14,48,27)$
$L = (9,11,16,14)(10,13,15,12)(1,17,41,40)(4,20,44,37)(6,22,46,35)$
$R = (25,27,32,30)(26,29,31,28)(3,38,43,19)(5,36,45,21)(8,33,48,24)$
$U = (1,3,8,6)(2,5,7,4)(9,33,25,17)(10,34,26,18)(11,35,27,19)$
$D = (41,43,48,46)(42,45,47,44)(14,22,30,38)(15,23,31,39)(16,24,32,40)$

after the move σ, the facet $i \in \{1, 2, \ldots, 48\}$ is at the $\sigma(i)$ position.
The Rubik's cube group is $G_R = \langle F, B, L, R, U, D \rangle \subset S_{48}$. If you want to simulate the Rubik's cube, this can be done by using SAGE [16].

2.3 Cryptographic notations.

When X is a finite set, $\boxed{x \in_R X}$ means that we take a random element in X with a uniform probability.
In an interactive Protocol, there are two entities: the prover and the verifier. The Prover wants to convince the verifier that she knows a secret. Both interact and at the end, the verifier accepts or refuses. In Zero-Knowledge Protocols there is a possibility of fraud. A cheater will be able to answer some of the questions (but not all of them). The protocol must be designed such that an answer to one of the questions does not give any indication on the secret but if someone is able to answer all the questions then this will reveal the Prover's secret. We will use the following definitions in order to describe the properties that we want to be satisfied by our protocols:

1. The protocol has **perfect correctness** if a legitimate prover is always accepted.
2. The protocol is **statistically zero knowledge** if there exists an efficient simulating algorithm U such that for every feasible Verifier strategy V, the distributions produced by the simulator and the proof protocol are statistically indistinguishable.
3. The protocol is **proof of zero knowledge with error knowledge** α if there is a knowledge extractor K and a polynomial Q such that if p denotes the probability that K finds a valid witness for x using its access to a prover P^* and p_x denotes the probability that P^* convinces the honest verifier on x, and $p_x > \alpha$, then we have $p \geq Q(p_x - \alpha)$.

In our protocols, we will need string commitment schemes. A string commitment function is denoted by Com. The commitment scheme runs in two phases. In the first phase, the sender computes a commitment value $c = Com(s; \rho)$ and sends c to the receiver, where s is the committed string and ρ is a random string. In the second phase, the sender gives (s, ρ) and the receiver verifies if $c = Com(s; \rho)$. we require the two following properties of Com.

1. The commitment scheme is **statistically hiding** if for uniform (x, ρ) and (x', ρ') the distributions $Com(x, \rho)$ and $Com(x', \rho')$ are statistically indistinguishable. This means that the commitment to x reveals (almost) no information on x even to an infinitely powerful Verifier.

2. The commitment scheme is **computationally binding** if the probability to that two different values (x, ρ) and (x', ρ') produce the same $c = Com(x, \rho) = Com(x', \rho')$ is negligible in polynomial time, i.e. the chances to change the committed value after the first phase are very small.

A practical construction of such a commitment is given in [7].
For all of our schemes we will use an interactive commitment, it means that we have to send a key to unlock the commitment. We use the notation $Com_k(x)$ to design such a commitment of x with the key k where k is a 80-bit random word.

2.4 The repositioning group

In this Section, we will define precisely the repositioning group for a given set of elements $\mathcal{F}$. We will see in the next Sections that the existence of this group is the keystone of our schemes.

Definition 1 *Let $\mathcal{F} = \{f_1, \ldots, f_\alpha\} \subset G$, where G is a group. If there exists a subgroup $H \subset G$ such that ${f_1}^H = \{h^{-1} f_1 h \mid h \in H\} = \mathcal{F}$ then H is called a **repositioning** group of $\mathcal{F}$.*

Remark. In the case of the Rubik's cube, with $\mathcal{F} = \{F, B, L, R, U, D\}$, it is easy to see that we can go from one move to another by rolling the cube like a dice.

Proposition 1 *We suppose $\mathcal{F}$ has a repositioning group H. If we choose $\tau \in_R H$, $P({f_i}^\tau = f_j) = \frac{1}{\alpha}$ for all $(i, j) \in \{1; \ldots; \alpha\}^2$.*

Proof. Since ${f_1}^H = \mathcal{F}$, for all $i \in \{1, \ldots; \alpha\}$ there exists $\tau_i \in H$ such that ${f_1}^{\tau_i} = f_i$. Then, for all $j \in \{1; \ldots; \alpha\}$, ${f_i}^{{\tau_i}^{-1}\tau_j} = f_j$. We denote by $\tau_{ij} = {\tau_i}^{-1}\tau_j$. Now we have the equivalence:

$$ {f_i}^\tau = f_j \iff {f_k}^{\tau_{ki}\tau\tau_{j\ell}} = f_\ell $$

for all $k, \ell \in \{1; \ldots; \alpha\}$. So $\{\tau \in H \mid {f_i}^\tau = f_j\}$ and $\{\tau \in H \mid {f_k}^\tau = f_\ell\}$ are in bijection and have the same cardinality.

Remark. It is not easy to find a repositioning group in the general case. When the group elements in $\mathcal{F}$ are not conjugate of each other, it is even impossible. We will still see a way to do this for the general case (i.e. for any set of generators $\mathcal{F}$), with the help of an extended set. Nevertheless the general construction is often not the optimal solution. For example, in the case of the Rubik $5 \times 5 \times 5$ the orientation preserving group of the cube enables us to work on ${S_{144}}^2$ instead of ${S_{144}}^{12}$.

3 Various Factorization Problems

For all the following problems, we have a finite group G with a set of α generators $\mathcal{F} = \{f_1; f_2; \ldots; f_\alpha\}$, containing all authorized permutations. Of course we have $f_i \neq f_j$, if $i \neq j$.
$id \in G$ is the neutral element of G.

Problem 1: *Solving the puzzle.*
Given $x_0 \in G$, find $d \in \mathbb{N}$ and $i_1, i_2, \ldots, i_d \in \{1; 2; \ldots; \alpha\}$ such that

$$x_0 f_{i_1} f_{i_2} \ldots f_{i_d} = id$$

Remark This problem is equivalent to the factorization problem in G with elements of $\mathcal{F}$ because:

$$x_0 f_{i_1} f_{i_2} \ldots f_{i_d} = id \quad \Longleftrightarrow \quad x_0^{-1} = f_{i_1} f_{i_2} \ldots f_{i_d}$$

Problem 2: *Solving the puzzle in a given number of moves.*
Given $x_0 \in G$ and $d \in \mathbb{N}^*$, find $i_1, i_2, \ldots, i_d \in \{1; 2; \ldots; \alpha\}$ such that

$$x_0 f_{i_1} f_{i_2} \ldots f_{i_d} = id$$

Proposition 2 *We can find a solution of problem 2 with $O(d\alpha^{d/2})$ computations if d is even.*

Proof. This is a meet-in-the-middle attack. We notice that $f_{i_1} f_{i_2} \ldots f_{i_d} = x_0$ is equivalent to $x_0 f_{i_1} \ldots f_{i_{d/2}} = (f_{i_d})^{-1} \ldots (f_{i_{d/2+1}})^{-1}$.
So, for each $i_1, i_2, \ldots, i_{d/2} \in \{1, \ldots, \alpha\}$ we compute

$$\begin{aligned} Y_{i_1 i_2 \ldots i_{d/2}} &= x_0 f_{i_1} f_{i_2} \ldots f_{i_{d/2}} \\ \text{and} \quad Z_{i_1 i_2 \ldots i_{d/2}} &= (f_{i_1})^{-1} (f_{i_2})^{-1} \ldots (f_{i_{d/2}})^{-1} \end{aligned}$$

Then we look for a collision between Y and Z.

Remark. There are other techniques of factorization [12] that are using a tower of groups. Nevertheless these techniques do not lead us to the minimal solution.
In this paper we will study how to transform these difficult problems into a zero-knowledge argument of knowledge identification scheme. In other words: we will study how to prove that we have a solution of one of these problems without revealing anything of the solution.

4 With Rubik's cube $3 \times 3 \times 3$

4.1 Introduction

We will first describe a zero-knowledge authentication scheme based on Rubik's classical cube $3 \times 3 \times 3$. We do this in order to introduce the main ideas with this relatively simple example. However with Rubik's

cube $3 \times 3 \times 3$ the complexity of problem 2 is much smaller than 2^{80} and therefore we cannot use it for cryptographic security (for cryptographic applications we will use the Cube $5 \times 5 \times 5$ as we will see below).
We have in fact about 43.2×10^{18} different positions for this Rubik's cube, so about 2^{61} or 6^{25}. If we consider that half a turn counts as one move, we know that God's number (i.e. the minimal number of moves necessary to unscramble any position of the Rubik's cube) is 20 [15]. Nevertheless in our case we do not authorize un-clockwise quarter turns and half turns. So it seems reasonable to choose for problem 2 the value $d = 24$, and the security will be about $6^{24/2} = 6^{12} \approx 2^{30}$ computations.

4.2 Hiding the secret

First we have to hide the permutation we make to go from a position to another, without hiding that we make one authorized permutation, i.e. one element of $\mathcal{F}$. An easy way to do this with the cube $3 \times 3 \times 3$ is to roll the cube like a dice (we always consider that the centers of the faces do not move or do not exist).
Let H be the group of the orientation-preserving symmetry of the cube. We have $H = \langle h_1, h_2 \rangle$ where h_1 is the cube rolling on its back , and h_2 the cube laying on the table but turning as a whole one clockwise quarter of a turn. To be more precise we have:

$$h_1 = RL^{-1}(2, 39, 42, 18)(7, 34, 47, 23)$$
$$h_2 = UD^{-1}(13, 37, 29, 21)(12, 36, 28, 20)$$

It is easy to check that $|H| = 24$, because for each face up, we have 4 choices for the face in front. Moreover we have $U^H = \mathcal{F}$.

Proposition 3 *If $f \in_R \mathcal{F}$ and $\tau \in_R H$, then f^τ is a random variable with a uniform law on $\mathcal{F}$.*

Proof. This is a direct consequence of Proposition 1.

Illustration. Let $x_0 \in G_R = \langle \mathcal{F} \rangle$ be one position of the cube and $x_1 = x_0 f$, the following diagram is commutative (i.e. $f\tau = \tau f^\tau$):

$$\begin{array}{ccc} x_0 & \xrightarrow{f} & x_1 \\ {\scriptstyle\tau}\downarrow & & {\scriptstyle\tau}\downarrow \\ x_0\tau & \xrightarrow{f^\tau} & x_1\tau \end{array}$$

Secondly, we want to hide each of the conjugate authorized moves, at each step of the resolution. For this we use a mask, a random permutation of G called σ_0. If $f_{i_1}, f_{i_2}, \ldots f_{i_d}$ are the secret moves, we hide their conjugate moves this way (by defining σ_j for all $j \in \{1; 2; \ldots; d\}$): ${f_{i_1}}^\tau = \sigma_0 \sigma_1^{-1}$, then ${f_{i_2}}^\tau = \sigma_1 \sigma_2^{-1}$, ..., and ${f_{i_d}}^\tau = \sigma_{d-1} \sigma_d^{-1}$. So we have $f_{i_1} \ldots f_{i_d} = \sigma_0 {\sigma_d}^{-1}$.

4.3 ZK protocol

In this subsection,we will give the general protocol for any group G, any set of generators $\mathcal{F}$ of a large subgroup G_R of G, and we suppose that this set has a repositioning group $H \subset G$. We will prove in the next subsection that it is a zero knowledge argument of knowledge scheme. We can also use this protocol for the puzzle called $S41$ described in appendix C.

Public:

- A group G.
- A set $\mathcal{F} = \{f_1, \ldots, f_\alpha\} \subset G$ of generators of G_R
- A repositioning group $H \subset G$ such that ${f_1}^H = \mathcal{F}$.
- $d \in \mathbb{N}$, $d \geq 3$
- G' subgroup of G generated by $\mathcal{F}$ and H. $G' = \langle \mathcal{F}, H \rangle$.
- K a set of keys, $|K| \geq 2^{80}$.

Secret key: $i_1, i_2, \ldots, i_d \in \{1, 2, \ldots, \alpha\}$.
Public key: $x_0 = (f_{i_1} f_{i_2} \ldots f_{i_d})^{-1}$
Scheme (one round):

Prover		**Verifier**
Picks $\tau \in_R H$, $\sigma_0 \in_R G'$, $k_*, k_0, k_1, \ldots, k_d \in_R K$ **Computes** $\forall j \in \{1, \ldots, d\}$, $\sigma_j = ({f_{i_j}}^\tau)^{-1} \sigma_{j-1}$ $c_0 = Com_{k_*}(\tau)$ $\forall i \in \{0, \ldots d\}$, $s_i = Com_{k_i}(\sigma_i)$	$c_0, s_0, \ldots, s_d$ $\longrightarrow$	
	$\longleftarrow$ q	**Picks** $q \in_R \{0, \ldots d\}$
Case $q = 0$	$\tau,\ \sigma_0$ $\longrightarrow$ $k_*,\ k_0,\ k_d$	**Computes** $\sigma_d = \tau^{-1} x_0 \tau \sigma_0$ **Checks** $\tau \in H$, $Com_{k_*}(\tau) = c_0$, $Com_{k_0}(\sigma_0) = s_0$, $Com_{k_d}(\sigma_d) = s_d$ **If** all tests ok **then accepts else rejects.**
Case $q \neq 0$	${f_{i_q}}^\tau,\ \sigma_q$ $\longrightarrow$ $k_{q-1},\ k_q$	**Computes** $\sigma_{q-1} = {f_{i_q}}^\tau \sigma_q$ **Checks** ${f_{i_q}}^\tau \in \mathcal{F}$, $s_{q-1} = Com_{k_{q-1}}(\sigma_{q-1})$ $s_q = Com_{k_q}(\sigma_q)$ **If** all tests ok **then accepts else rejects.**

Remark. Since H is a small group, we can change a little the protocol by not sending c_0 in the first phase, and only sending σ_0 in the first case of the third phase. Then the Verifier will try all the possible values for τ. So it is quite obvious that it is not the size of H that secures the scheme.
Illustration. If we define for all $k \in \{1; \ldots; d\}$, $x_k = x_{k-1} f_{i_k}$, we have the following commutative diagram:

$$\begin{array}{ccccccc}
x_0 & \xrightarrow{f_{i_1}} & x_1 & \xrightarrow{f_{i_2}} & \ldots & x_{d-1} & \xrightarrow{f_{i_d}} & x_d = id \\
\tau\downarrow & & \tau\downarrow & & & \tau\downarrow & & \tau\downarrow \\
x_0\tau & \xrightarrow[\sigma_0{\sigma_1}^{-1}]{{f_{i_1}}^{\tau}} & x_1\tau & \xrightarrow[\sigma_1{\sigma_2}^{-1}]{{f_{i_2}}^{\tau}} & \ldots & x_{d-1}\tau & \xrightarrow[\sigma_{d-1}{\sigma_d}^{-1}]{{f_{i_d}}^{\tau}} & \tau
\end{array}$$

With $q = 0$, the Verifier will check that the exterior composition way is correct:

$$\begin{array}{ccc}
x_0 & & x_d = id \\
\tau\downarrow & & {\tau^{-1}}\uparrow \\
x_0\tau & \xrightarrow[\sigma_0{\sigma_d}^{-1}]{} & \tau
\end{array}$$

With $q \neq 0$, the Verifier checks one of the meshes:

$$\begin{array}{ccc}
 & \xrightarrow{f_{i_q}} & \\
\tau\downarrow & & \tau\downarrow \\
 & \xrightarrow[\sigma_{q-1}{\sigma_q}^{-1}]{{f_{i_q}}^{\tau}} &
\end{array}$$

Here τ is not revealed, we just have a random element σ_{q-1} of G and a random element of $\mathcal{F}$, so we give no information on the secret.

4.4 Proof of ZK protocol

Correctness. Obviously, a legitimate Prover will always be accepted.

Proof of zero knowledge with error knowledge $\frac{d}{d+1}$. We first suppose that a Prover can answer correctly for all possible values of q (i.e. is accepted by the Verifier). Since the commitment scheme is computationally binding, we can state that:

- σ_0 revealed for $q = 0$ is the same as the one computed for $q = 1$.
- σ_i for $i \in \{1; \ldots; d-1\}$ revealed for $q = i$ is the same as the one computed for $q = i + 1$.
- σ_d revealed for $q = d$ is the same as the one computed for $q = 0$.

For all $i \in \{1; \ldots; d\}$, the Verifier has checked that $\sigma_{i-1}{\sigma_i}^{-1} \in \mathcal{F}$, so let $u_i \in \{1; \ldots; \alpha\}$ such that $f_{u_i} = \sigma_{i-1}{\sigma_i}^{-1}$.
With $q = 0$, the Verifier established $id = x_0\tau\sigma_0{\sigma_d}^{-1}\tau^{-1}$, so

$$\begin{aligned}
id &= x_0\tau(\sigma_0{\sigma_1}^{-1})(\sigma_1{\sigma_2}^{-1})\ldots(\sigma_{d-1}{\sigma_d}^{-1})\tau^{-1} \\
&= x_0\tau f_{u_1} f_{u_2} \ldots f_{u_d}\tau^{-1} = \tau f_{u_1}\tau^{-1}\tau f_{u_2}\tau^{-1}\ldots\tau f_{u_d}\tau^{-1} \\
&= x_0 {f_{u_1}}^{\tau^{-1}} {f_{u_2}}^{\tau^{-1}} \ldots {f_{u_d}}^{\tau^{-1}}
\end{aligned}$$

Hence we have a solution of the initial problem.
Therefore, if we consider a Cheat Prover i.e. a person who does not have a solution to the initial problem, there is at least one of the Verifier's request (one of the q value) that will lead to a rejection. So the probability that the Cheat Prover can convince a Honest Verifier is less than $\frac{d}{d+1}$.

Statistically zero knowledge. Firstly we show that for a legitimate prover, each answer has a uniform probability over the corresponding set.

- $q = 0$.
  The Prover gives $(\tau, \sigma_0, k_*, k_0, k_d) \in H \times G' \times K^3$ which are all independent random values over the concerning set.
- $1 \leq q \leq d$.
  The Prover gives ${f_{i_q}}^{\tau}, \sigma_q, k_{q-1}, k_q) \in F \times G' \times K^2$. Since we have $f_{i_1} \dots f_{i_q} = \tau\sigma_0\sigma_q^{-1}\tau^{-1}$, if we define $h = \tau^{-1}{f_{i_q}}^{-1} \dots {f_{i_1}}^{-1}\tau$, then we have $\sigma_q = h\sigma_0$ with σ_0 picked at random in G', so σ_q is a random permutation independent from ${f_{i_q}}^{\tau}$. Moreover, since $\tau \in_R H$, ${f_{i_q}}^{\tau}$ is uniformly chosen in F (see section 4.2, or below Proposition 1 for the generalization). So we have again a uniform probability over $F \times G' \times K^2$.

Secondly, we construct a black-box simulator which takes x_0 without knowing the secret, and interacts with a Cheating Verifier **CV**. We show that the simulator can impersonate the honest prover with probability $\frac{1}{d+1}$. The simulator randomly chooses a value $q^* \in_R \{0; 1; \dots; d\}$, this is a prediction what value **CV** will not choose. We consider two cases:

- $q^* = 0$
  The simulator picks $\tau \in_R H$, $f'_1, \dots, f'_d \in_R F$ and $\sigma_0 \in_R G'$. Then it computes for all $k \in \{1; \dots; d\}$ $\sigma_k = {f'_k}^{-1}\sigma_{k-1}$.
- $1 \leq q^* \leq d$
  It picks $f'_1, f'_2, \dots, f'_{q^*-1}, f'_{q^*+1}, \dots f'_d \in_R \mathcal{F}$. Then it picks $\tau \in_R H$ and $\sigma_0 \in_R G'$. It computes $f'_{q^*} \in G'$ (not necessary in $\mathcal{F}$) such that $x_0 f'_1 \dots f'_d = id$, and for all $k \in \{1; \dots; d\}$ $\sigma_k = {f'_k}^{\tau^{-1}}\sigma_{k-1}$

It is easy to check that, except for $q = q^*$, every request of **CV** will have a satisfying answer. So the probability that it fails is $\frac{1}{d+1}$. Moreover, when only the successful interactions are recorded, the communication tape is indistinguishable from what would have been obtained from an execution performed by the real Prover.

4.5 Number of rounds for the cube $3 \times 3 \times 3$

We quit the general case to consider our classical cube $3 \times 3 \times 3$. Here we will discuss of the number of times (r) the prover will do the protocol (one protocol is considered as one round), in order to prove with a good probability that she knows the secret. If we set this probability to $1-2^{-m}$, we must have $\left(\frac{d}{d+1}\right)^r \leq 2^{-m}$, so it gives $r \approx md\ln(2)$. For example, with $m = 30$ and $d = 24$, only 500 rounds are necessary. We simulate this scheme with a 3GHz computer, and a non-optimized algorithm (we used a rather slow hash function for the commitment), and it has token less than 1 second to simulate 100 times all the protocol.

5 Rubik's cube $5 \times 5 \times 5$

For practical authentication we need a puzzle with at least 2^{160} different states. The Rubik's cube $4 \times 4 \times 4$ has (only) about 2^{152} positions. Thus, we choose the next cube, i.e. the cube $5 \times 5 \times 5$ which has about 2^{247} different positions (computation with Sage).

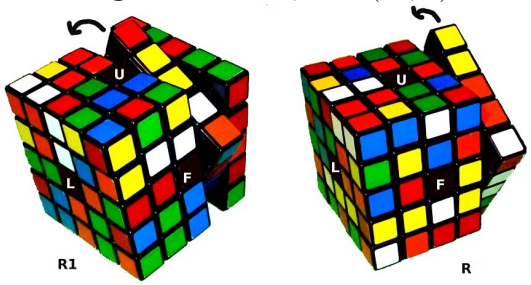
5.1 Mathematical representation

We write numbers on each facet, except the centers. For the manipulation of the cube, we consider only 12 basic permutations. We will choose here the 6 clockwise quarter turns of the upper crown of each face (U, D, F, B, R, L), and the 6 clockwise quarter turns of the first intermediary crown of each face $(U_1, D_1, F_1, B_1, R_1, L_1)$. Nevertheless other choices are possible. We have:

$$G_R = \langle U, D, \ldots, L, U_1, D_1, \ldots, L_1 \rangle \subset S_{144}$$

5.2 Hiding the secret

Fig. 1. Twin cubes, move (R_1, R)



Just rolling the cube is not enough to hide an authorized move. This will only shuffle independently (U, D, F, B, R, L) and $(U_1, D_1, F_1, B_1, R_1, L_1)$. We need a new idea. A way to do this is to duplicate the cube, and so we will consider the group $G_R \times G_R$. Then, each time we use R_1 on the first cube, we will use R on the second cube (see figure 1), each time we use R on the first cube, we will use R_1 on the second cube, each time we use L_1 on the first cube, we use L on the second cube and so on.
We will call $\boldsymbol{e}$ the exchange of the two cubes, $\boldsymbol{e}$ is an extra element that exchanges the coordinates:

$$e^2 = id \quad \text{and} \quad \forall (a, b) \in G_R \times G_R, \quad \boldsymbol{e}(a, b)\boldsymbol{e} = (b, a).$$

To prove the existence of such an element, we can use an injective group morphism from G^2 to S_{288}, because we can decide that the facets of the

second cube are numbered from 145 to 288.
Then $e = (1{,}145)(2{,}146)\dots(144{,}188)$ satisfies the requested properties. For convenience of notations we will still use the notation in G^2.
We set $\boldsymbol{F} = \{(U, U_1), \dots, (L, L_1), (U_1, U), \dots, (L_1, L)\}$ and $\boldsymbol{G}_R = \langle \boldsymbol{F} \rangle \subset G_R \times G_R$. The size of $\boldsymbol{G}_R$ is about the same as G_R. A computation with Sage gives $|G_R| \approx 2^{300}$ and $|\boldsymbol{G}_R| \approx 2^{364}$. We will hide the move by rolling in the same way the two cubes, and exchanging (or not) the two cubes. So we set $\boldsymbol{H} = \langle (h_1, h_1), (h_2, h_2), \boldsymbol{e} \rangle$. This time our repositioning group $\boldsymbol{H}$ has 48 elements: the 24 previous repositioning moves, and all these elements combined with the exchange of the cube (before or after, it has no importance).

5.3 ZK Protocol

The protocol in the previous section works for every set of generators with a repositioning group. In this case, we manage to construct a repositioning group by considering the group G^2 (or G^3 for the cubes $6\times6\times6$ and $7 \times 7 \times 7$, G^n for the cubes $(2n)^3$ and $(2n+1)^3$). We will see in the next section that we can construct in all the cases, a repositioning group by considering the group G^α. For the cube $5 \times 5 \times 5$, we manage to diminish the size of the group because the authorized moves already have some symmetry (rolling the cube shuffles some of the moves).
Then we need to adapt our scheme to the new problem: we will only care to rearrange the first cube, in other words, when we check for $q = 0$ the external way, we just look at the first coordinate in the set G^n ($n = 2$ for the cube $5 \times 5 \times 5$). See appendix B for the details of the scheme. The proof of the zero knowledge argument of knowledge is almost the same as the previous one.

5.4 Choice of d and the number of rounds for the cube $5 \times 5 \times 5$.

If we follow the generic attacks in Proposition 2, we see that we can choose $d = 42$. We have in fact $12^{42} \approx 2^{150}$ which is much smaller than the cardinality of the total number of positions. Nevertheless most of the times when we choose i_1, i_2 ..., i_d the solution is not unique because we can invert the permutations that commute. Then we can impose for the secret that two consecutive chosen permutations of $\mathcal{F}$ must be equal or do not commute. Then there are only $12 \times 9^{d-1}$ possible combinations. With $\boxed{d = 48}$ we have $12 \times 9^{47} \approx 2^{152}$, and $d \times 9^{d/2} > 2^{80}$.
The **number of necessary rounds** is $\boxed{988}$, since $\left(\frac{47}{48}\right)^{988} \leq 2^{-30}$.

6 Generalization with any group

6.1 General method for any set of generators

Let G be a group, $\alpha \in \mathbb{N}$ ($\alpha \geq 2$), $\mathcal{F} = \{f_1, \dots, f_\alpha\}$ such that $\langle \mathcal{F} \rangle = G$. We work with the group G^α and define for all $i \in \{1; \dots; \alpha\}$

$$\boldsymbol{f}^i = (f_i, f_{i+1}, \dots, f_\alpha, f_1, \dots f_{i-1})$$

and we define an extra element $\boldsymbol{h}$ of G^α that verifies:

$$\boldsymbol{h}^\alpha = id \quad \text{and} \quad \forall (a_1, \dots a_\alpha) \in G^\alpha, \quad \boldsymbol{h}^{-1}(a_1, \dots, a_\alpha)\boldsymbol{h} = (a_2, \dots, a_\alpha, a_1)$$

Again, we can prove the existence of such an element thanks to an injective group morphism from G^α to $S_{\alpha n}$, constructed with an injective morphism from G to S_n, because we know from the well-known theorem of Cayley that every finite group can be considered as a subgroup of a symmetric group [11]. Then h is defined in $S_{\alpha n}$ by:

$$\forall i \in \{1; \dots; \alpha n\}, \quad h(i) = \begin{cases} i + n & \text{if } i \leq (\alpha - 1)n \\ i - (\alpha - 1)n & \text{if } i > (\alpha - 1)n \end{cases}$$

Then $\boldsymbol{H} = \langle \boldsymbol{h} \rangle$ is a repositioning group of $\boldsymbol{F} = \{\boldsymbol{f}^1; \dots; \boldsymbol{f}^\alpha\}$.
We can use appendix B to construct our scheme.

6.2 ZK with finite factorization in symmetric groups

Here we consider the case where we do not fix the number of factors, we just give an upper bound of it. This case may seem more difficult than the previous ones, it is in fact a particular case of the previous subsection. We just have to add $f_0 = Id$ to the set of authorized functions, and fix the value d at the diameter of the group, i.e. the maximum distance between two vertices of the Cayley graph of the group. Then we use the same techniques as in previous subsection, it means that we work in $G^{\alpha+1}$ with some extra elements. We will give details of this technique in an extended version of the paper.

7 Efficiency

We suppose we have a symmetric group G whose cardinal is superior to 2^{160} and with a system of generators $\mathcal{F} = \{f_1; f_2; \dots; f_\alpha\}$ so that there exists a permutation h of order α with $f_i = {f_1}^{h^{i-1}}$ for all $i \in \{1; \dots; \alpha\}$. We denote $H = \langle h \rangle$ and H is a repositioning group of $\mathcal{F}$. The system parameters can be built from only h and f_1, so it can take only 320 bits, but in this case we have to compute at each round ${f_1}^{h^j}$ and it will cost about $\lceil \frac{3}{2} \ln_2 \alpha \rceil$ products of permutations (22 if $\alpha = 9240$).
The secret key is an element $(i_1, i_2, \dots, i_d)$ of $\{1; \dots; \alpha\}^d$, so we need $\lceil d \log_2(\alpha) \rceil$ bits.
The public key is $x_0 = f_{i_d}^{-1} \dots f_{i_1}^{-1} \in G$, it takes about 160 bits if the cardinal of G is close to 2^{160}.
In order to compare with existing schemes, we will compute these values for the puzzle $S41$ (see appendix C). We mention in the following table the two different ways to implement $S41$, the first one with all the group H in memory, and the second one with only one generator of H and one element of $\mathcal{F}$. We will denote this last one by $S41'$. We can see in the following table that in terms of performance, our scheme is not so different from the other ones. Moreover, $S41'$ is the most compact of all the schemes, in terms of system parameters. And in both cases, no arithmetic operations are needed.

Table 1. Comparison of 3-pass schemes on 80-bit security against key-recovery attack when the impersonation probability is less than 2^{-30}

	SD [17]	CLE [18]	PP [13]	$S41$	$S41'$
round	52	52	73	260	260
system parameter (bit)	122,500	4,608	28,497	1,478,560	320
public key (bit)	350	288	245	165	165
secret key (bit)	700	192	177	165	165
communication (bit)	59,800	45,517	100,925	673,180	673,180
arithmetic op. (times/field)	$2/S_{700}$	$4/S_{24}$	$2/S_{161}$,S_{177}	0	0
permutations (times/size)	$2/S_{700}$	$4/S_{24}$	$2/S_{161}$,S_{177}	$3/S_{41}$	$23/S_{41}$ (***)
hash function (times)	4	4	8	14 (*)	14 (*)
				2.08 (**)	2.08 (**)
best known recovery attack	2^{87}	2^{84}	$> 2^{74}$	2^{82}	2^{82}

(*) Prover (**) Verifier (***) mean value

8 Conclusion

In this paper, we have studied several authentication schemes built on various factorization problems in non abelian groups. Firstly we proposed zero-knowledge protocols based on different problems with the Rubik's cube and several other generalized cubes. Then we led the generalization for any non abelian group.

The keystone to our constructions relies on the existence of a repositioning group. Whereas the construction of such a group is quite natural for the Rubik's cube $3 \times 3 \times 3$, the existence of the repositioning group needs a special construction for generalized Rubik's cubes. We also explained how to proceed in the general case. Besides, we showed how to construct a random puzzle over a small set that is suitable for the general scheme and can be used for a security in 2^{80}.

Moreover, it is also possible to transform these authentication schemes into signature schemes with the standard transformation used in the "Fiat-Shamir" protocol with a hash function [3].

Our constructions are much more efficient than those obtained with general process [5]. Other puzzles, not mentioned here, can be used in the same way for authentication, but there exist puzzles based on some PSpace complete problems or too dissymmetric puzzles that would be worth having specific analysis.

References

1. Pierre Colmez. Le Rubik's cube, groupe de poche. ENS Ulm, may 2010.
2. Erik D. Demaine, Martin L. Demaine, Sarah Eisenstat, Anna Lubiw, and Andrew Winslow. Algorithms for Solving Rubik's Cubes. In Camil Demetrescu and Magnús M. Halldórsson, editors, *Algorithms – ESA 2011*, volume 6942 of *Lecture Notes in Computer Science*, pages 689–700. Springer Berlin Heidelberg, 2011.

3. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology – Eurocrypt 1990*, volume 473 of *Lecture Notes in Computer Science*, pages 481–486. Springer-Verlag, 1990.
4. Michael R. Garey and David S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completness*. W.H Freeman and Co, 2nd, 1991 edition, 1979.
5. Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM*, 38:690–728, July 1991.
6. Oded Goldreich and Y. Oren. Definitions and properties of Zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994.
7. Shai Halevi and Silvio Micali. Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing . In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 201–215. Springer-Verlag, 1996.
8. David Joyner. *Adventures with Group Theory: Rubik's Cube, Merlin's Machine, and Other Mathematical Toys*. The Johns Hopkins University Press, 2nd edition, 2008.
9. Graham Kendall, Andrew J. Parkes, and Kristian Spoerer. A Survey of NP-Complete Puzzles. *ICGA Journal*, 31(1):13–34, 2008.
10. Dexter Kozen. Lower bounds for natural proof systems. In *FOCS*, pages 254–266, 1977.
11. Serge Lang. Algebra Revised 3rd Edition. 2002.
12. Christophe Petit and Jean-Jacques Quisquater. Rubik's for cryptographers. *IACR Cryptology ePrint Archive*, 2011:638, 2011.
13. David Poincheval. A New Identification Scheme based on the Perceptrons Problem. In Alfredo de Santis, editor, *Advances in Cryptology – EUROCRYPT 1995*, volume 950 of *Lecture Notes in Computer Science*, pages 319–328. Springer-Verlag, 1995.
14. L. Pyber and E. Szabó. Growth in finite simple groups of Lie type of bounded rank. *ArXiv e-prints*, May 2010.
15. Tomas Rokicki, Herbert Kociemba, Morley Davidson, and John Dethrige. God's number is 20. http://cube20.org.
16. W. A. Stein et al. *Sage Mathematics Software (Version 4.7-OSX-32bit-10.5)*. The Sage Development Team, 2011. `http://www.sagemath.org`.
17. Jacques Stern. A New Identification Scheme based on Syndrome Decoding. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO 1993*, volume 773 of *Lecture Notes in Computer Science*, pages 13–21. Springer, 1993.
18. Jacques Stern. Designing Identification Schemes with Keys of Short Size. In Yvo G. Desmedt, editor, *Advances in Cryptology – CRYPTO 1994*, volume 839 of *Lecture Notes in Computer Science*, pages 164–173. Springer, 1994.

A Possible connections between NP Complete, NP Space and Rubik Problems

In section 3 we have seen several problems based on the Rubik's cube or on generalized Rubik's cubes. When one of the parameters of these puzzles (for example the size n of the cube) becomes large, we wonder how will grow the complexity, asymptotically speaking. We notice that we do not know if some of these problems are NP-complete yet (cf [9] p. 27). Moreover, it is plausible that they are not NP-complete because they have a power of description too limited to describe all the problems of the NP class.
Nevertheless, as we will explain further, some of NP-complete problems have a real similarity with the Rubik's cubes puzzles. So we can consider that these problems, used in this article for authentication, are part of a neighboring class, or a larger class, which is proved NP-complete or NP-space. This is not a proof of the difficulty of Rubik's cube related problems, but it is an indirect argument suggesting it could be true.
Example 1 From [4] p. 280 and [10] we know that the problem "Finite Function Generation" is P-space complete.
Finite Function Generation
INSTANCE: Finite set A, a collection F of functions $f\colon A \to A$ and a specified function $h\colon A \to A$.
QUESTION: Can h be generated from the functions in F by composition ?
Remark. We can notice that here the number of composition functions to be found is not considered, unlike for the Rubik's cube problems where this value d seems to be critical for the complexity.
Example 2 From [4] p. 213, we know that the problem "Longest path" is NP complete.
LONGEST PATH
INSTANCE: Graph $G = (V, E)$, length $l(e) \in \mathbb{Z}^+$ for each $e \in E$, positive integer K, specified vertices $s, t \in V$
QUESTION: Is there a simple path in G from s to t of length K or more, i.e. whose edge lengths sum to at least K ?
Remark. This problem remains NP complete if $l(e) = 1$ for all $e \in E$. Therefore this problem has some similarities with our Rubik problems for going from one position to another. However, as noticed in [4] p. 79 this problem becomes polynomial when we change "of length K or more" by "of length K or less". Nevertheless if we model our graph G such that each vertex is a position of a Rubik's cube $n \times n \times n$, the number of vertices (i.e. possible Cubes) will grow exponentially in n.

B Protocol when the set of generators has no obvious repositioning group

Public:
- A group G^n.
- A set $\mathcal{F} = \{f_1, \ldots, f_\alpha\}$ of generators of $G_R \subset G$

- A set $\boldsymbol{F} = \{\boldsymbol{f}^1, \ldots, \boldsymbol{f}^\alpha\} \subset G^n$ with $\boldsymbol{f}^i = (f_1^i = f_i, f_2^i, \ldots f_n^i)$ for all $i \in \{1, \ldots \alpha\}$.
- A repositioning group $\boldsymbol{H}$ such that ${\boldsymbol{f}^1}^{\boldsymbol{H}} = \boldsymbol{F}$.
- $d \in \mathbb{N}$, $d \geq 3$
- A group $\boldsymbol{G}_R$ generated by $\boldsymbol{F}$ and $\boldsymbol{H}$. $\boldsymbol{G}_R = \langle \boldsymbol{F}, \boldsymbol{H} \rangle$.
- K a set of keys, $|K| \geq 2^{80}$.

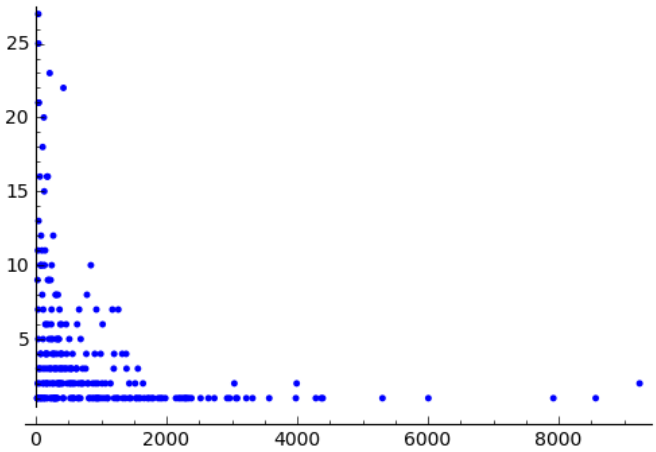
Secret key: $i_1, i_2, \ldots, i_d \in \{1, 2, \ldots, \alpha\}$.
Public key: $x_0 = (f_{i_1} f_{i_2} \ldots f_{i_d})^{-1}$ (or $\boldsymbol{X}^0 = (x_0, id, \ldots, id)$)
Scheme (one round):

Prover		**Verifier**
Picks $\boldsymbol{\tau} \in_R \boldsymbol{H}$, $\boldsymbol{\sigma}_0 \in_R \boldsymbol{G}_R$, $k_*, k_0, k_1, \ldots, k_d \in_R K$ **Computes** $\forall j \in \{1, \ldots, d\}$, $\boldsymbol{\sigma}_j = ({\boldsymbol{f}^{i_j}}^{\boldsymbol{\tau}})^{-1} \boldsymbol{\sigma}_{j-1}$ $c_0 = Com_{k_*}(\boldsymbol{\tau})$ $\forall i \in \{0, \ldots, d\}$, $s_i = Com_{k_i}(\boldsymbol{\sigma}_i)$	$\xrightarrow{c_0, s_0, \ldots, s_d}$	
	$\xleftarrow{q}$	**Picks** $q \in_R \{0, \ldots d\}$
Case $q = 0$	$\xrightarrow[k_*,\ k_0,\ k_d]{\boldsymbol{\tau},\ \boldsymbol{\sigma}_0,\ \boldsymbol{\sigma}_d}$	**Computes** $\boldsymbol{X}^d = \boldsymbol{X}^0 \boldsymbol{\tau} \boldsymbol{\sigma}_0 \boldsymbol{\sigma}_{\boldsymbol{d}}^{-1} \boldsymbol{\tau}^{-1}$ **Checks** $\boldsymbol{X}_1^d = id$ $\boldsymbol{\tau} \in \boldsymbol{H}$, $Com_{k_*}(\boldsymbol{\tau}) = c_0$, $Com_{k_0}(\boldsymbol{\sigma}_0) = s_0$, $Com_{k_d}(\boldsymbol{\sigma}_d) = s_d$ **If** all tests ok **then accepts** **else rejects.**
Case $q \neq 0$	$\xrightarrow[k_{q-1},\ k_q]{{\boldsymbol{f}_{i_q}}^{\boldsymbol{\tau}},\ \boldsymbol{\sigma}_q}$	**Computes** $\boldsymbol{\sigma}_{q-1} = {\boldsymbol{f}_{i_q}}^{\boldsymbol{\tau}} \boldsymbol{\sigma}_q$ **Checks** ${\boldsymbol{f}_{i_q}}^{\boldsymbol{\tau}} \in \boldsymbol{F}$, $s_{q-1} = Com_{k_{q-1}}(\boldsymbol{\sigma}_{q-1})$ $s_q = Com_{k_q}(\boldsymbol{\sigma}_q)$ **If** all tests ok **then accepts** **else rejects.**

C A new puzzle called $S41$

We will present here a new puzzle whose performances seem interesting. We call it $S41$ because we work in the group $G = S_{41}$, which is is the

first symmetric group whose cardinal is superior to 2^{160}. We have in fact $|G| \approx 2^{165}$.
With SAGE, we take two random elements h and f_1 in S_{41} until they generate all the group. In the following chart, we can see horizontally the order of $f_1^{\langle h \rangle}$ (α), and vertically the number of solutions for 1000 tries.



Then, we choose the instance with the biggest α, in order to have a smallest value for d, and in consequence, the smallest value for the number of rounds of the scheme. Here is the instance:

$$\begin{aligned} h = &(1, 14, 39, 19, 31, 18, 37)(3, 36, 4, 23, 20, 34, 16, 25, 17, 26, 35) \\ &(5, 13, 30, 33)(6, 7, 10)(8, 24, 15, 38, 41, 27, 11, 9) \\ &(12, 40, 32, 21, 28)(22, 29), \end{aligned}$$

and

$$\begin{aligned} f_1 = &(1, 11, 31, 6, 17, 34, 25, 24, 22, 12, 4, 28, 3, 14, 5, 27, 32, 13, 26, 8, 23, 2, \\ &20, 41, 19, 10, 40, 15, 38, 16, 37, 39, 35, 21, 18)(7, 29, 36)(9, 30). \end{aligned}$$

We set $H = \langle h \rangle$ and $\mathcal{F} = {f_1}^H$. With SAGE we have checked that $\mathcal{F}$ is a set of generators of G and $|H| = |\mathcal{F}| = \alpha = 9240$. We can fix $d = 12$ for a security in 82 bits. And for an impersonation probability less than 2^{-30} only $r = 260$ rounds are needed.