Zero-Knowledge Proofs from the Action Subgraph

Giacomo Borin, Edoardo Persichetti and Paolo Santini

University of Trento, Sapienza University of Rome, Florida Atlantic University and Marche Polytechnic University

Abstract. In this work, we investigate techniques to amplify the soundness of zero-knowledge proofs of knowledge for cryptographic group actions. We explore the use of a particular graph generated from the group action of random element and provide a fully general protocol with only minimal assumptions on the group action properties. This technique can be seen also as generalization of MPC-in-the-head approach for the context of (non-abelian) group actions. We show that a straightforward translation of the paradigm is unlikely to provide a practical improvement over the simpler construction of a 3-pass Sigma protocol. We then describe a novel approach and show that it yields a computational advantage, therefore laying the ground for new, efficient protocols.

Keywords. Group Actions, Zero-Knowledge, Signature Scheme

1 Introduction

The recent call for signatures issued by the National Institute of Standards and Technology (NIST) [19] has rekindled the community's interest in trying to produce efficient post-quantum schemes using a variety of methods. Among these, cryptographic group actions represent one of the most powerful tools, being the backbone of public-key cryptography ever since the seminal work of Diffie and Hellman [8] and El Gamal [10]. Indeed, the field of post-quantum cryptographic group actions has boomed in the last few years, with several works based on various notions of isomorphism, be that isogenies on elliptic curves [13, 6, 4], code equivalence [5, 3], matrix equivalence [7], lattice isomorphism [9], trilinear forms [20] and others. At the same time, a new trend to build efficient signatures emerged, exploiting the MPC-in-the-head paradigm [18], which led to the development of efficient schemes, again, from a variety of mathematical assumptions [15, 11, 12], or even none (i.e. using only symmetric primitives as in [17]).

1.1 Our Contribution

In this work, we explore the possibility of employing the MPC-in-the-head techniques on top of group actions. We do this by studying what we call the *action subgraph*, and we do so with only minimal assumptions; most importantly, we do not require the commutativity property, which is often the biggest obstacle for protocol design. We show that it is possible to translate the framework successfully to this case, although it requires some non-trivial modifications to enable the secret sharing which is at its core; then, in Section 3, we provide a fully general protocol that yields a proof of knowledge for cryptographic group actions. We analyze the scheme's performance, and in particular signature size; looking at the number of required group action computations, we conclude that a straightforward rendition is unlikely to be competitive with traditional 3-pass zero-knowledge identificatoin (ZK-ID) schemes which use fixed-weight challenges. In other words, increasing soundness without additional subtlety is not better than simply increasing the number of repetitions in a scheme with soundness 1/2. In Section 4, therefore, we illustrate a novel technique that allows to obtain a computational advantage during the verification phase.

Remark 1. During the write-up of this work, we became aware of another recent preprint [16] which verges on topics closely related to this work. Upon discussion with the author, we established that the two works are independent and concurrent, and agreed to keep them separate. We would like to thank the author of [16] for fruitful conversations on the subject.

2 Cryptographic Group Actions

A *group action* is a well-known object in mathematics. It can be described as a function, as shown below, where X is a set and G a group.

$$\begin{aligned} \star : G \times X &\to X \\ (g, x) &\to g \star x \end{aligned}$$

A group action's only requirement is to be *compatible* with the group; using multiplicative notation for G, and denoting with e its identity element, this means that for all $x \in X$ we have $e \star x = x$ and that moreover for all $g, h \in G$, it holds that $h \star (g \star x) = (h \cdot g) \star x$. A group action is also said to be:

- *Transitive*, if for every $x, y \in X$, there exists $g \in G$ such that $y = g \star x$;
- *Faithful*, if there does not exist a $g \in G$ such that $x = g \star x$ for all $x \in X$, other than the identity;
- *Free*, if an element $g \in G$ is equal to identity whenever there exists an $x \in X$ such that $x = g \star x$;
- *Regular*, if it is free and transitive.

The adjective *cryptographic* is added to indicate that the group action in question has additional properties that are relevant to cryptography. For instance, a cryptographic group action should be *one-way*, i.e. given randomly chosen $x, y \in X$, it should be hard to find $g \in G$ such that $g \star x = y$ (if such a g exists). Indeed, the problem of finding this element is known as the *vectorization* problem, or sometimes *Group Action Inverse Problem (GAIP)*.

Problem 1 (GAIP)**.** *Given x and y in X, find, if any, an element $g \in G$ such that $y = g \star x$.*

Finally, other useful properties for group actions include those that make it *effective*, such as for instance the existence of efficient (probabilistic polynomial-time) algorithms for membership testing, sampling, computation (of the group operation and $\star$) etc. More on these properties can be read in [2].

3 Proving Equivalence via Random Paths

We now describe a first approach for a new proof-of-knowledge strategy for cryptographic group actions. As we mentioned in the preamble, we formulate our protocols considering the most general possible setting for group actions. Thus, the only properties we require for our construction are that the action is transitive and faithful.

3.1 The Action Graph

Let $\mathcal{O}(x)$ denote the orbit of x under the action of G, that is

$$\mathcal{O}(x) = \{g \star x \mid x \in G\}.$$

If the action of G is faithful, then $\mathcal{O}(x)$ contains $|G|$ elements. Also, one of them is the public key element $x' = g \star x$ for some secret $g \in G$. We can represent such an orbit through a simple, ordered graph $\mathcal{G}$ whose vertices are the elements in $\mathcal{O}(x)$. Any pair of vertices (x_1, x_2) is connected by a pair of edges (g_1, g_2) such that $x_2 = g_1 \star x_1$ and $x_1 = g_2 \star x_1$ when $\star$ is faithful, $g_2 = g_1^{-1}$. We call $\mathcal{G}$ the *action graph* of G on $X = \mathcal{O}(x)$. We observe also that $\mathcal{G}$ is connected: any pair of vertices (x_1, x_2) is connected by two edges (one for each direction).

Let us now assume that the prover performs a random walk on $\mathcal{G}$, starting from x and ending in x_N. This can be done by generating N random elements of G uniformly at random $\{g_1, g_2, \cdots, g_N\}$, and letting them act on x sequentially, that is

$$x_0 = x \xrightarrow{g_1} x_1 \xrightarrow{g_2} x_2 \xrightarrow{g_3} \cdots \xrightarrow{g_{N-1}} x_{N-1} \xrightarrow{g_N} x_N.$$

Note that, for $i \geqslant 1$, it holds that $x_i = (g_i \cdot g_{i-1} \cdots g_2 \cdot g_1) \star x$. From now on, we will refer to the path going from x to x_N as *random path*. We can think that the random path defines a subgraph $\mathcal{S} \subset \mathcal{G}$, built as follows:

- it has $N + 2$ vertices $\{x_0 = x, x_1, \cdots, x_N, x'\}$;
- it has N edges of the form (x_{i-1}, x_i), for $i = 1, \cdots, N$. Each such edge is labelled with g_i;
- it has N edges (x', x_i), for $i = 1, \cdots, N$. Each such edge is labelled with $g'_i = g_i \cdot g_{i-1} \cdots g_1 \cdot g^{-1}$.

We will refer to $\mathcal{S}$ as *action subgraph*; an example of how this graph looks like is shown in Figure 1. Note that the graph may be enriched with additional edges: indeed, the vertices in $\mathcal{S}$ would form a connected graph, assuming one draws all edges. However, as we describe in the following, the edges we are considering we are considering are the only ones which can be used to build a zero knowledge proof system, and (some) missing edges can be easily recomputed from the information which is provided by the prover.

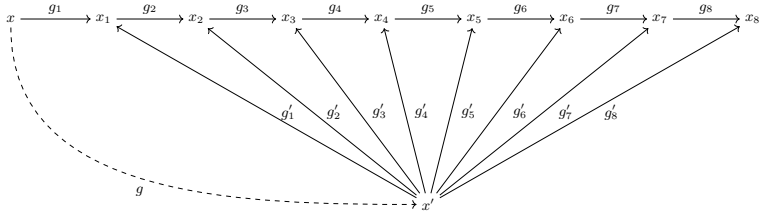


Fig. 1: Example of action subgraph, for $N = 8$.

We now observe that knowing a path $x \mapsto x'$ on $\mathcal{S}$ is possible only if one knows also a secret element g such that $g \star x = x'$. Indeed, let us consider a path $x \mapsto x'$ in $\mathcal{S}$ along the edges $P = \{(x_0, x_1), (x_1, x_2), \cdots, (x_{i-1}, x_i), (x_i, x')\}$, for some $i \in \{1, \cdots, N\}$. Since (x_i, x') is labeled by $g'_i = g_i \cdots g_1 \cdot g^{-1}$ and each (x_{j-1}, x_j) has label g_j, then the secret g can be easily recovered. Indeed, consider that

$$\begin{aligned}(g'_i)^{-1} \cdot (g_1 \cdots g_i) &= \left((g_i \cdots g_1) \cdot g^{-1}\right)^{-1} \cdot (g_1 \cdots g_i) \\ &= \left(g \cdot (g_i \cdots g_1)^{-1}\right) \cdot (g_1 \cdots g_i) = g. \end{aligned} \tag{1}$$

3.2 Proofs of Knowledge from the Action Subgraph

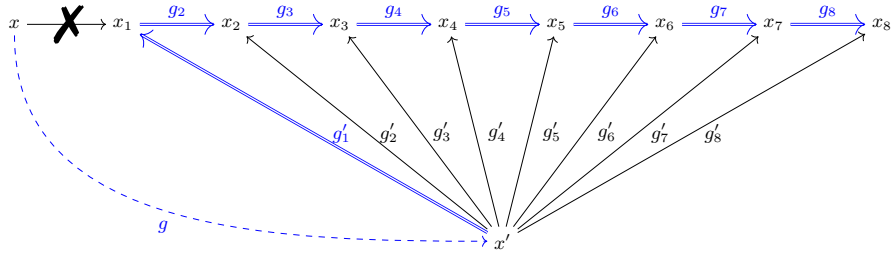
The main idea, to build a ZK proof of knowledge, is that of proving knowledge of a path $x \mapsto x'$ without revealing all the edges. This would prevent an adversary from recovering the secret key by applying (1). The subgraph will be constructed in the usual fashion, so that elements $g_1, \cdots, g_N$ will be sampled uniformly at random from G and the prover will commit to the corresponding random path, i.e., to everything that binds all the edges from x to x_N. Then, the verifier will select a random *interruption*, that is, an index $i \in \{0, \cdots, N\}$. The prover has to show that the graph has been honestly obtained, i.e., it is the same regardless of the interruption (the challenge value). To do this, he will provide:

- all the edges from x to x_i: this can be done by revealing g_j, for $j \leqslant i$. If $i = 0$, then no edge of this type is provided;
- the path from x' to x_{i+1}: this can be done by revealing g'_{i+1}. If $i = N$, no edge of this type is provided;
- all the edges from x_{i+1} to x_N: this can be done by revealing g_j, for $i + 1 \leqslant j \leqslant N$. If $i = N$, no edge of this type is provided.

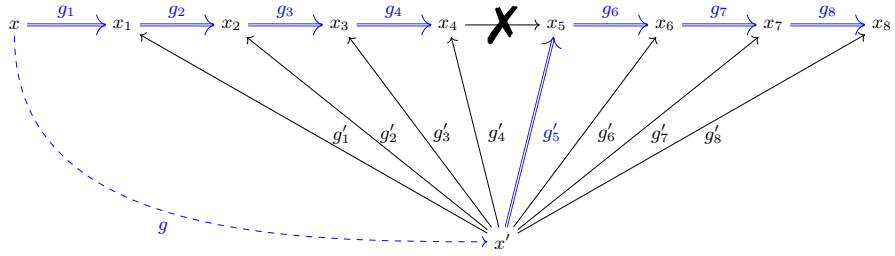
A proof of knowledge would then be of the form $P(i) = \left(g'_i, \{g_j\}_{j \neq i+1}\right)$. Note that, if $i = N$ (i.e., no interruption is actually asked), the prover instead reveals all labels $\{g_j\}_{1 \leqslant j \leqslant N}$. To sum up, the proof of knowledge is a function of the sole challenge index $i \in \{0, \cdots, N\}$, and has the following form

$$P(i) = \begin{cases} \left(g'_{i+1}, \{g_j\}_{j \neq i+1}\right) & \text{if } i < N, \\ \{g_j\}_{1 \leqslant j \leqslant N} & \text{if } i = N. \end{cases} \tag{2}$$

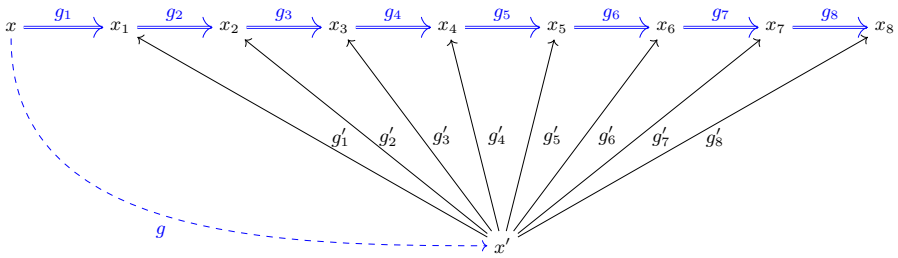
An example of how the proof of knowledge is constructed is shown in Figure 2. Note that, from the information provided in the proof as in (2), one can also recover all the edges g'_j with $j > i + 1$, but this is not useful to retrieve the secret g. Indeed, the vertices in the subgraph can be divided into groups: $V = \{x, x_1, \cdots, x_i\} \subset \mathcal{O}(x)$ and $V' = \{x_{i+1}, x_{i+1}, \cdots, x_N\} \subset \mathcal{O}(x')$. What the verifier does is checking that both V and V' are connected subgraphs. However, the prover never reveals an edge connecting a vertex in V with one in V': this guarantees that g cannot be recovered from the proof $P(i)$. A representation of the situation is depicted in Figure 3.



(a) Interruption $i = 0$



(b) Interruption $i = 4$



(c) Interruption $i = 8$

Fig. 2: Examples of proofs of knowledge, for three cases with different interruptions; the edges that are revealed by the prover are highlighted with double blue arrows.

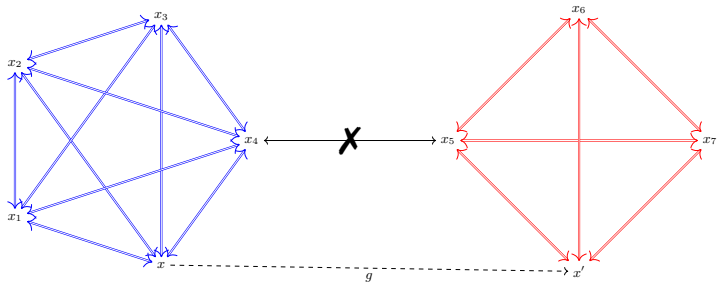


Fig. 3: Example of all the edges learned for the case of the interruption $i = 4$.

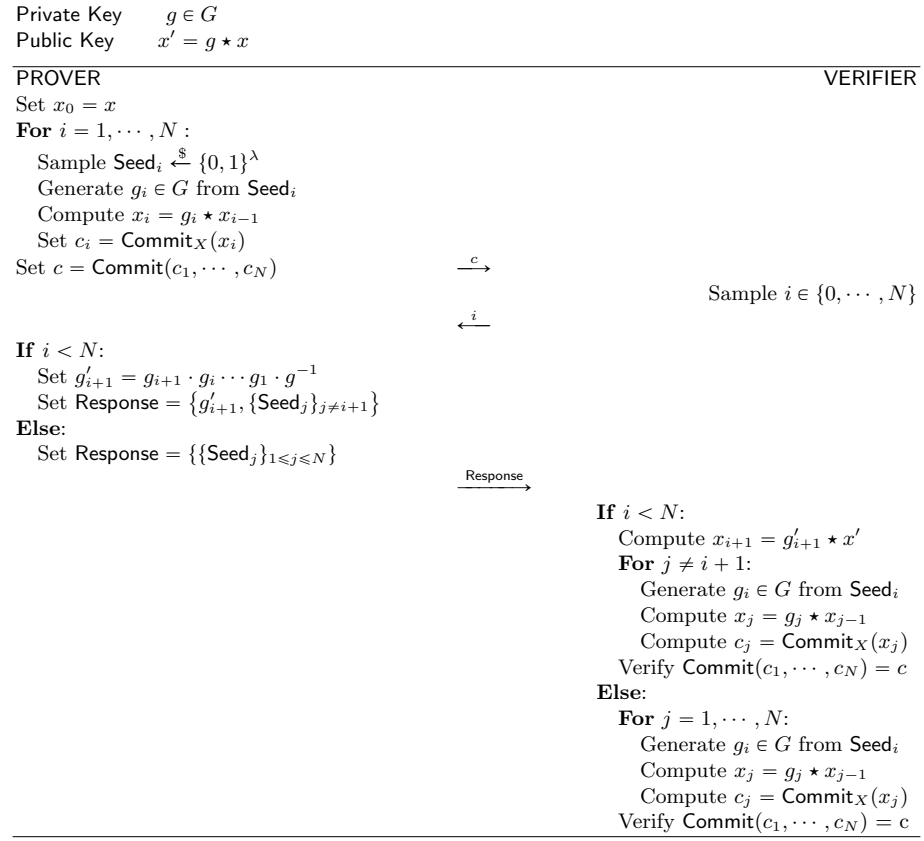


Fig. 4: A single round of the new ZK-ID protocol based on interruptions in the action subgraph.

The corresponding ZK-ID protocol is shown in Figure 4. Note that all group elements $g_1, \cdots, g_i$ can be compactly communicated using seeds, while for g'_i this is not possible. When $i = N$ (i.e., no interruption is asked), the proof is obtained by revealing all seeds, and each edge in the random path is verified. Using the standard technique of using a PRNG tree, revealing the random edges (i.e., edges labeled by some g_j) costs $\lambda \log_2(N)$ bits in case $i \neq N$, and only λ bits in case $i = N$.

We consider two commitment functions. While Commit can be any generic commitment function (say, a hash function), Commit_X is specific to how elements of X are represented. Since the element of X are the nodes of the graph we will also refer to Commit_X as node commitment. We generically assume that Commit_X requires a larger computational cost, with respect to Commit. For example, in the code-based case [3], each commitment to a code requires to perform a Gaussian elimination on a matrix of dimensions around 2^8.

Remark 2. When $N = 1$, the protocol collapses to a traditional Sigma protocol with challenge space of size 2. Indeed, regardless of i, the prover opens only one edge. If $i = 0$, the provided edge is the one g'_1, if $i = 1$, the verifier sends g_1. For what concerns seeds, the formulas we provided remain valid: for $i = 0$ no seed is required, while for $i = 1$ the response contains just one seed.

The security properties of the resulting protocol are proved in Appendix A.1. We report below the main result.

Theorem 1. *Assuming the hardness of GAIP (Problem 1), the protocol in Figure 4 is a secure Zero-Knowledge identification protocol, in the random oracle model, with soundness error:*

$$\varepsilon = \frac{1}{N+1}.$$

3.3 Multiple Key Pairs and Unbalanced Challenges

As in [3], the soundness of the protocol can be amplified by using more secrets to create the public values. Namely, the prover can choose M secrets $g^{(1)}, \cdots, g^{(M)}$ and compute the corresponding public values as $x'_i = g^{(i)} \star x$, for $i = 1, 2, \cdots, M$. We can also use the notation $g^{(0)} = e$ and $x'_0 = x$ if required. The action subgraph is computed in the same way, but now the verifier challenges the prover by specifying, together with the position of the interruption, also the public value from which the proof should be provided. In this way, the soundness error is reduced from $\frac{1}{N+1}$ to $\varepsilon = \frac{1}{1+MN}$.

Let ℓ_G denote the bit size of elements in G. When $i < N$, the prover has to reveal one element of G, together with $N - 1$ seeds. As we have already seen, using a standard PRNG tree, they can be efficiently communicated using only $\lambda \log_2(N)$ bits. Overall, one round with an interruption has a response of size $\ell_G + \lambda \log_2(N)$. Instead, when $i = N$ (i.e., no interruption), the response corresponds to the unique seed that has been used to generate the tree. This would require only λ bits: indeed, regardless of ℓ_G, rounds with no interruption have a response which is much smaller than that of rounds with an interruption. One can make use of this fact and consider, when applying the Fiat-Shamir transformation, challenge vectors with a fixed number of entries with value N. In other words, assuming t parallel repetitions are executed, the challenge vector would become a vector $\mathbf{b} \in \{0, \cdots, N\}^t$, such that there are only $w < t$ entries with values $< N$. In other words, there are always $t - w$ rounds in which the verifier chooses $i = N$ (i.e., no interruption), while an interruption is selected only in w rounds. To get the most from this optimization, one can use a unique PRNG tree to generate the seeds for each round. Namely, signature generation starts by sampling a *master seed* $\mathsf{MSeed} \xleftarrow{\$} \{0,1\}^\lambda$. This seed is used to generate t *round seeds* $\mathsf{RSeed}^{(i)}$ and then, for each round, $\mathsf{Seed}^{(i)}$ is used as the root of another PRNG tree, to obtain N $\mathsf{Seed}^{(i)}_j$. Let $J = \{i \in \{1, \cdots, t\} \,|b_i < N\}$: then, the seeds which would be required in the signature are of two types:

- the $t - w$ round seeds $\left\{\mathsf{RSeed}^{(i)}\right\}_{i \notin J}$;
- for each index $i \in J$, the $N - 1$ seeds $\left\{\mathsf{Seed}_j^{(i)}\right\}_{j \neq b_i + 1}$.

Communicating the round seeds requires only $\lambda w \log_2(t/w)$ bits while, for each round with $i \notin J$, communicating the $N - 1$ seeds takes $\lambda \log_2(N)$ bits. When considering multiple public values, we pair $\mathbf{b}$ with another vector $\mathbf{c} \in \{1, \cdots, M\}^t$, where c_i defines the public value from which the interruption is asked. If i is such that $b_i = N$, the value of c_i is useless. The resulting protocol preserves the special soundness property, with an overall soundness error of

$$\varepsilon = \frac{1}{\binom{t}{w}(MN)^w}.$$

Signature Size. The formulation with M public values and a fixed-weight challenge vector encompasses all the other ones, so we make the signature size explicit only for this version. The signature size, in bits, of the protocol is

$$\underbrace{w\lambda \log_2(t/w)}_{\text{Master seeds}} + w\Big(\underbrace{\lambda \log_2(N)}_{\text{Round seeds}} + \underbrace{\ell_G}_{\text{Non-random edge}} \Big) = \lambda w \log_2(tN/w) + w\ell_G. \quad (3)$$

4 Reducing Verification Time

Taking a closer look at the formulation in the previous section, one can see that there is no practical advantage in choosing $N > 1$.

In fact, consider two different instantiations of the protocol in Figure 4:

- protocol Π, having $N > 1$, t rounds, w rounds with $i < N$ and M public values.
- protocol Π', having $N' = 1$, $t' = tN$ rounds, $w' = w$ rounds with $i < N$ and $M' = M$ public values.

By (3), that the signature size is exactly the same for both protocols, apart from slight differences due to roundings. For protocol Π, both signature generation and verification require to compute Commit_X for tN times. For protocol Π', this function is called $t' = tN$ times: hence, for the running time, we expect to have no meaningful difference between the two protocols. Consider now the soundness errors, which are respectively

$$\varepsilon = \frac{1}{\binom{t}{w}(MN)^w} \text{ and } \varepsilon' = \frac{1}{\binom{t'}{w'}(M'N')^{w'}} = \frac{1}{\binom{tN}{w}M^w}.$$

Then, for any $w > 1$ and $N > 1$, we have that ε is strictly greater than ε'. Indeed, it is possible to show that

$$\binom{t}{w}(MN)^w < \binom{tN}{w}M^w \implies \binom{t}{w}N^w < \binom{tN}{w}.$$

To this end, observe that

$$\binom{t}{w} N^w = \left(\prod_{i=0}^{w-1} \frac{t-i}{w-i} \right) N^w = \prod_{i=0}^{w-1} \frac{N(t-i)}{w-i}$$
$$= \prod_{i=0}^{w-1} \frac{tN - iN}{w-i} < \prod_{i=0}^{w-1} \frac{tN-i}{w-i} = \binom{tN}{w}.$$

In the end, the two protocols have the same computational complexity (for both signing and verifying) and the same communication cost, but protocol Π' has lower soundness error. So, protocol Π' is preferable for all relevant aspects. To reduce the soundness error for protocol Π, we need to either use a larger value for t and/or for w, but this would result either in an increased computational complexity and/or in a larger communication cost. In practice, this shows that a straightforward approach is not helpful in improving signature size with respect to a classical Sigma protocol with challenge space of size 2; this is the case for the protocol presented in the previous section, as well as that of [16], where the latter corresponds to the protocol of Figure 4, with $N = 1$.

In the next section, we argue that the idea of using paths in the action graph is still useful, although in a somewhat unexpected aspect. Indeed, we show that we can greatly reduce the computational complexity on the verifier's side, at the cost of a (slight) increase in signature size. We are able to reduce the number of times a group actions needs to be computed, as well as the number of calls to Commit_X. Intuitively, this yields a considerable speed up the verification algorithm; for instance, in schemes like [3] and [7], the computational cost is dominated by the cost of evaluating the reduced row-echelon form of a matrix. In principle, the same holds for schemes based on isogenies, like [4], although in this case, the strategy used is radically different, since $l_G \simeq \lambda$, and there is therefore no incentive in unbalancing the challenges.

4.1 Skipping Edges in the Action Graph

In Figure 5 we describe a modified version of the protocol in Figure 4. For the sake of simplicity, the use of PRNG trees is made implicit. An example can be seen in Figure 6. We see that computation of the group action is skipped for x_1, x_2, x_3 and x_4: it is enough to combine $g_4 \cdot \ldots \cdot g_1$ and then apply the result to x. Analogously, we need to compute the group action and Commit_X only for x_4, x_5, x_6, x_7 and x_8.

Theorem 2. *Assuming the hardness of GAIP (Problem 1), the protocol in Figure 5 is a secure Zero-Knowledge identification protocol, in the random oracle model, with soundness error:*

$$\epsilon = \frac{1}{N+1}$$

Proof. As before, this proof is also found in the appendix (Appendix A.2), due to space constraints.

Private Key $g \in G$
Public Key $x' = g \star x$

PROVER		VERIFIER
Set $x_0 = x$		
For $i = 1, \cdots, N$:		
Sample $\mathsf{Seed}_i \xleftarrow{\$} \{0,1\}^\lambda$		
Generate $g_i \in G$ from Seed_i		
Compute $x_i = g_i \star x_{i-1}$		
Set $c_i = \mathsf{Commit}_X(x_i)$		
Set $c = \mathsf{Merkle.Root}(c_1, \cdots, c_N)$	$\xrightarrow{c}$	
		Sample $i \in \{0, \cdots, N\}$
	$\xleftarrow{i}$	
If $i < N$:		
Set $g'_{i+1} = g_{i+1} \cdot g_i \cdots g_1 \cdot g^{-1}$		
Set $\mathsf{Response} = \{g'_{i+1}, \{\mathsf{Seed}_j\}_{j \neq i+1}, \{c_l\}_{l<i}\}$		
Else:		
Set $\mathsf{Response} = \{\{\mathsf{Seed}_j\}_{1 \leqslant j \leqslant N}, \{c_l\}_{l<i}\}$	$\xrightarrow{\text{Response}}$	
		Retrieve g_k from Seed_k, for all $k \neq i+1$
		Compute $x_i = (g_i \cdots g_1) \star x$
		Compute $c_i = \mathsf{Commit}_X(x_i)$
		If $i < N$:
		Compute $x_{i+1} = g'_{i+1} \star x'$
		For $j > i+1$:
		Generate $g_i \in G$ from Seed_i
		Compute $x_j = g_j \star x_{j-1}$
		Compute $c_j = \mathsf{Commit}_X(x_j)$
		Verify $c = \mathsf{Merkle.Root}(c_1, \cdots, c_N)$

Fig. 5: A single round of the new ZK-ID protocol, based on interruptions in the action subgraph, that **skips** several part of the Commit_X evaluations.

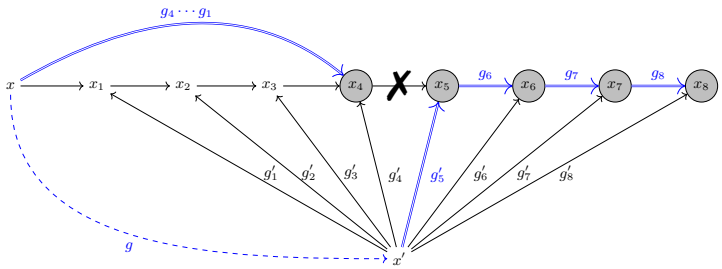


Fig. 6: Example of proof of knowledge with skipped edges, for $i = 4$. The nodes for which the verifier has to compute Commit_X are given by gray circles.

Computational Complexity. On the prover's side, this new version of the protocol does not modify anything. Indeed, they still need to compute both the group action and the Commit_X function for tN times. However, we can achieve significant reductions on the verifier's side. Indeed, all rounds with no interruption require to compute only one group action, as well as only one Commit_X. For the rounds with an interruption, on average, group actions and node commitments are computed, on average, only around $N/2 + 1$ times. In fact the challenge

0 requires N evaluations, the challenges $i > 0$ requires $N + 1 - i$; the sum is $N + N(N+1)/2$ that divided by $N + 1$ can be rounded to $1 + N/2$. Thus, in the end, the average number of times node commitments are computed during verification is

$$\#_{\mathsf{Commit}_X} = (t - w) + w\left(\frac{N}{2} + 1\right) = t + w\frac{N}{2}.$$

The ratio between the number of node commitments during verification and during the signature generations is the following:

$$\eta = \frac{\#_{\mathsf{Commit}_X}}{tN} = \frac{t + w\frac{N}{2}}{tN} = \frac{1}{N} + \frac{w}{2t}$$

As we shall see in the next section, this implies a significant reduction for the computational complexity. Yet, before giving details, we render the protocol to a signature scheme and propose a final optimization, which allows to further save some communication cost. We also provide a precise estimate for the average signature size.

4.2 Trading Signature Size with Computational Efficiency

Let $c_j^{(u)}$ denote the j-th commitment for the i-th round. In every round of the new protocol, the verifier recomputes $c_N^{(u)}$. Hence, the check on $c_N^{(u)}$ can be postponed to the end of the verification procedure; for this, it is enough that the prover commits to $\mathsf{Commit}\big(c_N^{(1)}, \cdots, c_N^{(t)}\big)$. Starting from this consideration, we see that the Merkle tree can be built, more conveniently, using only the intermediate nodes in the graph (e.g., the nodes $x_1^{(u)}, \cdots, x_{N-1}^{(u)}$). Let $\mathcal{T}^{(u)}$ be the *round Merkle tree* constructed from $x_1^{(u)}, \cdots, x_{N-1}^{(u)}$, with root $h^{(u)}$. Furthermore, we consider a *master Merkle tree* $\overline{\mathcal{T}}$ built from $h^{(1)}, \cdots, h^{(t)}$, with root $\bar{h}$. The prover computes the challenge $\mathsf{Challenge}\big(\{c_N^{(u)}\}_{1 \leqslant u \leqslant t}, \bar{h}\big)$. We parse the challenge as a pair of vectors $\big(i^{(1)}, \cdots, i^{(t)}\big) \times \big(z^{(1)}, \cdots, z^{(t)}\big) \in \{0, \cdots, N\}^t \times \{1, \cdots, M\}^t$. The prover will also sample a unique *master seed* $\mathsf{MSeed} \xleftarrow{\$} \{0,1\}^\lambda$, which is used to derive the *round seeds* $\overline{\mathsf{Seed}}^{(1)}, \cdots, \overline{\mathsf{Seed}}^{(t)} = \mathsf{SeedTree}(\mathsf{MSeed})$. For round u, the u-th round seed is further expanded into N seeds as $\mathsf{Seed}^{(1)}, \cdots, \mathsf{Seed}^{(N)} = \mathsf{SeedTree}(\overline{\mathsf{Seed}}^{(u)})$.

If round u has an interruption (i.e., $i^{(u)} = N$), the verifier will locally recompute some of the intermediate nodes, namely, all nodes $x_j^{(u)}$ with $j \geqslant i^{(u)}$. We write $x_0^{(u)} = x$ so that this is true also for $i^{(u)} = 0$. The verifier will need the proof that these nodes are the same the prover commited to. This can be provided with the *round Merkle proof* which we denote by $\mathsf{MerkleProof}^{(u)} = \mathcal{T}^{(u)}.\mathsf{Proof}\big(1, \cdots, i^{(u)} - 1\big)$. Also, the verifier will need the seeds to generate the group elements $g_1^{(u)}, \cdots, g_{i^{(u)}}^{(u)}, g_{i^{(u)}+2}^{(u)}, \cdots, g_N^{(u)}$. The prover will provide the *seed path* $\mathsf{SeedPath}^{(u)} = \mathsf{SeedPath}\big(i^{(u)} + 2, \cdots, N, \overline{\mathsf{Seed}}^{(u)}\big)$.

All rounds without an interruption can be handled together. Let U be the set of indices for rounds without interruption, that is, $U = \{u \mid i^{(u)} = N\}$. The prover will need all round seeds for rounds indexed by U; they can be efficiently communicated through their path in the PRNG seed tree generated by MSeed; the path is computed by the function $\mathsf{SeedPath}(U, \mathsf{MSeed})$.

For the sake of completeness, in Figure 7 we report the version of the protocol with this last modification. In the figure, we have included all the relevant details, such as using salt and indexing the commitments (to prevent pre-image collisions attacks). Also, we have used Hash to indicate the generic Commit function. Before deriving the signature size we consider that, for the trees we are employing, sizes of proofs and paths are smaller: this is due to the fact that the seeds we do not need to reveal, or the nodes for which we need to provide a Merkle proof, are always adjacent.

Size of Merkle Proofs and Seed Paths. For the master Merkle proof, we cannot make any assumption on the positions of the nodes for which the proof is provided. Taking into account multiple paths, the size of the master Merkle proof would be $2w\lambda \log_2(t/w)$, and, for the master seed path, the number of required bits is $w\lambda \log_2(t/w)$. Instead, for round Merkle proofs and seed paths, we know that the relevant nodes are always adjacent; this allows to reduce sizes of proofs and paths. We describe what happens with Merkle proofs; for seed paths, the reasoning is analogous.

Let us first start with the simple example of an interruption on the middle of the action graph (i.e., $i = \frac{N}{2}$): in this case, the round Merkle tree is made by two sub-trees, each with $N/2$ nodes in the base layer. The nodes for the tree on the right are computed by the verifier, while the prover provides the Merkle proof for the tree on the left. Yet, this proof would only be constituted by a single digest. We now tackle the general case, and consider an interruption in a generic position i: in this case, the number of nodes for which we need to provide the Merkle proof is

$$s(i) = \begin{cases} 0 & \text{if } i < 2, \\ i - 1 & \text{if } i \leqslant 2. \end{cases}$$

Let $\mathbf{s}(i)$ denote the binary representation of $s(i)$: then, the Merkle proof will contain a unique hash for each one in $\mathbf{s}(i)$. Let $\mathrm{wt}\big(s(i)\big)$ be the Hamming weight of $s(i)$; since each specific value for i happens with probability $1/N$, the average size of the round Merkle proof is

$$\ell_{\mathsf{Merkle}}(N) = \frac{2\lambda}{N} \sum_{i=0}^{N-1} \mathrm{wt}\big(s(i)\big). \tag{4}$$

Private Key $g^{(1)}, \cdots, g^{(M)} \in G$
Public Key set elements $x'_i = g_i \star x$, for $i = 1, \cdots, M$

//Generate commitments
Sample $\mathtt{Salt}, \mathtt{MSeed} \xleftarrow{\$} \{0;1\}^\lambda$
Generate $\left\{\overline{\mathsf{Seed}}^{(u)}\right\}_{1 \leqslant u \leqslant t} = \mathsf{SeedTree}(\mathtt{Mseed}, \mathtt{Salt})$ //Round seeds from master seed
Set $\{x_0^{(u)}\}_{1 \leqslant u \leqslant t} = x$
For $u = 1, \cdots, t$:
  Generate $\left\{\mathsf{Seed}_j^{(u)}\right\}_{1 \leqslant j \leqslant N} = \mathsf{SeedTree}(\overline{\mathsf{Seed}}^{(u)}, \mathtt{Salt})$ //Seeds for round u
  For $j = 1, \cdots, N-1$:
    Sample $g_j^{(u)} = \mathsf{PRNG}\left(\mathsf{Seed}_j^{(u)}\right)$
    Compute $x_j^{(u)} = g_j^{(u)} \star x_{j-1}^{(u)}$
    Compute $c_j^{(u)} = \mathsf{Commit}_X\left(x_j^{(u)}, \mathtt{Salt}, u, j\right)$
  Set $\mathcal{T}^{(u)} = \mathsf{MerkleTree}\left(c_1^{(u)}, \cdots, c_{N-1}^{(u)}\right)$ //Round Merkle tree
  Compute $h^{(u)} = \mathcal{T}^{(u)}.\mathsf{Root}$ //Round Merkle root
  Sample $g_N^{(u)} = \mathsf{PRNG}\left(\mathsf{Seed}_N^{(u)}\right)$
  Compute $x_N^{(u)} = g_N^{(u)} \star x_{N-1}^{(u)}$
  Compute $c_N^{(u)} = \mathsf{Commit}_X\left(x_N^{(u)}, \mathtt{Salt}, u, N\right)$
Set $\bar{\mathcal{T}} = \mathsf{MerkleTree}\left(\left\{h^{(u)}\right\}_{1 \leqslant u \leqslant t}\right)$ //Master Merkle tree
Compute $h = \bar{\mathcal{T}}.\mathsf{Root}$ //Master Merkle root
Compute $c = \mathsf{Hash}\left(\left\{c_N^{(u)}\right\}_{1 \leqslant u \leqslant t}, h\right)$
//Generate challenge vector
Set $\left\{i^{(u)}, z^{(u)}\right\}_{1 \leqslant u \leqslant t} = \mathsf{Challenge}(c, \mathtt{Salt}, m)$
Set $U = \{u \mid i^{(u)} = N\}$ //Indices of rounds with no interruption
Compute $\mathtt{MSeedPath} = \mathsf{SeedPath}(U, \mathtt{Mseed}, \mathtt{Salt})$ //Master seed path
Compute $\mathtt{MMerkleProof} = \bar{\mathcal{T}}.\mathsf{Proof}\,(U)$ //Master Merkle proof
For u such that $i^{(u)} < N$:
  Set $g'^{(u)}_{i^{(u)}+1} = g^{(u)}_{i^{(u)}+1} \cdot g^{(u)}_{i^{(u)}} \cdots g^{(u)}_1 \cdot g^{-1}_{z^{(u)}}$
  Set $\mathtt{SeedPath}^{(u)} = \mathsf{SeedPath}\left(i^{(u)}, \overline{\mathsf{Seed}}^{(u)}, \mathtt{Salt}\right)$ //Round seed path
  Compute $\mathtt{MerkleProof}^{(u)} = \mathcal{T}^{(u)}.\mathsf{Proof}\left(i^{(u)}, \cdots, N-1\right)$ //Round Merkle proof

$$\xrightarrow{\mathtt{Salt}, c, \left\{g'^{(u)}_{i^{(u)}+1}, \mathtt{SeedPath}^{(u)}, \mathtt{MerkleProof}^{(u)}\right\}_{u \notin U}, \mathtt{MSeedPath}, \mathtt{MMerkleProof}}$$

Fig. 7: The generic signature algorithm based on the action subgraph with skipped node commitments.

In practice, this is always slightly less than $\lambda \log_2(N)$. For instance, if $N = 128$, we get that the average size is 6.9λ. For the seed path the reasoning is the same, but now each node in the tree is a binary string with length λ; so, the average size of the round seed path is

$$\ell_{\mathsf{Seed}}(N) = \frac{\lambda}{N} \sum_{i=0}^{N-1} \mathrm{wt}\big(s(i)\big). \tag{5}$$

This, in practice, is always slightly less than $\frac{\lambda}{2} \log_2(N)$ bits.

Private Key $g^{(1)}, \cdots, g^{(M)} \in G$
Public Key set elements $x'_i = g_i \star x$, for $i = 1, \cdots, M$
Input signature $\mathtt{Salt}, c, \left\{g'^{(u)}_{i^{(u)}+1}, \mathtt{SeedPath}^{(u)}, \mathtt{MerkleProof}^{(u)}\right\}_{u \notin U}, \mathtt{MSeedPath}, \mathtt{MMerkleProof}$

Set $\left\{i^{(u)}, z^{(u)}\right\}_{1 \leqslant u \leqslant t} = \mathsf{Challenge}(c, \mathtt{Salt}, m)$ //Retrieve challenges
Set $U = \{u \mid i^{(u)} = N\}$ //Indices of rounds with no interruption
For $u = 1, \cdots, t$:
  If $i^{(u)} = N$: //Ephemeral case
    Retrieve $\overline{\mathtt{Seed}}^{(u)}$ from $\mathtt{MSeedPath}$
    Retrieve $\left\{\mathtt{Seed}_j^{(u)}\right\}_{1 \leqslant j \leqslant N} = \mathsf{SeedTree}(\overline{\mathtt{Seed}}^{(u)}, \mathtt{Salt})$
    Sample $g_j^{(u)} = \mathsf{PRNG}\left(\mathtt{Seed}_j^{(u)}\right)$ for $j = 1, \ldots N$
    Compute $x_N^{(u)} = \left(g_N^{(u)} \cdots g_1^{(u)}\right) \star x$
    Compute $c_N^{(u)} = \mathsf{Commit}_X\left(x_N^{(u)}, \mathtt{Salt}, u, N\right)$
  Else : //Non ephemeral case
    Retrieve $\left\{\mathtt{Seed}_j^{(u)}\right\}_{j \neq i^{(u)}+1}$ from $\mathtt{SeedPath}^{(u)}$
    Sample $g_j^{(u)} = \mathsf{PRNG}\left(\mathtt{Seed}_j^{(u)}\right)$ for $j \neq i^{(u)} + 1$
    Compute $x_{i^{(u)}}^{(u)} = \left(g_{i^{(u)}} \cdots g_1\right) \star x$
    Compute $c_{i^{(u)}}^{(u)} = \mathsf{Commit}_X\left(x_{i^{(u)}}^{(u)}, \mathtt{Salt}, u, i^{(u)} + 1\right)$
    Compute $x_{i^{(u)}+1}^{(u)} = g'^{(u)}_{i^{(u)}+1} \star x'_{z^{(u)}}$
    Compute $c_{i^{(u)}+1}^{(u)} = \mathsf{Commit}_X\left(x_{i^{(u)}+1}^{(u)}, \mathtt{Salt}, u, i^{(u)} + 1\right)$
    For $i^{(u)} + 1 < j \leqslant N$:
      Compute $x_j^{(u)} = g'^{(u)}_{i^{(u)}+1} \star x_{j-1}^{(u)}$
      Compute $c_j^{(u)} = \mathsf{Commit}_X\left(x_j^{(u)}, \mathtt{Salt}, u, j\right)$
  Compute $h^{(u)} = \mathcal{T}^{(u)}.\mathsf{Root}$ from $\mathtt{MerkleProof}^{(u)}$ and $\left\{c_j^{(u)}\right\}_{j \geqslant i^{(u)}}$
Compute $h = \bar{\mathcal{T}}.\mathsf{Root}$ from $\mathtt{MMerkleProof}$ and $\left\{h^{(u)}\right\}_{u \notin U}$ //Master Merkle root
Compute $c_{\mathsf{Ver}} = \mathsf{Hash}\left(\left\{c_N^{(u)}\right\}_{1 \leqslant u \leqslant t}, h\right)$
Return Valid if $c_{\mathsf{Ver}} = c$

Fig. 8: The generic signature verification algorithm based on the action subgraph with skipped node commitments.

Signature Size. We are finally ready to derive the corresponding average signature size, which is

$$w\Big(\underbrace{\ell_{\mathsf{Seed}}(N)}_{\text{Round seeds}} + \underbrace{\ell_{\mathsf{Merkle}}(N)}_{\text{Round Merkle proof}} + \underbrace{\ell_G}_{\text{Non random edge}}\Big) + \underbrace{w\lambda\log_2(t/w)}_{\text{Master seed path}} + \underbrace{2w\lambda\log_2(t/w)}_{\text{Master Merkle proof}} + \underbrace{3\lambda}_{\text{Hash and Salt}} . \quad (6)$$

References

[1] M. Abdalla et al. "From Identification to Signatures Via the Fiat–Shamir Transform: Necessary and Sufficient Conditions for Security and Forward-Security". In: *IEEE Transactions on Information Theory* 54.8 (2008), pp. 3631–3646.

[2] N. Alamati et al. “Cryptographic Group Actions and Applications”. In: *ASIACRYPT*. Springer. 2020, pp. 411–439.

[3] A. Barenghi et al. “LESS-FM: fine-tuning signatures from the code equivalence problem”. In: *Post-Quantum Cryptography: 12th International Workshop, PQCrypto 2021, Daejeon, South Korea, July 20–22, 2021, Proceedings 12*. Springer. 2021, pp. 23–43.

[4] W. Beullens, T. Kleinjung, and F. Vercauteren. “CSI-FiSh: efficient isogeny based signatures through class group computations”. In: *Advances in Cryptology–ASIACRYPT 2019: 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8–12, 2019, Proceedings, Part I*. Springer. 2019, pp. 227–247.

[5] J.-F. Biasse et al. “LESS is More: Code-Based Signatures Without Syndromes”. In: *AFRICACRYPT*. Ed. by A. Nitaj and A. Youssef. Springer, 2020, pp. 45–65.

[6] W. Castryck et al. “CSIDH: An Efficient Post-Quantum Commutative Group Action”. In: *Advances in Cryptology – ASIACRYPT 2018*. Springer. 2018, pp. 395–427.

[7] T. Chou et al. “Take your MEDS: Digital Signatures from Matrix Code Equivalence”. In: *AFRICACRYPT* (2023).

[8] W. Diffie and M. Hellman. “New directions in cryptography”. In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: 10.1109/TIT.1976.1055638.

[9] L. Ducas and W. van Woerden. “On the lattice isomorphism problem, quadratic forms, remarkable lattices, and cryptography”. In: *Advances in Cryptology–EUROCRYPT 2022: 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30–June 3, 2022, Proceedings, Part III*. Springer. 2022, pp. 643–673.

[10] T. Elgamal. “A public key cryptosystem and a signature scheme based on discrete logarithms”. In: *IEEE Transactions on Information Theory* 31.4 (1985), pp. 469–472. DOI: 10.1109/TIT.1985.1057074.

[11] T. Feneuil, A. Joux, and M. Rivain. “Shared permutation for syndrome decoding: New zero-knowledge protocol and code-based signature”. In: *Designs, Codes and Cryptography* 91.2 (2023), pp. 563–608.

[12] T. Feneuil, A. Joux, and M. Rivain. “Syndrome Decoding in the Head: Shorter Signatures from Zero-Knowledge Proofs”. In: *Advances in Cryptology – CRYPTO 2022*. Vol. 13508. LNCS. Springer, 2022, pp. 541–572.

[13] L. D. Feo and S. Galbraith. “SeaSign: Compact Isogeny Signatures from Class Group Actions”. In: *EUROCRYPT 2019*. Ed. by Y. Ishai and V. Rijmen. Vol. 11478. Lecture Notes in Computer Science. Springer, 2019, pp. 759–789. DOI: 10.1007/978-3-030-17659-4_26. URL: https://doi.org/10.1007/978-3-030-17659-4%5C_26.

[14] A. Fiat and A. Shamir. “How to prove yourself: Practical solutions to identification and signature problems”. In: *CRYPTO*. Springer. 1986, pp. 186–194.

[15] S. Gueron, E. Persichetti, and P. Santini. "Designing a Practical Code-Based Signature Scheme from Zero-Knowledge Proofs with Trusted Setup". In: *Cryptography* 6.1 (2022), p. 5.
[16] A. Joux. *MPC in the head for isomorphisms and group actions*. Cryptology ePrint Archive, Paper 2023/664. `https://eprint.iacr.org/2023/664`. 2023. URL: `https://eprint.iacr.org/2023/664`.
[17] D. Kales and G. Zaverucha. "Improving the performance of the picnic signature scheme". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2020), pp. 154–188.
[18] J. Katz, V. Kolesnikov, and X. Wang. "Improved non-interactive zero knowledge with applications to post-quantum signatures". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 525–537.
[19] NIST. *Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process*. URL: `https://csrc.nist.gov/projects/pqc-dig-sig/standardization/call-for-proposals`. 2023.
[20] G. Tang et al. "Practical post-quantum signature schemes from isomorphism problems of trilinear forms". In: *Advances in Cryptology–EUROCRYPT 2022: 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30–June 3, 2022, Proceedings, Part III*. Springer. 2022, pp. 582–612.

A Proofs of Security

We provide here the proofs of security that have been omitted in the main body of the paper. Recall also that, by using Fiat-Shamir [1, 14], we can transform the described ZK-ID protocols into signature schemes secure against chosen-message attacks in the random oracle model.

A.1 Protocol of Figure 4

The completeness of the protocol is trivial from the description in Section 3.1; we therefore proceed to prove Zero-Knowledge and 2-Special Soundness.

Zero Knowledge. We want to show that a simulator that knows, in advance, the challenge value can produce a valid transcript which is statistically distributed as the one that would be produced by an honest prover. To this end, we consider a simulator that, on input $i \in \{0, \cdots, N\}$, proceeds as follows:

- if $i = N$, it samples seeds $\mathtt{Seed}_1, \cdots, \mathtt{Seed}_N$ and generates the random path and the commitments; then, outputs all the seeds. Obviously, this output is statistically distributed as the one of the honest prover;
- if $i < N$, it selects random seeds $\mathtt{Seed}_1, \cdots, \mathtt{Seed}_i$ and uses them to compute $\{g_j\}_{j \leqslant i}$ and $\{x_j\}_{j \leqslant i}$ inductively as $x_j = g_j \star x_{j-1}$. Then, the simulator selects uniformly at random $\widetilde{g} \in G$ and uses it to compute $\widetilde{x}_{i+1} = \widetilde{g} \star x'$. Finally,

if $i+1 < N$, it selects the remaining random seeds $\texttt{Seed}_{i+2}, \cdots, \texttt{Seed}_N$, it uses them to compute $\{g_j\}_{j \geqslant i+2}$ and to inductively compute $\widetilde{x}_{i+2} = g_{i+2} \star \widetilde{x}_{i+1}, \ldots, \widetilde{x}_N = g_N \star \widetilde{x}_{N-1}$. The simulator hashes $x_1, \ldots, x_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_N$ to produce the commitments, which indistinguishable from those of a honest execution. As output, the simulator provides $\texttt{Response} = \{\widetilde{g}, \{\texttt{Seed}_j\}_{j \neq i+1}\}$. The elements $\widetilde{g}$ and $\{\texttt{Seed}_j\}_{j \neq i+1}$ are independent and uniformly distributed over G. This is also the case of the response of an honest prover because g'_{i+1} has the form

$$g'_{i+1} = \underbrace{g_{i+1}}_{\substack{\text{Secret and} \\ \text{ephemeral}}} \cdot \underbrace{g_i \cdot g_{i-1} \cdots g_1}_{\substack{\text{Public and} \\ \text{ephemeral}}} \cdot \underbrace{g^{-1}}_{\text{Secret}} .$$

Since g_{i+1} is independent from the $\{g_j\}_{j \leqslant i+1}$, g'_{i+1} is also distributed uniformly at random independently from $\{\texttt{Seed}_j\}_{j \neq i+1}\}$. One can immediately check that the simulator's response leads to a valid transcript.

2-*Special Soundness.* Let us consider two accepting transcripts with the same commitment, but different challenge values, say, $i_0 \neq i_1$. Without loss of generality, let $i_0 < i_1$. We show that there is an efficient extractor that can compute a witness. The idea of the proof is that of showing that, using the edges contained in the two transcripts, the extractor is always able to obtain the element g such that $x' = g \star x$.

From the response to i_1, using the walk $x \mapsto x_{i_1}$, we get $\tilde{r}$ such that $c_{i_0+1} = \mathsf{Commit}_X(\tilde{r} \star x)$ since $i_0 + 1 \leqslant i_1$. From the response to i_0 we have g'_{i_0+1} that satisfies $c_{i_0+1} = \mathsf{Commit}_X(g'_{i_0+1} \star x')$. By the collision resistance of the Commit_X function we have that $\tilde{r} \star x = g'_{i_0+1} \star x'$ that implies $x' = (g'^{-1}_{i_0+1}\tilde{r}) \star x$, thus we have obtained the secret g and solved the GAIP.

The 2-Special Soundness immediately implies that the soundness error is the reciprocal of the cardinality of the challenge space $(N+1)$, as claimed by the Theorem.

A.2 Protocol of Figure 5

As before, the completeness of the protocol is trivial from the description in Section 3.1. To prove Zero-Knowledge, we can use the same simulator of Appendix A.1 to generate the seeds and the group elements. To generate the commitment values $\{c_l\}_{l<i}$, we can simply consider random values on the Commit_X co-domain, since we are in the random oracle model.

The 2-Special Soundness can also be proven with similar strategies as in Appendix A.1, with the difference of focusing on the final tail. For clarity, we still include here a detailed proof with an example.

2-Special Soundness. Let us consider two accepting transcripts with the same commitment, but different challenge values, say, $i_0 \neq i_1$. Without loss of generality, let $i_0 < i_1$. We show that there is an efficient extractor that can compute a

witness. The idea of the proof is that of showing that, using the edges contained in the two transcripts, the extractor is always able to obtain the element g such that $x' = g \star x$.

By the collision resistance of $\mathsf{Merkle.Root}$, we get the same committed values $c_1, ..., c_N$. From the response to i_1, we get $\tilde{r}$ such that $c_{i_1} = \mathsf{Commit}_X(\tilde{r} \star x)$. From the response to i_0, we have $\hat{r} = g'_{i_0+1}$ for $i_1 = i_0 + 1$ or $\hat{r} = g_{i_1} \cdots g_{i_0+2} \cdot g'_{i_0+1}$ otherwise (observe that in this case $i_1 > i_0 + 1$), that satisfies $c_{i_1} = \mathsf{Commit}_X(\hat{r} \star x')$. By the collision resistance of the Commit_X function we have that $\tilde{r} \star x = \hat{r} \star x'$ that implies $x' = (\hat{r}^{-1}\tilde{r}) \star x$, thus we have obtained the secret g and solved GAIP.

Once again, the 2-Special Soundness immediately implies that the soundness error is the reciprocal of the cardinality of the challenge space, as claimed by the Theorem.

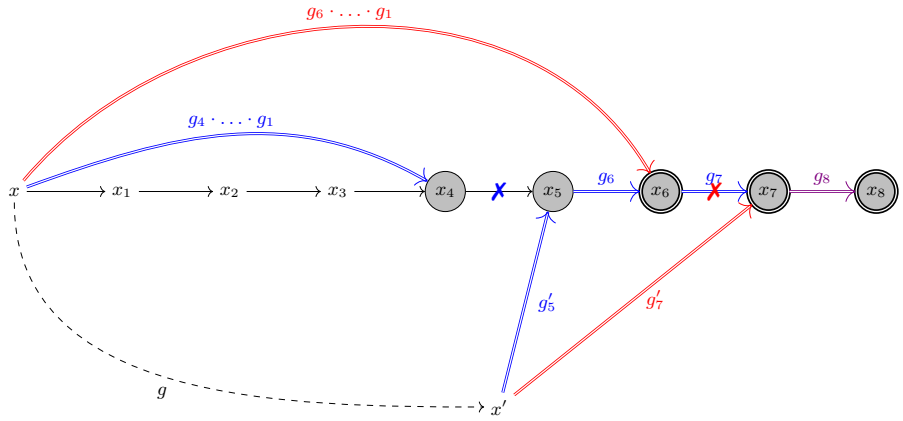
Example. A graphical representation of the extractor's behavior is given in Figure 9. For instance, for the case $i_0 = 4$ and $i_1 = 6$, which is reported in the top half of the figure, the extractor can recover the secret g using the following path $x \mapsto x_6 \mapsto x_5 \mapsto x'$. For the case $i_0 = 4$, $i_1 = 8$, the secret can be extracted from the path

$$x \mapsto x_8 \mapsto x_7 \mapsto x_6 \mapsto x_5 \mapsto x'.$$

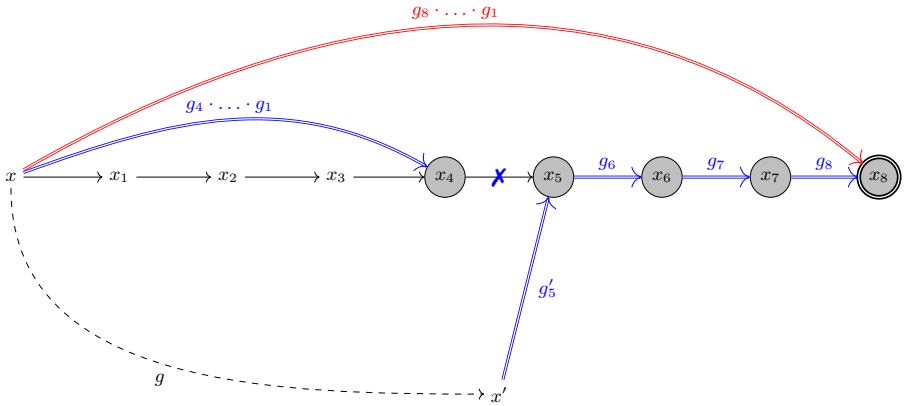
Note that commitment verification after the interruption plays a crucial role, since his implies that there are common nodes in the paths associated to the two transcripts. Without this, the knowledge extractor would not work. existence of these paths For instance, let us consider again the case depicted in Figure (a) and assume that only the final element x_8 is verified. Elements for the transcript i_1 are denoted with *. In this case, the extractor would know the following two paths

$$x' \mapsto x_5 \mapsto x_6 \mapsto x_7 \mapsto x_8,$$
$$x' \mapsto x_7^* \mapsto x_8^*.$$

Notice that the transcripts also contains paths of the form $x \mapsto x_4$ and $x \mapsto x_6^*$, but there is no guarantee that indeed $x_6 = x_6^*$. The extractor knows only that $x_8 = x_8^*$ (since the last commitment is checked in both transcripts), but this would give only information about a circular loop that starts and ends in x'.



(a) Case $i_0 = 4$, $i_1 = 6$



(b) Case $i_0 = 4$, $i_1 = N = 8$

Fig. 9: Representation of the action graph available to the knowledge extractor, for two pairs of accepting transcripts. Colors have the following meaning: blue is associated to knowledge provided only in transcript i_0, red to knowledge provided only in transcript i_1, violet (red + blue) to knowledge that is common for both transcripts. Analogously, nodes with double outline are commitments that are common for both transcripts.