New Straight-Line Extractable NIZKPs for Cryptographic Group Actions

Andrea Flamini[1], Federico Pintore[2], Edoardo Signorini[3], and Giovanni Tognolini[2]

[1] Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
[2] Università di Trento, Trento, Italy
[3] Telsy, Turin, Italy

Abstract. Group-action-based Sigma protocols have proven to be truly flexible for the design of advanced cryptosystems. When they are compiled via the Fiat-Shamir transform, the resulting NIZKPs can be efficiently optimized using standard techniques within the fixed-weight framework. Unfortunately, the same techniques are not as effective when they are combined with straight-line extractable NIZKPs (e.g. Fischlin, CRYPTO '05), a key building block in the design of complex protocols.
This work introduces the GAO (Group Action Oriented) transform, a new generic compiler that produces straight-line extractable NIZKPs from Sigma protocols while significantly simplifying the analysis of the fixed-weight framework. The GAO transform is then optimized in two different ways. First, by defining a collision predicate that results to be even more effective combined with GAO than with the Fischlin transform, for which was originally introduced (Kondi and shelat, ASIACRYPT '22). Second, adopting a technique, which we call Stretch-and-Compress (SC), that enables trading a smaller number of expensive operations for a larger number of inexpensive ones, which surprisingly leads to a reduction of both the computational complexity and the proof size. Our final transform, obtained from GAO applying the two optimizations, is called SC-Coll-GAO.
The practical advantages of the SC-Coll-GAO transform are theoretically motivated and concretely tested on the Sigma protocol underlying the Fiat-Shamir digital signature LESS, a code-based candidate that recently advanced to the second round of the NIST standardization process specifically purposed for post-quantum signatures. Remarkably, when compared to the Fiat-Shamir LESS baseline, SC-Coll-GAO incurs a computational cost increase by 50–60%, while signature sizes grow by only 10–20%.

1 Introduction

Non-Interactive Zero-Knowledge Proofs of Knowledge (in the following simply NIZKPs) are cryptographic primitives that allow a prover to prove knowledge of a secret to a verifier in a non-interactive way, without leaking any information about the secret itself. Informally, these proofs admit the existence of an algorithm, called extractor, that, having access to a prover who generates valid proofs, can

learn the secret. A standard way to build NIZKPs consists in compiling Sigma protocols using general-purpose transforms [39,64,62,72,40], with the Fiat-Shamir transform being the most common.

The reason behind the success of Fiat-Shamir NIZKPs is that they are generally very compact and computationally efficient, since they are obtained by computing a single transcript of the underlying Sigma protocol [68,18,58], as long as its challenge space is sufficiently large. However, extractors for Fiat-Shamir NIZKPs must be given rewindable access to the prover to extract the secret [64]. This reliance on rewinding introduces severe limitations that motivate the development of efficient straight-line extractable NIZKPs.

Straight-Line Extractable NIZKPs. Straight-line extractability [40] is a strengthened extraction requirement for NIZKPs guaranteeing that an extractor can recover the secret given only (1) the random-oracle queries issued by a successful prover and (2) the generated proof, without rewinding the prover. In the random oracle model, this property eliminates three fundamental limitations of rewinding-based extraction.

First, rewinding is incompatible with composable security frameworks such as the UC model [20], necessitating straight-line extractable NIZKPs for protocols proven secure in these settings [57,24,23]. Schemes relying on straight-line extractability to achieve UC security include threshold signatures [56,32,33] and anonymous credential systems [47]. The need for straight-line extractability also arises when defining security properties for advanced primitives such as blind signatures [53,52] or anonymous credentials supporting blind issuance of attributes [22,41], where the security experiment must be aware of the attributes blindly issued to the adversary.

Second, straight-line extractability simplifies the security analysis of multi-party protocols where reductions must rewind multiple corrupted actors controlled by an adversary. Nested rewinding prevents proving concurrent security of protocol executions [69,63] due to super-polynomial reduction time. A concrete example is the recently proposed threshold LESS signature [7], whose security analysis requires straight-line extractable NIZKPs, currently forcing the use of alternatives to the Fiat-Shamir transform that lead to a significant efficiency overhead.

Third, security of Fiat-Shamir NIZKPs in the plain random oracle model incurs a quadratic tightness loss due to the rewinding needed to extract the secret, as formalized by the Forking Lemma [11,64]. This means that primitives obtained by applying Fiat-Shamir might have fewer bits of security than claimed [28]. While this security loss is often ignored in practice, defining efficient NIZKPs and signatures [12,42] with tight reductions to the security of the underlying assumption is necessary to argue provable security.

As straight-line extractability addresses the limitations related to tightness, composability, and concurrency described above, it is widely used as a building block in the design of cryptographic primitives and protocols, making it an active field of research.

Existing Solutions. Pass [62] and later Unruh [72] proposed general-purpose transforms to turn Sigma protocols into straight-line extractable NIZKPs. They are both based on the cut-and-choose technique, used to define a new Sigma protocol which the Fiat-Shamir transform is applied to. Within the new Sigma protocol, the prover commits to multiple transcripts having the same first message and different predetermined challenges, then they open the commitment specified by the verifier's challenge. In both transforms, the proof incurs an undesirable overhead of at least λ^2 bits in the proof size, where λ is the security parameter.

An alternative eliminating size overhead is the Fischlin transform [40,24,54]. Its compactness makes it the most commonly used approach when straight-line extractability is needed. The Fischlin transform is based on the execution of a proof-of-work where the prover generates r first messages of the underlying Sigma protocol and, for each of them, looks for a challenge satisfying a hash-inversion predicate. This forces the prover to send transcripts for the same first message and different challenges to a random oracle, and this is the key to prove the straight-line extractability.

The general-purpose transforms described above only use random oracles to achieve straight-line extraction. Other recent transforms, designed for specific Sigma protocols, rely on advanced primitives — such as homomorphic commitment schemes [50] or additively homomorphic encryption [23] — making security of the derived NIZKPs rely on additional computational assumptions. An alternative approach for digital signatures, targeting reduction tightness, is based on the lossy keys technique, used for instance in Lossy CSI-FiSh [38]. While effective for obtaining a tight reduction, the approach in [38] doubles the public key size, the signing cost, and the verification time[4].

As for other works whose focus is on turning a specific class of interactive protocols into straight-line extractable NIZKPs, such as [34] for commit-and-open Sigma protocols, [24,54] for the Schnorr Sigma protocol, and [66] for multi-round interactive protocols, in this work we target a class of Sigma protocols that we call *seed-friendly*. In particular, we focus on building setup-free transforms in the ROM, applicable across diverse seed-friendly Sigma protocols. We call them straight-line extractable transforms throughout.

Seed-Friendly Sigma Protocols and the Fixed-Weight Framework. In this work, we consider the class of Sigma protocols with small challenge spaces that are first-message retrievable and for which responses to a special challenge $\tilde{\mathsf{ch}}$ can be compressed to a short seed. We name the Sigma protocols in this class *seed-friendly Sigma protocols*. For seed-friendly Sigma protocols, the growth in proof size determined by parallel repetitions can be reduced by forcing many compact transcripts for $\tilde{\mathsf{ch}}$ and only a few heavier transcripts for challenges $\mathsf{ch} \neq \tilde{\mathsf{ch}}$. Proofs can be further compressed if the prover generates the seeds using seed trees, reducing the number of seeds to reveal to derive all the compact responses.

[4] The signature size itself remains unchanged despite the increased public-key size, creating an unfavorable trade-off compared to other transformations in terms of compactness.

We gather all the techniques used to generate proofs with a fixed number of transcripts for a challenge different from $\tilde{\mathsf{ch}}$ in what we call the *fixed-weight framework*. Concretely, given a proof weight ρ, the designer determines the minimum value L so that $\binom{L}{\rho} > 2^{\lambda}$ and repeats the underlying Sigma protocol L times in parallel, restricting to challenges of weight ρ[5]. The responses for the ρ repetitions with $\mathsf{ch} \neq \tilde{\mathsf{ch}}$ are usually mathematically-structured objects substantially bigger than seeds, whereas the other $L - \rho$ responses, encodable as seeds, are revealed using seed trees. A similar approach was recently investigated for multi-round identification schemes [9,8] and used in signature schemes like CROSS [6].

Group-Action-Based Sigma Protocols. A flourishing line of research involves post-quantum cryptographic group actions, which have proven to be particularly flexible for the construction of advanced cryptosystems such as linkable commitments [36,44,49], updatable encryption [55,59], ring signatures [13], group signatures [12], threshold signatures [7], multi-signatures [37], and blind signatures [51,46].

Given a cryptographic group action $\star : G \times X \to X$, where G is a finite group and X a finite set, it is possible to construct a Sigma protocol $\Sigma_\star$ to prove knowledge of a witness $g \in G$ for a public statement $(x_0, x_1) \in X \times X$ such that $x_1 = g \star x_0$. $\Sigma_\star$ has a binary challenge space, i.e. $\{0, 1\}$, and responses to challenge 0 can be compressed to seeds as they do not involve the witness g; *it is therefore a seed-friendly Sigma protocol*. To obtain a larger challenge space, $\Sigma_\star$ is repeated in parallel, obtaining a Sigma protocol whose challenges are elements of $\{0, 1\}^L$, with L being the number of repetitions. As parallel repetitions significantly impact compactness, the optimizations from the fixed-weight framework described above, together with seed trees for randomness generation and disclosure [13] and multiple public keys [31] are routinely employed.

$\Sigma_\star$ has been instantiated across multiple post-quantum settings: code-based (LESS [15], MEDS [25]), isogeny-based (CSI-FiSh [14], CSI-SharK [4], PEGASIS [30]), lattice-based [35], and based on trilinear forms (ALTEQ [71]). All these protocols are seed-friendly, making our transforms directly applicable. The increasing number of advanced primitives built from cryptographic group actions suggests that the need for efficient straight-line extractable NIZKPs in this setting will grow; this need is already tangible in [7], whose threshold signature construction requires straight-line extractable NIZKPs but currently relies on the Fischlin or Unruh transform at significant efficiency cost.

Our Contribution In light of *(i)* the effectiveness of the fixed-weight technique for Fiat-Shamir-based signatures from seed-friendly Sigma protocols and *(ii)* applications requiring composable or concurrent security guarantees where straight-line extractability is key, the following research question naturally arises:

[5] In this work, we identify the challenge space $\mathsf{ChSpace}$ of a seed-friendly Sigma protocol with $\{0, 1, \ldots, |\mathsf{ChSpace}| - 1\}$, and $\tilde{\mathsf{ch}}$ with 0. The weight of a challenge in $\mathsf{ChSpace}^{\alpha}$ is the number of components $\mathsf{ch}_i \neq \tilde{\mathsf{ch}}$.

(Q1): Can general-purpose straight-line extractable transforms be effectively integrated within the fixed-weight framework?

Given the overhead introduced by both Pass and Unruh transforms [62,72], they appear to be unsuitable to target compact proofs, therefore constraining Q1 to the Fischlin transform [40].

Fixed-Weight Framework and Fischlin Transform. As a first contribution, we negatively answer Q1. A naive approach would apply the Fischlin transform to the α-fold parallel repetitions of the base Sigma protocol Σ, with α sufficiently large to guarantee completeness, and testing challenges in increasing order of weight to prioritize those responses compressible to seeds. The fundamental limitation of this approach is that it searches independently for valid transcripts in each of the r repetitions of Σ^α. In this case, the expected overall weight of the challenge in a proof is r times the expected weight of the individual repetitions, far exceeding practical targets. We further investigate a variant where α is larger, and proofs with challenges exceeding a target overall weight are rejected. However, we find that achieving low weights requires high computational costs. Even worse, parameter selection lacks closed-form solutions and requires expensive simulations for each candidate set.

Given the negative answer to (Q1), due to the difficulties in applying fixed-weight techniques to the Fischlin transform, we pursue a complementary approach and raise the following research question:

(Q2): Is it possible to design new general-purpose straight-line extractable transforms, leveraging on the seed-friendliness of Sigma protocols, that (i) are inherently in the fixed-weight framework, (ii) achieve computational efficiency, and (iii) enable direct parameter selection?

We answer this second question affirmatively by presenting a new transform. Our main result is introduced in a modular way, by defining first a general straight-line extractable transformation (GAO), which is optimized via the adoption of a collision predicate (Coll-GAO) and a technique (Stretch-and-Compress, or SC) that enhances efficiency and compactness. Combining Coll-GAO with the SC technique, we obtain our final and most efficient transform, named SC-Coll-GAO.

To concretely assess the effectiveness of SC-Coll-GAO, we apply it to the seed-friendly Sigma protocol underlying the LESS digital signature [15], which has recently advanced to the second round of the NIST standardization effort for digital signatures [61]. Among group-action-based signature schemes, LESS offers the best computational performance and achieves near-optimal response sizes (approximately 2λ bits for non-compressed responses), close to the theoretical lower bound. Therefore, efficiency results obtained for LESS provide a lower bound on the improvements achievable for other group-action frameworks with less compact representations. Interestingly, we show that the proof size and computational complexity of NIZKPs obtained by applying SC-Coll-GAO achieve overheads of only 10–20% and 50–60%, respectively, compared to the NIZKPs obtained by applying the Fiat-Shamir transform with the fixed-weight techniques.

These results are particularly relevant as Fiat-Shamir parameterizations ignore the security loss incurred due to rewinding, whereas straight-line extractable transforms provide tight security proofs. More broadly, our transforms provide a setup-free alternative in the ROM for obtaining signatures with tight reductions from seed-friendly Sigma protocols, without the key-generation modifications required by lossy techniques [38].

1.1 Our Techniques

The key challenge tackled by our work is integrating the fixed-weight framework with straight-line extractability. As discussed above, the Fischlin transform searches independently in each of its r repetitions of $\Sigma_\star^\alpha$, causing the overall proof weight to grow proportionally with r. Increasing α, to reduce the per-repetition weight, wastes most repetitions of $\Sigma_\star$, making the computation impractical: the prover typically finds a challenge satisfying the hash predicate at low weight, leaving most of the enlarged challenge space unused. To overcome this, we seek to realize a paradigm in which all first messages of the underlying Sigma protocol form a unified pool, limiting in this way the waste of repetitions, and such that the protocol guarantees that exactly ρ of them contribute to soundness, allowing for fixed-weight challenges.

GAO Transform. Our first contribution is the GAO (Group Action Oriented) transform, a straight-line extractable compiler for seed-friendly Sigma protocols that achieves *fixed proof weight* by construction. The prover computes L first messages $f_1, \ldots, f_L$ of the underlying Sigma protocol Σ and searches across all of them for exactly ρ *target components*, i.e. repetitions for which they can build a transcript with challenge $\mathsf{ch}_i \neq \tilde{\mathsf{ch}}$ satisfying an inversion predicate (which consists in finding a preimage of a b-bit hash digest). The remaining $L - \rho$ repetitions use the special challenge $\tilde{\mathsf{ch}}$, whose responses are compressible to seeds. Unlike Fischlin, where each repetition must independently satisfy the predicate, GAO treats all L first messages as a shared resource: exactly ρ out of L must succeed, each contributing a factor 2^{-b} to soundness for a total of $2^{-b\rho}$. This change of paradigm, while requiring care in the security analysis, avoids requiring parallel repetitions with a sufficiently large challenge space[6], working instead directly with the base Sigma protocol $\Sigma_\star$.

This design enables direct parameter selection: for a target security and weight ρ, the minimum L ensuring a suitable completeness error (typically 2^{-40}) is determined via a closed-form expression, analogously to Fiat-Shamir. Like Fischlin, GAO is provably secure in the ROM without additional setup; the extractor observes the prover's random oracle queries, identifies transcripts sharing the same first message under distinct challenges, and extracts the witness via special soundness. A limitation of GAO is that computing first messages, which are the dominant cost for seed-friendly Sigma protocols, must be done L times. This motivates the introduction of the Stretch-and-Compress technique.

[6] In the group-action setting, Fischlin requires $\Sigma_\star^\alpha$ with $\alpha \approx 2^{b+5}$ [24] to guarantee weight at most 1 for each repetition.

The Stretch-and-Compress Technique (SC). Our second contribution is an optimization technique that reduces the computational cost of GAO by exploiting the asymmetry between the expensive generation of first messages and the cheap evaluation of random oracle queries. The idea of the technique is to operate the parameter selection for the transformation in two phases. In the *stretch phase*, we select parameters yielding a completeness error much larger than the target (e.g., 2^{-1} instead of 2^{-40}), which permits a smaller L and thus fewer first-message computations. In the *compress phase*, the prover reuses the same L first messages and sequentially attempts proof generation with up to k independent random oracles $\mathsf{H}_1, \ldots, \mathsf{H}_k$. Since oracle evaluations are independent, choosing $k = 40$ compresses the overall completeness error from 2^{-1} to $(2^{-1})^{40} = 2^{-40}$. Compensating for the adversary's ability to exploit any of the k oracles requires a slight increase in the predicate hardness, but this overhead is marginal. Beyond its direct efficiency benefit, the SC technique enables transforms whose completeness error resists closed-form analysis, as we discuss next.

Using a Collision Predicate. Our final and most efficient transform, SC-Coll-GAO, applies the SC technique to an intermediate transform called Coll-GAO, which replaces the inversion predicate of GAO with a collision-based predicate, inspired by Kondi and shelat [54]. Instead of ρ individual hash preimages, the prover searches for $\rho/2$ ordered pairs of target components whose transcripts produce colliding hash values, reducing query complexity since each first message can contribute to multiple collisions simultaneously. For security, colliding pairs must be *non-crossing*: given a pair (i_1, i_2) with $i_1 < i_2$, no other pair (j_1, j_2) satisfies $i_1 \leq j_1 \leq i_2$. The collision predicate is more effective in GAO than in the Fischlin setting of [54]: while Fischlin restricts the collision search to two consecutive first messages, GAO searches over a growing number of pools as the prover scans through first messages, reducing the probability of wasted same-pool collisions. Since the completeness error under the collision predicate lacks a tractable closed-form expression, we target a stretched completeness error that can be estimated empirically, then compress it via the SC technique. The intermediate Coll-GAO transform (without SC) is presented in Section 6 as a stepping stone for the security proof.

Experimental Evaluation. We evaluate SC-Coll-GAO on the LESS signature scheme [15], a code-based candidate in the second round of the NIST post-quantum signature standardization process. LESS employs the fixed-weight technique with Fiat-Shamir, making it an ideal benchmark for the cost of straight-line extractability. Targeting the LESS specifications [5] for NIST security level I, we provide optimal parameterizations for SC-Coll-GAO minimizing computational complexity for each challenge-space size and target weight. Compared to Fiat-Shamir LESS, SC-Coll-GAO incurs a proof-size overhead of only 10–20% and a computational overhead of 50–60%. While one might expect a Fischlin-style transform to be at least an order of magnitude more expensive, these results show that the overhead is remarkably contained, making SC-Coll-GAO a practical

choice. All results can be reproduced using the tool available at https://github.com/edoars/gao_transform.

Limitations and Future Work Our security analysis is in the classical random oracle model. The GAO extraction mechanism, like that of Fischlin, relies on observing random oracle queries, which is known to be problematic in the quantum random oracle model (QROM) [3]. Recent techniques [50,34] reduce QROM overhead for specific algebraic structures; achieving QROM security for our transforms through these techniques or more generic approaches [72] is valuable future work.

For isogeny-based instantiations where group actions are free, our transforms apply directly under standard assumptions. For other group actions, we currently assume hardness of the Stabilizer Computational Problem. This assumption can potentially be removed if the Sigma protocol is strong special-sound, as defined by Kondi and shelat [54]: in this case, randomizing challenge selection maintains security even without unique responses. A randomized GAO variant could extend to protocol composition, notably OR-proofs as in Calamari & Falafl [13].

We also see our work as a starting point for extending these techniques to broader classes of interactive protocols, including k-special-sound Sigma protocols (e.g., Stern-like protocols [70]) and multi-round protocols exploiting fixed-weight techniques (e.g., CROSS [6]).

Outline The paper is organized as follows. In Section 2 we recall some preliminaries and present the (formal) notion of *seed-friendly Sigma protocol* and the (informal) notion of *fixed-weight framework*. We also describe our performance-evaluation methodology for the straight-line extractable transforms that we present. In Section 3 we introduce the fixed-weight Fischlin transform, a variant of the Fischlin transform that aims at minimizing the weight of challenges in the proofs by enlarging the challenge space; this transform is then analyzed when applied to the Sigma protocol for cryptographic group actions $\Sigma_\star$. In Section 4 we introduce the GAO transform, our first straight-line extractable transform tailored to seed-friendly Sigma protocols. In Section 5 we present the Stretch-and-Compress technique that allows to improve the GAO NIZKPs making proofs more compact, and their computation more efficient. We call this optimized transform SC-GAO. In Section 6 we introduce another optimization (orthogonal with respect to the Stretch-and-Compress technique) for the GAO transform, which is based on the use of a collision predicate, yielding the Coll-GAO transform. This optimization can be later combined with the Stretch-and-Compress technique to obtain our final transform referred to as SC-Coll-GAO transform. Concrete comparisons with the Fiat-Shamir LESS signature are provided for each of the newly-introduced transforms, within the sections where they are presented.

2 Preliminaries

Throughout the paper we adopt a standard notation, detailed in Appendix A. The only non-standard notation we use is the following: we write $a(\lambda) \approx b(\lambda)$ for two functions $a(\lambda)$ and $b(\lambda)$ that are negligibly close.

Straight-Line Extractable NIZKPs In this work we consider the definition of straight-line extractable NIZKP in [40]. Informally, a NIZKP for a binary relation $\mathcal{R}$, with straight-line (or online) extractability, completeness error $\epsilon_c^{\mathsf{Prove}}$ and soundness error $\epsilon_s^{\mathsf{Ext}}$, consists of a pair of oracle-calling PPT algorithms $(\mathsf{Prove}, \mathsf{Verify})$ that satisfy the following properties:

1. *Completeness*: Let H be a random oracle. Prove, on input a statement-witness pair $(x, w) \in \mathcal{R}$, generates an invalid proof $\pi = \perp$ for x with probability $\epsilon_c^{\mathsf{Prove}}$, i.e. $\epsilon_c^{\mathsf{Prove}} = \Pr[\perp \leftarrow\!\$\ \mathsf{Prove}(x, w) \mid (x, w) \in \mathcal{R}]$.
2. *Zero-Knowledge*: For any PPT distinguisher $\mathcal{D}$, there exists a simulator Sim such that, for any statement x chosen by $\mathcal{D}$, Sim can produce a valid proof for x without knowing the associated witness. The generation of this proof must be indistinguishable from the generation of an honestly generated proof.
3. *Straight-Line Extractor*: There exists a PPT algorithm Ext — called straight-line extractor — that, for any adversary $\mathcal{A}$, if $\mathcal{A}$ can generate a valid proof π for a statement x, and Ext is given x, π and all random-oracle queries made by $\mathcal{A}$, Ext will be able to extract a witness w for x with probability $\geq (1 - \epsilon_s^{\mathsf{Ext}})$.

For a formal definition of straight-line extractable NIZKPs we refer to Section A.1 (Definition 8).

Seed-Friendly Sigma Protocols We consider the standard definition of Sigma protocol [29]. We say that a Sigma protocol Σ is *effective* if it satisfies the *first-message entropy* and *unique responses* properties. Σ is *first message retrievable* if its transcripts can be encoded using only the challenge and the response (see Section A.2 for formal definitions).

In this section we define a special class of Sigma protocols that we call *seed-friendly*.

Definition 1 (Seed-Friendly Sigma Protocol)**.** *We say that a Sigma protocol* $\Sigma = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ *for a binary relation* $\mathcal{R}$*, with challenge space* $\mathsf{ChSpace}$*, is seed-friendly if the following four properties*[7] *are satisfied:*

1. *it is effective;*
2. *it is first-message retrievable;*
3. *its challenge-space size is at most* $\mathsf{poly}(\lambda)$*;*

[7] Property 1 is to ensure that it is possible to compile Σ into a NIZKP using Fiat-Shamir, Fischlin and our compilers. While Properties 2 and 3 are not required to exploit Property 4, their concomitant validity allows one to take advantage of Property 4 more substantially.

4. *for any* $(x, w) \in \mathcal{R}$ *and first message* $f \leftarrow \mathsf{P}_1(x, w; r)$ *generated by* P_1 *using randomness* r*, there exists a special challenge* $\tilde{\mathsf{ch}} \in \mathit{ChSpace}$ *such that the associated response* z *can be computed as a public polynomial-time function* $z \leftarrow Z(x, r)$ *of the statement* x *and randomness* r*.*

Remark 1. Throughout this work, we will identify the challenge space $\mathsf{ChSpace}$ of a seed-friendly Sigma protocol with the set $\{0, 1, \ldots, |\mathsf{ChSpace}| - 1\}$, and assume that the special challenge $\tilde{\mathsf{ch}}$ is 0.

Property 4 implies that, whenever the prover receives from the verifier the challenge $\tilde{\mathsf{ch}}$, the prover can respond to the verifier by sending only the randomness r and letting the verifier retrieve the response by computing the public function $Z(x, r)$. This also implies that, if the randomness r is generated by computing a pseudorandom generator $\mathsf{PRG}(\mathsf{seed})$ on a random λ-bit seed seed, the prover can send the verifier the λ-bit seed, and the verifier can verify the transcript executing $\mathsf{V}_2\big(f, x, \tilde{\mathsf{ch}}, Z(x, \mathsf{PRG}(\mathsf{seed}))\big)$. Furthermore, we observe that for a seed-friendly Sigma protocol, a transcript for the special challenge $\tilde{\mathsf{ch}}$ can be encoded in $\log(\mathsf{poly}(\lambda))$ bits to represent $\tilde{\mathsf{ch}}$ and a λ-bit seed for the response.

Seed-friendly Sigma protocols arise naturally across diverse settings[8]. A classic example is the Sigma protocol for graph isomorphism [45].

In the post-quantum setting, the Sigma protocol $\Sigma_\star$ from cryptographic group actions (Section 2) is seed-friendly — under the Stabilizer Computation Problem assumption for unique responses — and has been instantiated across multiple families: code-based (LESS [15], MEDS [25]), isogeny-based (CSI-FiSh [14], CSI-SharK [4], PEGASIS [30]), lattice-based [35], and based on trilinear forms (ALTEQ [71]). The seed-friendly structure extends to more advanced protocols built on top of $\Sigma_\star$, such as the ring and group signature schemes Calamari and Falafl [13,12].

The Fixed-Weight Framework Sigma protocols can be used to construct NIZKPs by compiling them using generic transforms, for instance, the Fiat-Shamir [39,64] and the Fischlin [40] transforms. Generally, these transforms require the compiled Sigma protocol to have a *sufficiently large* challenge space, with the size depending on the transform one wants to employ[9]. When a Sigma protocol has a small challenge space, parallel repetition (see Section A.2, Definition 12) is used to enlarge it, thereby reducing the soundness error. However, this approach can result in large proofs, since a proof for the resulting protocol will contain multiple transcripts of the repeated Sigma protocol.

For a seed-friendly Sigma protocol Σ, it is possible to mitigate the increase in the size of the transcript due to the parallel repetition of Σ by employing the

[8] Seed-friendly definition is satisfied even by the Schnorr Sigma protocol with a restricted challenge space (see Section A.3 for details).

[9] For the Fiat-Shamir transform, the size of the challenge space must be exponential in the security parameter. Instead, for the Fischlin transform, the size of the challenge space must be at least polynomial and has further constraints.

techniques we gather in the *fixed weight framework*. The fixed-weight framework encompasses techniques to make NIZKPs derived from seed-friendly Sigma protocols produce proofs which contain a majority of *small* transcripts corresponding to the special challenge $\tilde{\mathsf{ch}}$, while limiting the number of transcripts with other challenges that require *full-size* responses. This approach is remarkably effective thanks to the use of *seed trees* to encode small responses.

Seed Trees. To reduce the response size, this optimization targets the way seeds are built and revealed. In particular, it consists in generating all the seeds used to compute the first messages of the repeated Sigma protocol Σ from a single λ-bit seed seed. This seed is given in input to a pseudo-random generator $\mathsf{PRG} : \{0,1\}^{\lambda} \to \{0,1\}^{2\lambda}$ and the first λ bits of the output will be the left child of seed and the remaining λ bits the right child. The procedure is repeated until all the needed leaves, i.e. the seeds that will be used, have been generated. This procedure determines a binary tree. Note that to reveal the seeds corresponding to the challenges $\mathsf{ch}_i = \tilde{\mathsf{ch}}$, it will be generally sufficient to reveal some of their ancestors. More concretely, suppose that Σ is repeated in parallel L times and that L can be written as $L = L_1 + L_2 + \cdots + L_u$, where the L_i's are powers of two with $L_1 > L_2 > \cdots > L_u$. Let $U \subseteq \{1, \ldots, L\}, |U| = \rho$ be the indices of the leaves of the tree that must remain hidden. Then, according to [16, Proposition 2], the number of seeds that must be revealed to disclose all the $L \setminus |U|$ leaves corresponding to the challenges $\mathsf{ch}_i = \tilde{\mathsf{ch}}$ is not greater than $N_{\mathsf{seed}} = \rho \log(L/\rho) + u - 1$.

The specific techniques within this framework depend on the compiler used to transform the Sigma protocol into a NIZKP. As a concrete example, we describe in Section A.4 the fixed-weight technique used within the Fiat-Shamir transform, which is the approach employed in many practical schemes such as the LESS digital signature [15].

Cryptographic Group Actions A trend that has gained traction in recent years is the use of cryptographic group actions [17] to design cryptographic schemes, as pseudo-random functions [2], commitment schemes [36,44,49], digital signature schemes [15,25,21,31,35,71] and advanced signature schemes [37,13,12,7,46]. In Section A.5 we recall the definition of cryptographic group actions and the main assumptions, namely the Group Action Inversion Problem (GAIP) and the Stabilizer Computation Problem (SCP). When the GAIP assumption holds, it is possible to build cryptographic primitives on top of the well known Sigma protocol from cryptographic group actions $\Sigma_\star$, which, when the SCP assumption holds, is also seed-friendly.

The Sigma protocol $\Sigma_\star$ Given a cryptographic group action $(G, X, \star)$, it is possible to define a Sigma protocol $\Sigma_\star = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ for the binary relation $\mathcal{R}_\star = \{(g, (x_0, x_1)) \in G \times X^2 | x_1 = g \star x_0\}$. On input $(g, (x_0, x_1))$, P_1 returns $f \leftarrow \tilde{g} \star x_0$, where $\tilde{g}$ is sampled uniformly at random from G. V_1 outputs a uniformly-random bit $\mathsf{ch} \leftarrow\!\!\$\ \{0,1\}$. P_2, given g,(x_0, x_1), f and ch outputs $z \leftarrow \tilde{g} g^{-\mathsf{ch}}$ as response. Finally, V_2 returns accept if $z \star x_{\mathsf{ch}} = f$.

In the following, we work under the hypothesis that the SCP assumption is hard, meaning that $\Sigma_\star$ is seed-friendly.

Remark 2. The SCP assumption is needed to guarantee the *unique responses* property of $\Sigma_\star$, which is required for effectiveness (Definition 1). For isogeny-based group actions, where the action of G on X is free (i.e. the stabilizer of every element is trivial), unique responses hold unconditionally and the SCP assumption is not needed; this is the case for CSI-FiSh [14], CSI-SharK [4] and PEGASIS [30]. For other instantiations such as LESS [15], MEDS [25] and ALTEQ [71], the SCP assumption is standard and widely adopted. We note that this assumption can potentially be removed by adopting the strong special-soundness framework of Kondi and shelat [54], where security is maintained without unique responses provided that the prover randomizes the challenge selection (see [54, Section 6.3]). We leave the investigation of a randomized GAO variant to future work; such a variant could also extend to protocol composition, notably OR-proofs as in Calamari & Falafl [13,12].

To enlarge its challenge space, $\Sigma_\star$ is generally repeated, with the impact on the proof size mitigated by fixed-weight optimizations and the use of seed trees. A further optimization, tailored for $\Sigma_\star$, decreases the proof size at the cost of bigger public keys [31] (see Section A.6).

The Fischlin Transform We recap the generic transform proposed by Fischlin [40]. Recently, Chen and Lindell revisited this transform [24], streamlining its description, reducing the number of parameters used to define it and simplifying the relations among the parameters that must be satisfied in order to obtain secure NIZKPs.

In the original formulation [40], given an underlying Sigma protocol Σ, on input $(x, w) \in \mathcal{R}$, the proof-generation algorithm $\mathsf{Prove}_{\mathtt{Fi}}$ produces r first messages $\overline{f} = (f_1, \ldots, f_r)$ and searches for $\mathsf{ch}_i \in \mathsf{ChSpace}, i \in [r]$ such that the inversion predicate is satisfied: $\mathsf{H}(x, \overline{f}, i, \mathsf{ch}_i, z_i) \in \{0\}^b \times \{0,1\}^{\lambda - b}$ (i.e. the first b bits are zero) for some integer parameter b, where $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^\lambda$ is a random oracle and z_i is the response corresponding to f_i and ch_i. This predicate is satisfied with probability 2^{-b} for each transcript, requiring an expected 2^b hash computations per successful transcript. The proof is given by the set of transcripts $(\{f_i\}_{i\in[r]}, \{\mathsf{ch}_i\}_{i\in[r]}, \{z_i\}_{i\in[r]})$, each satisfying the inversion predicate.

Analysis of the Fischlin Transform [24]. The size of the challenge space $\mathsf{ChSpace}$ of the underlying Sigma protocol Σ affects the completeness error $\epsilon_c^{\mathsf{Prove}_{\mathtt{Fi}}}$, i.e. the probability that $\mathsf{Prove}_{\mathtt{Fi}}$ does not find a suitable challenge for each first message (see Definition 8). In particular, under the assumption $|\mathsf{ChSpace}| = 2^\alpha$ for some $\alpha \in \mathbb{N}^*$, the completeness error is $\log \epsilon_c^{\mathsf{Prove}_{\mathtt{Fi}}} \leq -2^{\alpha - b} \log(e) + \log(r)$ (see [24, Section 3]). Exploiting this relation, the completeness error can be tuned with respect to the size of the challenge space, for fixed r and b. In particular, to guarantee a completeness error $\epsilon_c^{\mathsf{Prove}_{\mathtt{Fi}}} \leq 2^{-40}$, the number of possible challenges

must be

$$|\mathsf{ChSpace}| \geq 2^{b+5} \text{ (if } r < 64). \tag{1}$$

To achieve a soundness error $\epsilon_s^{\mathsf{Ext}_{\mathtt{Fi}}} < 2^{-\lambda}$ it is required that

$$b \cdot r \geq \lambda, \tag{2}$$

where $\mathsf{Ext}_{\mathtt{Fi}}$ is the straight-line extractor used to prove the straight-line extractability of the NIZKP obtained using the Fischlin transform [40,24]. Moreover, we observe that the expected random-oracle-query complexity of $\mathsf{Prove}_{\mathtt{Fi}}$ is $\mathbb{E}[Q] = 2^b \cdot r$.

Our Inversion Predicate. In this work we introduce a slight generalization that enables finer control over the expected query complexity by taking b rational.

Definition 2. *Let λ be a security parameter, let $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^\lambda$ be a hash function, and let $b \in \mathbb{Q}, 0 < b < \lambda$. Given $x \in \{0,1\}^*$ we say that x satisfies the inversion predicate if and only if $\mathsf{H}(x) < \lfloor 2^{\lambda-b} \rfloor$, where $\mathsf{H}(x)$ is interpreted as an integer in $[0, 2^\lambda - 1]$.*

The probability that the inversion predicate is satisfied is less than and negligibly close to 2^{-b} (as we discuss in detail in Section D.1). A formal definition of this slight modification of the Fischlin transform is available in Section A.7, Definition 19.

Performance Evaluation Methodology The fixed-weight framework for seed-friendly Sigma protocols involves a trade-off between proof size and computational cost. By allowing responses to a certain number of challenges to be encoded as compact seeds, the proof size can be significantly reduced. However, this reduction typically comes at the expense of an increased computational overhead, as more first messages must be generated to maintain the same security level. Our evaluation methodology focuses on quantifying both sides of this trade-off across different generic transforms that turn Sigma protocols into NIZKPs.

Metrics. We measure computational complexity in terms of fundamental operations: CC_f (generating a first message), CC_z (computing a response for $\mathsf{ch} \neq \tilde{\mathsf{ch}}$), and CC_H (hash function evaluation). Proof size accounts for ρ full responses and $L - \rho$ seed-encoded responses, assuming seed-tree optimization. Our transforms aim to minimize computational cost for a given target weight ρ.

Parameter Selection. A key advantage of the new transforms we propose is straightforward parameter selection: given target security levels and weight ρ, optimal parameters can be efficiently determined using the methodology in Section 5.3. In contrast, applying the fixed-weight framework to Fischlin encounters fundamental difficulties (Section 3.1), requiring expensive empirical simulations.

Table 1: LESS parameters for security parameter $\lambda = 128$, and baseline performance (in Megacycles) for a (L, ρ)-fixed-weight repetition and ℓ public keys.

Parameter Set	L	ρ	ℓ	**sig (B)**	CC
LESS-252-192	192	36	1	2609	115.4M
LESS-252-68	68	42	3	1825	41.0M
LESS-252-45	45	34	7	1329	27.2M

Running Example. Throughout this paper, LESS [15] serves as our reference point for comparisons. LESS is a code-based signature scheme recently advanced to Round 2 of the NIST post-quantum signature standardization process [61]. Among group-action-based signature schemes, LESS offers the best computational performance and achieves near-optimal response sizes (approximately 2λ bits for non-compressed responses), close to the theoretical lower bound. Therefore, efficiency results obtained for LESS provide a lower bound on the improvements achievable for other group-action frameworks with less compact representations. For details on LESS parameters, performance baselines, and implementation specifics we refer to Section A.8. Official parameterizations involving the (L, ρ)-fixed-weight optimization with seed trees and ℓ public-keys are shown in Table 1.

All performance values reported throughout this paper are obtained from LESS reference code[10] on a 8th Gen Intel(R) Core(TM) i7-8665U and measured in Megacycles (M). Following the above methodology, the Fiat-Shamir transform applied to a (L, ρ)-fixed-weight repetition has computational cost approximately equal to

$$L \cdot \mathsf{CC}_f + \mathsf{CC}_{\mathsf{H}} + \rho \cdot \mathsf{CC}_z, \quad \text{with } \mathsf{CC}_f = 0.6\text{M},\ \mathsf{CC}_H = 0.001\text{M},\ \mathsf{CC}_z = 0.005\text{M}.$$

3 Fixed-Weight Framework and Fischlin Transform

In this section we analyze the application of the Fischlin transform to $\Sigma_\star$. Since $\Sigma_\star$ has a binary challenge space, parallel repetition is necessary to achieve completeness error $\epsilon_c^{\mathsf{Prove}_{\mathsf{Fi}}} \leq 2^{-40}$ according to Equation (1). For a seed-friendly Sigma protocol, encoding responses for challenge $\tilde{\mathsf{ch}} = 0$ as seeds reduces proof size. A natural optimization is to search challenges in increasing order of (Hamming) weight, prioritizing lighter challenges that yield more compact encoding. In Section C.1, we derive an explicit formula for the expected overall weight $\mathbb{E}[W(r, b, \alpha)]$ of challenges in a Fischlin proof when using this weight-ordered search strategy. For $\lambda = 128$ and minimal α guaranteeing $\epsilon_c^{\mathsf{Prove}_{\mathsf{Fi}}} \leq 2^{-40}$, the expected weights remain impractically high (≥ 45) even with optimized parameters, as detailed in Table 5.

Considering the significant proof weights, it is natural to ask whether, for a pair (b, r) satisfying $b \cdot r = \lambda$, one can design a technique to decrease the total challenge weight in proofs obtained from compiling $\Sigma_\star$ via the Fischlin transform.

[10] https://github.com/less-sig/LESS

Fixed-weight technique applied to the Fischlin transform. A natural approach is to first choose a threshold weight ρ as the upper bound to the overall weight of the challenge ch in any proof π and then find the minimum α that guarantees, with probability no greater than 2^{-40}, that the weight of ch does not exceed ρ, otherwise the proof is discarded. This requirement can be trivially encoded in the definition of the proof-generation algorithm of the NIZKP obtained by compiling $\Sigma_\star$ using the Fischlin transform. We call this resulting variant of the Fischlin transform the *fixed-weight Fischlin transform* with parameters (b, r, α, ρ).

Remark 3. When the fixed-weight technique is used within the Fiat-Shamir transform on a Sigma protocol Σ, designers choose the parameter sets by exhaustive search [13,6,43]. In particular, for any fixed-weight ρ, the minimum number L of parallel repetitions of Σ that provides a suitable security level is determined. Then, among the listed pairs (ρ, L), the one achieving the *best* tradeoff between compactness and computational efficiency is picked. We wish this strategy could be extended to the fixed-weight Fishlin transform. As a first observation, the minimum number of first messages L depends not only on the fixed-weight ρ but also on the parameters b and r, adding a degree of freedom. More fundamentally, we will now show that determining L is especially computationally heavy, signaling a clear limitation of the fixed-weight Fishlin transform.

3.1 Hardness of Computing the Completeness Error of the Fixed-Weight Fischlin Transform

In the following we describe a procedure to determine the minimal parameter α that ensures that the weight of ch does not exceed ρ with probability no greater than the target completeness error. In particular, we employ a sampling-based approach combined with statistical confidence intervals and binary search optimization.

Methodology. Let $W(r, b, \alpha)$ describe the weight of a proof when the Fischlin transform is applied to Σ^α (see Definition 12). Given the random variable $W(r, b, \alpha)$ with fixed parameters r, b, a fixed-weight parameter ρ and a target completeness error $\epsilon_c^{\mathsf{Prove}_{\mathtt{Fi}},\rho}$, the goal is to find the minimal α such that $\Pr[W(r, b, \alpha) > \rho] \leq \epsilon_c^{\mathsf{Prove}_{\mathtt{Fi}},\rho}$. While the exact distribution of $W(r, b, \alpha)$ remains unknown, we can generate samples from it to estimate its distribution. The estimate relies on the basic observation that, when sampling from $W(r, b, \alpha)$, the probability of success (i.e. samples exceeding the weight ρ) follows a Bernoulli distribution $\mathrm{B}(p)$ with unknown parameter $p = \Pr[W(r, b, \alpha) > \rho]$. We employ sequential estimation via negative binomial Monte Carlo [65] to estimate p. For a candidate value of α, we generate samples $W_1, W_2, \ldots$ from $W(r, b, \alpha)$ sequentially until a given number N of successes is reached. The resulting number of trials is denoted by n and the uniformly minimum variance unbiased estimator of p is $\hat{p} = (N - 1)/(n - 1)$. Following the results of [60], we choose $N = 8$ to ensure that the normalized mean absolute error of the estimator $\hat{p}$ does not exceed

Table 2: Minimum values of the number of first messages $L = \alpha \cdot r$ for different Fischlin parametrizations (b, r) with completeness error $\epsilon_c^{\mathsf{Prove}_{\mathsf{Fi}},\rho} \leq 2^{-16}$ and a given proof weight ρ. We mark with a dash values exceeding $L = 25000$.

r	b	$\rho = 48$	$\rho = 36$	$\rho = 32$	$\rho = 28$	$\rho = 24$	$\rho = 22$	$\rho = 20$
43	2.98	1118	–	–	–	–	–	–
32	4	768	2080	6304	–	–	–	–
26	4.92	572	1794	2626	4654	–	–	–
22	5.82	418	1782	2552	3938	7326	16786	–
19	6.74	361	1520	2755	4351	6935	9766	16074
16	8	304	672	1120	5344	8992	11824	15936
14	9.14	280	658	882	1582	11564	15806	20762

30%[11]. Notice that the value of N is independent of p. In particular, choosing $N = 8$, we obtain that $\varepsilon(p) := \mathbb{E}[|\hat{p} - p|]/p < 0.3$, for all $p \in (0, 1)$. To find the minimal α that satisfies the constraint $(1 + \varepsilon(p))\hat{p} \leq \epsilon_c^{\mathsf{Prove}_{\mathsf{Fi}},\rho}$, we exploit the monotonic decreasing relationship between α and p, enabling efficient parameter search through binary search techniques.

In Table 2, we report the minimum values $L = \alpha \cdot r$ of first messages, for different Fischlin parametrizations (b, r) and a given target proof weight ρ. We use a relaxed threshold $\epsilon_c^{\mathsf{Prove}_{\mathsf{Fi}},\rho} = 2^{-16}$ rather than the target 2^{-40} as estimating parameters for 2^{-40} would require at least 2^{40} operations per simulation for each parameter candidate (b, r, ρ). As observed in Remark 3, optimal-parameter choice would require numerous such estimations across the parameter space, making the computational cost prohibitive, due to the multidimensional empirical simulations required. In addition, Table 2 illustrates a further fundamental limitation of applying the fixed-weight framework to Fischlin: even with the relaxed threshold, achieving low weights (e.g., $\rho < 22$) requires $L \geq 10000$, making first-message computation impractically expensive.

Implications for Group Actions. Our running example LESS achieves near optimal group element representation ($\approx 2\lambda$ bits) through canonical forms [26], enabling practical signatures with relatively high weights ($\rho = 36, 42, 34$ in Table 1). However, LESS represents an exceptional case. Schemes like MEDS [25] use 37λ-bit group elements, and similar group actions appear in commitment schemes [36,44] that support NIZKPs of committed elements, which can be turned into straight-line extractable NIZKPs. For such applications, reducing ρ translates directly to significant proof-size reduction, making smaller weights essential for compactness.

[11] When the value p is sufficiently small, e.g. 2^{-16}, this methodology applied with $N = 8$ provides a good approximation. The bigger is p, the bigger must be N to keep the expected distance between $\hat{p}$ and p small.

4 The Group Action Oriented Transform

In this section, we propose a new generic transformation which turns seed friendly Sigma protocols into straight-line extractable NIZKPs that are fixed-weight by design. We name the proposed transformation *GAO transform*, where GAO is an acronym of *Group Action Oriented*, as it was designed with the group action framework in mind. Nevertheless, the GAO transform can be applied to any effective Sigma protocol with small challenge space [12]; in the following, we describe it in this general setting. The polynomial-size challenge space is crucial to guarantee that the prover algorithms in all our transforms runs in polynomial time.

4.1 The GAO Transform

Intuition. The transformation instructs the prover to begin with the generation of L first messages of the Sigma protocol Σ. Then, the prover must construct exactly ρ transcripts for challenges different from $\tilde{\mathsf{ch}}$, the special challenge. These ρ transcripts must satisfy a predicate which is essentially the one used in the Fischlin transform [40]. The indices of the first messages which correspond to the ρ challenges different from $\tilde{\mathsf{ch}}$ will be called *target components*. The remaining $L-\rho$ first messages instead, will be part of transcripts for the special challenge $\tilde{\mathsf{ch}}$. Thanks to the seed-friendliness of the Sigma protocol Σ, these $L-\rho$ transcripts can be efficiently compressed and the resulting proof size will not be affected much by the inclusion of these $L-\rho$ transcripts for the special challenge $\tilde{\mathsf{ch}}$, thanks to the seed tree optimization.

More formally the transform is defined as follows.

Definition 3 (GAO Transform)**.** *Let λ be a security parameter and let $\Sigma = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ be a seed-friendly Sigma protocol for a binary relation $\mathcal{R}$, with challenge space $\mathsf{ChSpace}$. Let $L, \rho \in \mathbb{N}^*, b \in \mathbb{Q}$ with $0 < b < \lambda$, let $\tilde{\mathsf{ch}} = 0$ be the special challenge of Σ, and let $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^\lambda$ be a random oracle. The GAO transform, parametrized by $(L, b, \rho, \tilde{\mathsf{ch}})$, applied on Σ produces a NIZKP $(\mathsf{Prove}_\blacktriangle, \mathsf{Verify}_\blacktriangle)$ as defined in Transformation 1.*

In Transformation 1, we adopted the same inversion predicate we have introduced in Definition 2 for the Fischlin transform. In particular, given $\overline{f} = (f_1, \cdots, f_L)$, $d \leftarrow \mathsf{H}(x, \overline{f})$ and $i \in [L]$, we say that the transcript $(f_i, \mathsf{ch}_i, z_i)$ satisfies the *inversion predicate* if $\mathsf{H}(d, i, \mathsf{ch}_i, z_i) < \lfloor 2^{\lambda - b} \rfloor$.

4.2 Security of the GAO Transform

Theorem 1. *Let λ be a security parameter and let $\Sigma = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ be a seed-friendly Sigma protocol for a binary relation $\mathcal{R}$, with challenge*

[12] The challenge space of a Sigma protocol can always be restricted, so this transform can be applied to any effective Sigma protocol.

Transformation 1: GAO$[L, b, \rho, \tilde{\mathsf{ch}}]$

Let $\Sigma = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ be a seed-friendly Sigma protocol for $\mathcal{R}$ and let $(x, w) \in \mathcal{R}$.

$\mathsf{Prove}_{\blacktriangle,1}(x, w)$:
1: **for** $i \leftarrow 1, \ldots, L$ **do**
2: $\quad f_i \leftarrow\!\$\ \mathsf{P}_1(x, w)$
3: $\overline{f} \leftarrow (f_1, \ldots, f_L)$

$\mathsf{Prove}^{\mathsf{H}}_{\blacktriangle,2}(x, w, \overline{f})$:
1: $\mathcal{T} \leftarrow \emptyset$ ▷ Init target components
2: $d \leftarrow \mathsf{H}(x, \overline{f})$
3: **for** $i \leftarrow 1, \ldots, L$ **do**
4: $\quad$ **for** $\mathsf{ch}_i \in \mathsf{ChSpace} \setminus \{\tilde{\mathsf{ch}}\}$ **do**
5: $\quad\quad z_i \leftarrow\!\$\ \mathsf{P}_2(x, w, f_i, \mathsf{ch}_i)$
6: $\quad\quad$ **if** $\mathsf{H}(d, i, \mathsf{ch}_i, z_i) < \lfloor 2^{\lambda - b} \rfloor$ **then**
7: $\quad\quad\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{i\}$
8: $\quad\quad\quad$ **break**
9: $\quad\quad$ **if** $|\mathcal{T}| = \rho$ **then**
10: $\quad\quad\quad$ **break**
11: **if** $|\mathcal{T}| < \rho$ **then**
12: $\quad$ **return** $\perp$
13: **for** $i \leftarrow [L] \setminus \mathcal{T}$ **do**
14: $\quad \mathsf{ch}_i \leftarrow \tilde{\mathsf{ch}}$
15: $\quad z_i \leftarrow\!\$\ \mathsf{P}_2(x, w, f_i, \mathsf{ch}_i)$
16: **return** $\pi \leftarrow (\{f_i\}_{i\in[L]}, \{\mathsf{ch}_i\}_{i\in[L]}, \{z_i\}_{i\in[L]})$

$\mathsf{Prove}^{\mathsf{H}}_{\blacktriangle}(x, w)$:
1: $\overline{f} \leftarrow\!\$\ \mathsf{Prove}_{\blacktriangle,1}(x, w)$
2: $\pi \leftarrow\!\$\ \mathsf{Prove}^{\mathsf{H}}_{\blacktriangle,2}(x, w, \overline{f})$
3: **return** π

$\mathsf{Verify}^{\mathsf{H}}_{\blacktriangle}(x, \pi = (\{f_i\}_{i\in[L]}, \{\mathsf{ch}_i\}_{i\in[L]}, \{z_i\}_{i\in[L]}))$:
1: $\overline{f} \leftarrow (f_1, \ldots, f_L)$
2: $d \leftarrow \mathsf{H}(x, \overline{f})$
3: **if** $|\{\, i \in [L] \mid \mathsf{ch}_i \neq \tilde{\mathsf{ch}} \,\}| \neq \rho$ **then**
4: $\quad$ **return** reject
5: **for** $i \leftarrow 1, \ldots, L$ **do**
6: $\quad$ **if** $\mathsf{V}_2(x, f_i, \mathsf{ch}_i, z_i) = \text{reject}$ **then**
7: $\quad\quad$ **return** reject
8: $\quad$ **if** $\mathsf{ch}_i \neq \tilde{\mathsf{ch}}$ and $\mathsf{H}(d, i, \mathsf{ch}_i, z_i) \geq \lfloor 2^{\lambda - b} \rfloor$ **then**
9: $\quad\quad$ **return** reject
10: **return** accept

space $\mathsf{ChSpace}$ *and special challenge* $\tilde{\mathsf{ch}} = 0$. *Let* $(\mathsf{Prove}_{\blacktriangle}, \mathsf{Verify}_{\blacktriangle})$ *be the non-interactive protocol obtained by applying the GAO transform (Transformation 1), parametrized by the parameters* $(L, b, \rho, \tilde{\mathsf{ch}})$, *with* L *and* ρ *non-zero natural numbers and* b *a positive rational number less than* λ. *Then* $(\mathsf{Prove}_{\blacktriangle}, \mathsf{Verify}_{\blacktriangle})$ *is an straight-line extractable NIZKP for the relation* $\mathcal{R}$ *(in the random oracle model) with:*

1. *completeness error* $\epsilon_c^{\mathsf{Prove}_{\blacktriangle}} = \sum_{i=0}^{\rho-1} \binom{L}{i} p^i (1-p)^{L-i}$, *with* $p \approx 1 - (1 - 2^{-b})^{\ell}$;[13]
2. *soundness error* $\epsilon_s^{\mathsf{Ext}_{\blacktriangle}} \leq 2^{-b\rho}$;
3. *query complexity* $Q^{\mathsf{Prove}_{\blacktriangle}} \leq L \cdot \ell$;

where $S = \mathsf{ChSpace} \setminus \{\tilde{\mathsf{ch}}\}$ *and* $\ell = |S|$.

A full proof is available in Section B.1.

Proof Sketch. **Completeness:** According to our inversion predicate (Definition 2), each component is a target component with probability $p \approx 1 - (1 - 2^{-b})^{\ell}$. A honest prover fails in generating a proof if fewer than ρ targets components are found, which happens with probability

$$\epsilon_c^{\mathsf{Prove}_{\blacktriangle}} = \sum_{i=0}^{\rho-1} \binom{L}{i} p^i (1-p)^{L-i}. \tag{3}$$

[13] In Appendix D we argue that the real completeness error that we obtain using the inversion predicate is negligibly close to the value reported in this theorem. This is the reason why we use the symbol $\approx$ instead of the equality.

Zero-Knowledge: The simulator programs the random oracle and once it receives from the distinguisher a statement x the simulator samples random digests $\tau_i : S \to \{0,1\}^\lambda$ for each $i \in [L]$ that will be associated to the first message f_i and each challenge in $S = \mathsf{ChSpace} \setminus \{\tilde{\mathsf{ch}}\}$. The simulator identifies the first ρ indices where $\tau_i^{-1}([0, \lfloor 2^{\lambda-b} \rfloor]) \neq \emptyset$ and it runs the HVZK simulator of Σ for these components with respect to the associated challenge ch_i mapping τ_i to $[0, \lfloor 2^{\lambda-b} \rfloor]$ (i.e. satisfying the inversion predicate in Definition 2). The simulator programs $\mathsf{H}(d, i, \mathsf{ch}_i, z_i) = \tau_i(\mathsf{ch}_i)$, but also "implicitly programs" H on inputs (d, i, ch, z) where $i \in \{i \in [L], i \leq \max(\mathcal{T})\}$, $\mathsf{ch} \in S, \mathsf{ch} \neq \mathsf{ch}_i$ and $\mathsf{V}_2(x, f_i, \mathsf{ch}, z) = \mathsf{accept}$ to $\tau_i(\mathsf{ch})$[14]. This is because the distinguisher might know a witness for x and send queries for valid transcripts. In this case the simulator must answer sending consistent digests that are not in contrast with the selected target components in the proof.

$\mathsf{Sim}_\blacktriangle$ produces a proof π which is indistinguishable from a real proof because the outputs of the hash function H are sampled uniformly at random, and the $\mathsf{ch}_i \in S$ in the proof are selected as in a real-protocol execution. Also, the issue in the definition of the simulator highlighted in [67, A.5] (and remarked in [53]) is not a concern for our security proof because no random oracle input is arbitrarily assigned a digest $\leq \lfloor 2^{\lambda-b} \rfloor$. It is easy to see that the random oracle programming fails only with negligible probability thanks to the first-message entropy property of Σ.

Extraction: The extractor $\mathsf{Ext}_\blacktriangle$ we consider is defined as follows. It loops through the list of random oracle queries $Q_H(\mathcal{A})$ made by the adversary $\mathcal{A}$ and the transcripts $\{(f_i, \mathsf{ch}_i, z_i)_{i \in [L]}\}$ included in π, looking for two accepting transcripts $(f, \mathsf{ch}, z), (f, \mathsf{ch}', z')$ with $\mathsf{ch} \neq \mathsf{ch}'$. If the search is successful, $\mathsf{Ext}_\blacktriangle$ calls the witness extractor Ext of the underlying Sigma protocol Σ on the two triples, and outputs the extracted value.

Without loss of generality, we can restrict our analysis to adversaries $\mathcal{A}$ who only send "meaningful random-oracle queries", which we define as queries of the form (d, i, ch, z), with $d = \mathsf{H}(x, \overline{f}), \overline{f} = (f_1, \ldots, f_L), \mathsf{ch} \in S$ and $\mathsf{V}_2(x, f_i, \mathsf{ch}, z) = \mathsf{accept}$. Let us assume that $\mathcal{A}$ outputs a valid proof π without providing the extractor $\mathsf{Ext}_\blacktriangle$ with two accepting transcripts for the same first message and different challenges. Intuitively, this implies that, for all $i \in [L] \setminus \mathcal{T}$, $\mathcal{A}$ has never queried the random oracle for (d, i, ch, z) for some challenge $\mathsf{ch} \in S$. This means that $\mathcal{A}$ has not checked if these components could be target components. Similarly, for all $i \in \mathcal{T}$, $\mathcal{A}$ has never queried the random oracle for (d, i, ch, z_i) for another challenge $\mathsf{ch} \in S \setminus \{\mathsf{ch}_i\}$. This means that $\mathcal{A}$ must have guessed the right challenge for each target component in $\mathcal{T}$.

Therefore, $\mathcal{A}$ has sent only ρ meaningful queries $d_{i_1}, \ldots, d_{i_\rho}$ for the ρ components in $\mathcal{T}$. In order for the proof to be successfully verified, the transcripts associated to $d_{i_1}, \ldots, d_{i_\rho}$ must satisfy the inversion predicate, so the output of the random oracle on such queries $d_{i_1}, \ldots, d_{i_\rho}$ has to be smaller than $\lfloor 2^{\lambda-b} \rfloor$.

[14] By implicitly program we mean that it assigns the sampled digest to the input $(d, i, \mathsf{ch}, \cdot)$, where $\cdot$ will be substituted by a response z that the simulator can not compute on its own, but can verify if it receives the query from the distinguisher.

The random oracle queries are distinct, therefore the returned outputs — which are all smaller than $\lfloor 2^{\lambda-b} \rfloor$ — have been sampled uniformly at random and independently by the random oracle. Therefore the probability that these ρ digests are all smaller than $\lfloor 2^{\lambda-b} \rfloor$ is approximately but less than $(2^{-b})^{\rho}$. As a consequence,

$$\epsilon_s^{\mathsf{Ext}_\blacktriangle} \leq 2^{-b\rho}. \tag{4}$$

□

Before providing some data regarding the performance of the GAO transform, we introduce a technique to additionally improve its computational complexity.

5 Stretch-and-Compress Technique

We present a novel technique that can be used to improve the proof size and the computational complexity of NIZKPs obtained by applying the GAO transform[15]. This technique is particularly effective when the computational cost of creating first messages of Σ — the underlying Sigma protocol compiled via the GAO transform — exceeds *by far* the computational cost of hash queries. This is generally the case for the Sigma protocol $\Sigma_\star$ from cryptographic group actions.

We can define a new NIZKP $(\mathsf{Prove}_\blacksquare, \mathsf{Verify}_\blacksquare)$ that uses $(\mathsf{Prove}_\blacktriangle, \mathsf{Verify}_\blacktriangle)$ as a subroutine as follows: the subroutine algorithm $(\mathsf{Prove}_\blacktriangle, \mathsf{Verify}_\blacktriangle)$ is obtained by applying to Σ the GAO transform parametrized to achieve a *stretched* (larger than 2^{-40}) completeness error $\epsilon_c^{\mathsf{Prove}_\blacktriangle}$; the completeness error $\epsilon_c^{\mathsf{Prove}_\blacksquare}$ of $(\mathsf{Prove}_\blacksquare, \mathsf{Verify}_\blacksquare)$ is then *compressed* back to 2^{-40} by allowing $\mathsf{Prove}_\blacktriangle$ to interact with $k_{\max}$ distinct random oracles. For this reason, we call this general technique *Stretch-and-Compress*.

The main feature of this new compiler is the reduction of the number L of computationally expensive first messages of Σ that the prover must generate, at the expense of an increased number of computationally lighter hash queries the prover must perform.

More in detail, the proof-generation algorithm $\mathsf{Prove}_\blacksquare$ of the derived NIZKP generates the L first messages only once, then it tries to generate a proof π by interacting with up to $k_{\max}$ different random oracles. Basically, $\mathsf{Prove}_\blacksquare$ sequentially executes $\mathsf{Prove}_\blacktriangle^{\mathsf{H}_i}$ with different random oracles $\mathsf{H}_i, i \in [k_{\max}]$, until one of the executions of $\mathsf{Prove}_\blacktriangle^{\mathsf{H}_i}$ outputs a valid proof π'. As long as one of the $k_{\max}$ random oracles enables the subroutine $\mathsf{Prove}_\blacktriangle^{\mathsf{H}_i}$ to succeed, the algorithm $\mathsf{Prove}_\blacksquare$ will succeed. The resulting proof specifies the set of transcripts and the identifier of the random oracle that was used for their creation, so $\pi = (i, \pi')$.

[15] Actually, this technique can be applied to any NIZKP which requires the prover to perform expensive operations in a preliminary phase, while the success in the proof generation only depends on computationally cheap random-oracle queries.

5.1 The SC-GAO Transform

In this section we formally show how to use $(\mathsf{Prove}_{\blacktriangle}, \mathsf{Verify}_{\blacktriangle})$ as a subroutine for a novel NIZKP $(\mathsf{Prove}_{\blacksquare}, \mathsf{Verify}_{\blacksquare})$ obtained by applying to an underlying Sigma protocol Σ the SC-GAO transform.

Definition 4 (SC-GAO Transform)**.** *Let λ be a security parameter and let $\Sigma = (\mathsf{P}, \mathsf{V})$ be a seed-friendly Sigma protocol for a binary relation $\mathcal{R}$, with special challenge $\tilde{ch} = 0$. Let $k_{max}, L, \rho \in \mathbb{N}^*, b \in \mathbb{Q}$ with $0 < b < \lambda$, and let $\mathsf{H}_1, \ldots, \mathsf{H}_{k_{max}} \colon \{0,1\}^* \to \{0,1\}^\lambda$ be a collection of k_{max} independent random oracles. The* SC-GAO transform*, parametrized by $(L, b, \rho, \tilde{\mathsf{ch}}, k_{max})$, runs the algorithms $(\mathsf{Prove}_{\blacktriangle}, \mathsf{Verify}_{\blacktriangle})$ from Transformation 1, with parameter $(L, b, \rho, \tilde{\mathsf{ch}})$ and applied to Σ, to produce a NIZKP $(\mathsf{Prove}_{\blacksquare}, \mathsf{Verify}_{\blacksquare})$ as detailed in Transformation 2.*

Transformation 2: SC-GAO$[L, b, \rho, \tilde{\mathsf{ch}}, k_{\max}]$

Let $\Sigma = (\mathsf{P}, \mathsf{V})$ be a seed-friendly Sigma protocol for $\mathcal{R}$ and let $(x, w) \in \mathcal{R}$.

$\mathsf{Prove}_{\blacksquare}^{\{\mathsf{H}_i\}}(x, w)$:
1: $\overline{f} \leftarrow\$ \mathsf{Prove}_{\blacktriangle,1}(x, w)$
2: **for** $i \leftarrow 1, \ldots, k_{\max}$ **do**
3: $\quad \pi' \leftarrow\$ \mathsf{Prove}_{\blacktriangle,2}^{\mathsf{H}_i}(x, w, \overline{f})$
4: $\quad$ **if** $\pi' \neq \perp$ **then**
5: $\quad\quad$ **return** $\pi \leftarrow (i, \pi')$
6: **return** $\perp$

$\mathsf{Verify}_{\blacksquare}^{\{\mathsf{H}_i\}}(x, \pi = (i, \pi'))$:
1: **if** $i > k_{\max}$ **then**
2: $\quad$ **return** reject
3: **return** $\mathsf{Verify}_{\blacktriangle}^{\mathsf{H}_i}(x, \pi')$

The random oracles H_i can be obtained from a single random oracle H via oracle cloning [10]. For instance, we can define $\mathsf{H}_i(\cdot) = \mathsf{H}(\langle i \rangle \| \cdot)$, where $\langle i \rangle$ is a constant-size representation of the integer i. Note that for $k_{\max} = 1$ the SC-GAO transform (Definition 4) is essentially the GAO transform (Definition 3)[16].

Remark 4. The runtime of the algorithm $\mathsf{Verify}_{\blacksquare}$ is not affected by the use of multiple random oracles for the proof generation, because it receives the identifier of the random oracle used by the prover to generate the proof.

5.2 Security of the SC-GAO Transform

Let $\Sigma = (\mathsf{P}, \mathsf{V})$ be a seed-friendly Sigma protocol and $(\mathsf{Prove}_{\blacktriangle}, \mathsf{Verify}_{\blacktriangle})$ the NIZKP obtained by applying the GAO transform in Definition 3 with parameters $(L, b, \rho, \tilde{\mathsf{ch}})$. In the following theorem we characterize the security of the NIZKP obtained by applying the SC-GAO transform (Definition 4) to Σ, or equivalently the Stretch-and-Compress technique to $(\mathsf{Prove}_{\blacktriangle}, \mathsf{Verify}_{\blacktriangle})$.

[16] The only difference is the way the value d is computed, as $\mathsf{H}(1, x, \overline{f})$ instead of $\mathsf{H}(x, \overline{f})$.

Theorem 2. *Let λ be a security parameter and let $\Sigma = (\mathsf{P}, \mathsf{V})$ be a seed-friendly Sigma protocol for a binary relation $\mathcal{R}$ with special challenge $\tilde{\mathsf{ch}} = 0$, and $(\mathsf{Prove}_{\blacksquare}, \mathsf{Verify}_{\blacksquare})$ be the algorithms obtained by applying the SC-GAO transform (Definition 4) with parameters $(L, b, \rho, \tilde{\mathsf{ch}}, k_{max})$ to Σ. Let $(\mathsf{Prove}_{\blacktriangle}, \mathsf{Verify}_{\blacktriangle})$ be the subroutines obtained by applying the GAO transform with parameters $(L, b, \rho, \tilde{\mathsf{ch}})$ to Σ, and $\epsilon_c^{\mathsf{Prove}_{\blacktriangle}}, \epsilon_s^{\mathsf{Ext}_{\blacktriangle}}$ the corresponding completeness and soundness error determined according to Theorem 1. Then $(\mathsf{Prove}_{\blacksquare}, \mathsf{Verify}_{\blacksquare})$ is a straight-line extractable NIZKP for the relation $\mathcal{R}$ with:*

1. *completeness error $\epsilon_c^{\mathsf{Prove}_{\blacksquare}} = \left(\epsilon_c^{\mathsf{Prove}_{\blacktriangle}}\right)^{k_{max}}$;*
2. *soundness error $\epsilon_s^{\mathsf{Ext}_{\blacksquare}} = 1 - (1 - \epsilon_s^{\mathsf{Ext}_{\blacktriangle}})^{k_{max}}$ $(> \epsilon_s^{\mathsf{Ext}_{\blacktriangle}})$;*
3. *expected query complexity $\mathbb{E}[Q^{\mathsf{Prove}_{\blacksquare}}] \leq \frac{1}{1-\epsilon_c^{\mathsf{Prove}_{\blacktriangle}}} \cdot L \cdot \ell$, where $\ell = |\mathit{ChSpace}| - 1$.*

Proof sketch. **Completeness:** the proof exploits the independence of the random oracles $\mathsf{H}_1, \ldots, \mathsf{H}_{k_{\max}}$. An honest prover fails only when all $k_{\max}$ oracles fail, giving completeness error

$$\epsilon_c^{\mathsf{Prove}_{\blacksquare}} = \Pr[|\mathcal{T}_i| < \tau]^{k_{\max}} = \left(\epsilon_c^{\mathsf{Prove}_{\blacktriangle}}\right)^{k_{\max}}. \tag{5}$$

This enables exponential compression of the completeness error.

Zero-knowledge: Zero-knowledge follows the same arguments as Theorem 1 with the difference that it must take care also of the mapping of the queries to the $k_{\max}$ different random oracles. The simulator $\mathsf{Sim}_{\blacksquare}$ runs $\mathsf{Sim}_{\blacktriangle}$ as a subroutine sequentially across oracles, programming separate hash tables until it manages to generate its proof.

Extraction: The proof follows the proof of Theorem 1, but a dishonest adversary succeeds if any of the $k_{\max}$ oracles enables a forgery. The union bound yields a soundness error

$$\epsilon_s^{\mathsf{Ext}_{\blacksquare}} = 1 - (1 - \epsilon_s^{\mathsf{Ext}_{\blacktriangle}})^{k_{\max}} = 1 - (1 - 2^{-b\rho})^{k_{\max}}. \tag{6}$$

Clearly, $\epsilon_s^{\mathsf{Ext}_{\blacksquare}} > \epsilon_s^{\mathsf{Ext}_{\blacktriangle}}$, because $1 - \epsilon_s^{\mathsf{Ext}_{\blacksquare}} = (1 - \epsilon_s^{\mathsf{Ext}_{\blacktriangle}})^{k_{\max}} < (1 - \epsilon_s^{\mathsf{Ext}_{\blacktriangle}})$. □

For the full proof of the theorem see Section B.2.

5.3 Parameter Choice

The SC-GAO transform introduces a multi-dimensional parameter optimization problem due to the interaction between the stretch-and-compress technique and the underlying GAO parameters. We address the problem of minimizing computational cost subject to fixed security constraints and target weight ρ.

Applying the methodology established in Section 2, we seek to minimize an upper bound $\mathsf{CC}(\mathsf{Prove}_{\blacksquare})$ on the computational complexity of $\mathsf{Prove}_{\blacksquare}$. In particular, given a weight ρ, and target completeness and soundness errors $\epsilon_{\mathsf{C}}^{\mathsf{Prove}_{\blacksquare}}, \epsilon_{\mathsf{S}}^{\mathsf{Ext}_{\blacksquare}}$, we describe how to choose the parameters $(L, b, \rho, \tilde{\mathsf{ch}}, k_{\max})$ of

the NIZKP $(\mathsf{Prove}_{\blacksquare}, \mathsf{Verify}_{\blacksquare})$ so that the computational complexity of $\mathsf{Prove}_{\blacksquare}$ will be below $\mathsf{CC}(\mathsf{Prove}_{\blacksquare})$ with probability approximately 0.98. If we express the computational complexities w.r.t. CC_H, we can define $\gamma_f = \frac{\mathsf{CC}_f}{\mathsf{CC}_H}$ and $\gamma_z = \frac{\mathsf{CC}_z}{\mathsf{CC}_H}$. Then, the considered computational cost can be expressed as:

$$\begin{aligned} \mathsf{CC}(\mathsf{Prove}_{\blacksquare}) &\approx \left\lceil L \cdot \gamma_f \cdot \mathsf{CC}_H + \bar{Q} \cdot \mathsf{CC}_H + L \cdot \ell \cdot \gamma_z \cdot \mathsf{CC}_H \right\rceil \\ &= \left\lceil \mathsf{CC}_H \cdot L \left(\gamma_f + \bar{Q} + \ell \cdot \gamma_z\right) \right\rceil, \end{aligned} \tag{7}$$

where $\bar{Q}$ represents an upper bound on the number of hash queries that $\mathsf{Prove}_{\blacksquare}$ performs, set to three standard deviations above the expected value to ensure the bound holds with high probability. Notice that the number of hash queries $Q(\mathsf{Prove}_{\blacksquare})$ can be described as $L\ell \cdot X$, where X is the random variable counting the number of oracle queries made by the prover to get one success. In particular, X is distributed as a geometric distribution of parameter $p = 1 - \epsilon_{\mathsf{C}}{}^{\mathsf{Prove}_{\blacktriangle}}$. Hence, we obtain that $\bar{Q} = \frac{(1+3\sqrt{\epsilon_c^{\mathsf{Prove}_{\blacktriangle}}})L\ell}{1-\epsilon_c^{\mathsf{Prove}_{\blacktriangle}}}$ and the following one-sided tail bound for $Q(\mathsf{Prove}_{\blacksquare})$ easily follows:

$$\begin{aligned} \Pr\left[Q(\mathsf{Prove}_{\blacksquare}) \leq \bar{Q}\right] &= \Pr\left[X \leq \mathbb{E}[X] - 3\sqrt{\mathrm{Var}[X]}\right] = 1 - (1-p)^{\left\lfloor \frac{1+3\sqrt{1-p}}{p} \right\rfloor} \\ &\geq 1 - e^{-4} \approx 0.98. \end{aligned}$$

The key insight enabling optimization is that $k_{\max}$ controls a fundamental trade-off: smaller $k_{\max}$ values require more first messages L (expensive operations), while larger $k_{\max}$ values increase the expected number of hash queries (cheaper operations). Given target completeness error $\epsilon_c^{\mathsf{Prove}_{\blacksquare}} = 2^{-40}$ [24] and soundness error $\epsilon_s^{\mathsf{Ext}_{\blacksquare}} = 2^{-\lambda}$, the parameter dependencies follow the relationships shown in Figure 1. More in detail, the stretched completeness error of the underlying GAO transform becomes $\epsilon_c^{\mathsf{Prove}_{\blacktriangle}} = (2^{-40})^{1/k_{\max}}$ (Equation (5)), while the corresponding soundness error is $\epsilon_s^{\mathsf{Ext}_{\blacktriangle}} = 1 - (1 - 2^{-\lambda})^{1/k_{\max}}$ (Equation (6)). Finally, from Equation (4) we set $b = -\frac{\log_2(\epsilon_s^{\mathsf{Ext}_{\blacktriangle}})}{\rho}$, and determine the minimal L satisfying the completeness constraint via binary search on Equation (3).

The resulting optimization problem exhibits unimodal behavior in $k_{\max}$: the function $\mathsf{CC}(\mathsf{Prove}_{\blacksquare})$ has a single local minimum, enabling efficient optimization via ternary search. The optimal value $k^*_{\max}$ balances the competing costs of first message generation and hash computation according to the specific computational profile characterized by γ_f and γ_z.

Remark 5. The stretch-and-compress technique can be adapted to be employed with the NIZKP obtained by compiling a Sigma protocol with the Fischlin transform. However the choice of optimal parameters is not as simple as it is for the GAO transform (see Section C.2).

Table 3 presents optimal parameterizations for various challenge space sizes $\ell + 1$ and target weights ρ, demonstrating computational savings compared to the direct GAO application ($k_{\max} = 1$). We notice that, while these results

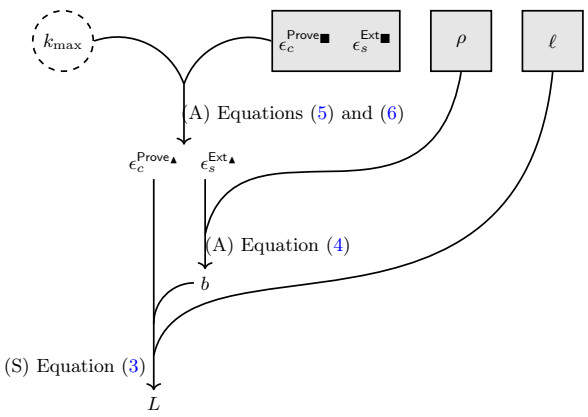


Fig. 1: Graphical description of the dependency of parameters in Definition 4. Given the values $\epsilon_c^{\mathsf{Prove}\blacksquare}, \epsilon_s^{\mathsf{Ext}\blacksquare}, \rho, \ell$ (in boxes filled in gray), express the values $\epsilon_c^{\mathsf{Prove}\blacktriangle}, \epsilon_s^{\mathsf{Ext}\blacktriangle}, b, L$ as functions of the maximum number $k_{\max}$ of random oracles that the prover can use (in the dashed circle). The arrows describe the nature of the dependency: the parameters at the tail of the arrows determine the value of the parameters at the head. This dependency is determined by the equation specified alongside the arrow. If the relation is analytical, i.e. we have an exact formula for the parameters at the head, we write (A) next to the equation while, if the parameter at the head of the arrow must be found using binary search, we write (S).

provide concrete performance comparisons against the official LESS implementation parameters, they are not optimal for the target signature size. Additional parameterizations are shown in Appendix E.

6 The Collision Predicate: SC-Coll-GAO Transform

In this section, we present an optimization to the GAO transform (Definition 3) which consists in applying a collision predicate similar the one proposed by Kondi and shelat [54] to optimize the Fischlin transform. We refer to this as the Collision GAO (Coll-GAO) transform, which is then used as a subroutine NIZKP for the Stretch-and-Compress technique introduced in Section 5, where it was applied to the GAO transform in Definition 4. In this way, we obtain our final (and most efficient) straight-line transform, i.e. SC-Coll-GAO.

Optimization by Kondi and shelat. In [54], the authors propose a modification of the Fischlin transform that leaves the framework unchanged but modifies the predicate to be satisfied by the transcripts. More specifically, while Fischlin proposes an inversion predicate where the prover searches transcripts for which the associated digest is 0^b, Kondi and shelat propose a *collision predicate* involving the division of transcripts into non-crossing pairs and the requirement that transcripts within each pair must be associated with the same digest in $\{0,1\}^{2b}$.

Table 3: Optimal parameters for GAO (Definition 3) and SC-GAO (Definition 4) to minimize the computational complexity, for a given choice of ℓ and ρ. The computational complexities and signature sizes are compared in gray with the LESS parameterizations (Table 1).

Transform	ℓ	ρ	k	b	L	$\bar{Q}$	CC	**sig (B)**
GAO	1	36	1	3.56	1094	1.1K	663M (×5.9)	4097 (×1.6)
SC-GAO			301	3.78	392	17.2K	254M (×2.3)	3233 (×1.2)
GAO	3	42	1	3.05	293	880	181M (×4.4)	3329 (×1.8)
SC-GAO			92	3.20	131	5.39K	86M (×2.1)	2529 (×1.4)
GAO	7	34	1	3.76	191	1.34K	123M (×4.5)	2593 (×2.0)
SC-GAO			62	3.94	85	5.58K	59.6M (×2.2)	1905 (×1.4)

This approach results in a reduced query complexity. However, reduced query complexity translates to reduced challenge weight if we consider the application of this transform to the Sigma protocol $\Sigma_\star$, or more generally to seed-friendly Sigma protocols. The authors show that the complexity of the query using this approach is reduced by 10–15% for values of $b \leq 8$ [54, Section 5.1] and this increases the efficiency by reducing the number of challenges to try to find a solution. The only downside is that, unlike the inversion predicate, the collision predicate requires the prover to remember the digests found while looking for a collision (and the associated challenge). However, this is not a significant issue unless the prover's storage availability is critically limited.

Our Collision Predicate. In the following, we consider a slight generalization that allows for more fine-grained control over the collision probability.

Definition 5. *Let λ be a security parameter, let $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^\lambda$ and let $b \in \mathbb{Q}, 0 < b < \lambda$. We interpret the output of H as an integer in $[0, 2^\lambda - 1]$ and we partition the set $[0, 2^\lambda - 1]$ into $k+1$ intervals according to a suitable partition $\mathcal{P}_{k,q,r}$ of $[0, 2^\lambda - 1]$, where $2^\lambda = k \cdot q + r$ with*

$$k = \left\lceil \frac{1-\beta}{\beta} \right\rceil \quad \text{and} \quad q = \left\lceil \frac{2^\lambda k - 2^\lambda \sqrt{\beta k^2 - (1-\beta)k}}{k^2 + k} \right\rceil . \tag{8}$$

In particular, the first k intervals of the partitions have length q and the last interval has length r. Then, we say that the collision predicate is satisfied for $x_1, x_2 \in \{0,1\}^*$ if and only if either $(t-1) \cdot q \leq \mathsf{H}(x_1), \mathsf{H}(x_2) < t \cdot q$ for $t \in [1, k]$ or $k \cdot q \leq \mathsf{H}(x_1), \mathsf{H}(x_2) < 2^\lambda$. With a slight abuse of notation, we will denote the verification of the collision predicate on x_1, x_2 by $\mathsf{H}_{2b}(x_1) = \mathsf{H}_{2b}(x_2)$, where $\mathsf{H}_{2b}(x)$ maps x to one of the intervals defined above.

By taking k and q as in Equation (8), we ensure that the collision predicate for any $x_1, x_2 \in \{0,1\}^*$ is satisfied with probability negligibly smaller than $\beta = 2^{-2b}$

for any $b \in \mathbb{Q}$ with $0 < b \ll \lambda$[17]. A detailed description of why we define our collision predicate as above is included in Section D.2 where we also show that our approach allows us to obtain a collision probability that is negligibly close to the target $\beta = 2^{-2b}$.

In the remainder of this section, we build on the insights of Kondi and shelat [54] to enhance the NIZKP obtained using the GAO transform introduced in Definition 3.

6.1 The Coll-GAO Transform

If we consider only even values of ρ, we can describe the Coll-GAO transform as an optimization of the GAO transform.

Overview of the Coll-GAO Transform. The prover $\mathsf{Prove}_{\blacklozenge}$, as in Definition 3, generates a set L of first messages. Then, it starts looping over $i \in [L]$ computing for each first message f_i, and challenges $\mathsf{ch}_j \in \mathsf{ChSpace} \setminus \tilde{\mathsf{ch}}$ the digests $d_{i,j} \in \{0, \ldots, 2^\lambda\}$ associated to the transcripts $f_i, \mathsf{ch}_j, z_{i,j}$. Instead of looking for a digest $\leq \lfloor 2^{\lambda - b} \rfloor$ the prover stores the intervals of the partition $\mathcal{P}_{k,q,r}$ the digests $d_{i,j}$ belong to, as long as it finds two distinct first messages (at indices (i_1, i_2)) for which two digests satisfy the collision predicate of Definition 5. When this happens, we say that it has found a collision, and it interrupts the search deleting all the digests stored during the search, and starts again from the following first message (at index $i_2 + 1$) the search for a new collision. This is repeated until it finds $\frac{\rho}{2}$ pairs. Note that, in this way, the indices of the colliding transcripts will be non-crossing, meaning that no index of one pair lies between the two indices of another pair. This is crucial to guarantee the soundness of our transform.

In the end, the proof will consist of the $\frac{\rho}{2}$ pairs of colliding transcripts, with the challenges that enable the collision, and the transcripts of all the other $L - \rho$ first messages, that will be associated with the challenge $\tilde{\mathsf{ch}}$.

Now we formally define the Coll-GAO transform.

Definition 6 (Coll-GAO Transform)**.** *Let λ be a security parameter and let $\Sigma = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ be a seed-friendly Sigma protocol for a binary relation $\mathcal{R}$, with challenge space $\mathsf{ChSpace}$. Let $L, \rho \in \mathbb{N}$, with ρ even, $b \in \mathbb{Q}$ with $0 < b < \lambda$, let $\tilde{\mathsf{ch}} = 0$ be the special challenge of Σ, and let $\mathsf{H} \colon \{0,1\}^* \to \{0,1\}^\lambda$ be a random oracle. The Coll-GAO transform, parametrized by $(L, b, \rho, \tilde{\mathsf{ch}})$, applied on Σ produces a NIZKP $(\mathsf{Prove}_{\blacklozenge}, \mathsf{Verify}_{\blacklozenge})$ as defined in Transformation 3*

Remark 6. This optimization is inspired by the observation of Kondi and shelat [54], but in our construction this approach is even more effective since the collision is not searched among two single pools of digests, but among a number of pools of ℓ digests that increases as the search for collision goes on.

[17] Note that b can not be too large, because the query complexity of the transform grows exponentially with b.

Transformation 3: Coll-GAO$[L, b, \rho, \tilde{\mathsf{ch}}]$

Let $\Sigma = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ be a seed-friendly Sigma protocol for $\mathcal{R}$ and let $(x, w) \in \mathcal{R}$. For an empty table $\mathcal{D}$, we follow the convention that each entry is initialized as $\perp$. In particular, we say that $d \in \mathcal{D}$ if $\mathcal{D}[d] \neq \perp$.

$\mathsf{Prove}_{\blacklozenge,1}(x, w)$:
1: **for** $i \leftarrow 1, \ldots, L$ **do**
2: $\quad f_i \leftarrow\!\!\$\ \mathsf{P}_1(x, w)$
3: $\overline{f} \leftarrow (f_1, \ldots, f_L)$

$\mathsf{Prove}^{\mathsf{H}}_{\blacklozenge,2}(x, w, \overline{f})$:
1: $\mathcal{P} \leftarrow$ empty table ▷ Init transcripts table
2: $\mathcal{D} \leftarrow$ empty table ▷ Init collision table
3: $d \leftarrow \mathsf{H}(x, \overline{f})$
4: **for** $i \leftarrow 1, \ldots, L$ **do**
5: $\quad$ **for** $\mathsf{ch} \in \mathsf{ChSpace} \setminus \{\tilde{\mathsf{ch}}\}$ **do**
6: $\quad\quad z_{i,\mathsf{ch}} \leftarrow\!\!\$\ \mathsf{P}_2(x, w, f_i, \mathsf{ch})$
7: $\quad\quad d_{i,\mathsf{ch}} \leftarrow \mathsf{H}_{2b}(d, i, \mathsf{ch}, z_{i,\mathsf{ch}})$
8: $\quad\quad$ **if** $\mathsf{CheckColl}(\mathcal{D}, d_{i,\mathsf{ch}}, i) \rightarrow (j, \mathsf{ch}', z_{j,\mathsf{ch}'}) \neq \perp$ **then**
9: $\quad\quad\quad \mathcal{P}[i] \leftarrow (\mathsf{ch}, z_{i,\mathsf{ch}})$
10: $\quad\quad\quad \mathcal{P}[j] \leftarrow (\mathsf{ch}', z_{j,\mathsf{ch}'})$
11: $\quad\quad\quad \mathcal{D} \leftarrow$ empty table
12: $\quad\quad\quad$ **break**
13: $\quad\quad$ **else**
14: $\quad\quad\quad \mathcal{D}[d_{i,\mathsf{ch}}] \leftarrow \mathcal{D}[d_{i,\mathsf{ch}}] \cup (i, \mathsf{ch}, z_{i,\mathsf{ch}})$
15: $\quad$ **if** $|\mathcal{P}| = \rho$ **then**
16: $\quad\quad$ **break**
17: **if** $|\mathcal{P}| < \rho$ **then**
18: $\quad$ **return** $\perp$
19: **for** $i \leftarrow \{\, i \in [L] \mid i \notin \mathcal{P} \,\}$ **do**
20: $\quad z_i \leftarrow\!\!\$\ \mathsf{P}_2(x, w, f_i, \tilde{\mathsf{ch}})$
21: $\quad \mathcal{P}[i] \leftarrow (i, \tilde{\mathsf{ch}}, z_i)$
22: **return** $\pi \leftarrow (\overline{f}, \mathcal{P})$

$\mathsf{Prove}^{\mathsf{H}}_{\blacklozenge}(x, w)$:
1: $\overline{f} \leftarrow\!\!\$\ \mathsf{Prove}_{\blacklozenge,1}(x, w)$
2: $\pi \leftarrow\!\!\$\ \mathsf{Prove}^{\mathsf{H}}_{\blacklozenge,2}(x, w, \overline{f})$
3: **return** π

$\mathsf{Verify}^{\mathsf{H}}_{\blacklozenge}(x, \pi = (\overline{f}, \mathcal{P}))$:
1: $d \leftarrow \mathsf{H}(x, \overline{f})$
2: **for** $i \leftarrow 1, \ldots, L$ **do**
3: $\quad (\mathsf{ch}_i, z_i) \leftarrow \mathcal{P}[i]$
4: $\mathcal{T} \leftarrow \{\, i \in [L] \mid \mathsf{ch}_i \neq \tilde{\mathsf{ch}} \,\}$
5: **if** $|\mathcal{T}| \neq \rho$ **then**
6: $\quad$ **return** reject
7: **for** $i \leftarrow 1, \ldots, L$ **do**
8: $\quad$ **if** $\mathsf{V}_2(x, f_i, \mathsf{ch}_i, z_i) =$ reject **then**
9: $\quad\quad$ **return** reject
10: **for** $j \leftarrow 1, \ldots, \rho/2$ **do**
11: $\quad (i_{j_1}, i_{j_2}) \leftarrow (\mathcal{T}[2j-1], \mathcal{T}[2j])$
12: $\quad$ **if** $\mathsf{H}_{2b}(d, i_{j_1}, \mathsf{ch}_{i_{j_1}}, z_{i_{j_1}}) \neq \mathsf{H}_{2b}(d, i_{j_2}, \mathsf{ch}_{i_{j_2}}, z_{i_{j_2}})$ **then**
13: $\quad\quad$ **return** reject
14: **return** accept

$\mathsf{CheckColl}(\mathcal{D}, d, i)$:
1: **if** $d \notin \mathcal{D}$ **then**
2: $\quad$ **return** $\perp$
3: **for** $(j, \mathsf{ch}', z_{j,\mathsf{ch}'}) \leftarrow \mathcal{D}[d]$ **do**
4: $\quad$ **if** $j \neq i$ **then**
5: $\quad\quad$ **return** $(j, \mathsf{ch}', z_{j,\mathsf{ch}'})$
6: **return** $\perp$

6.2 Security of the Coll-GAO Transform

Theorem 3. *Let λ be a security parameter and let $\Sigma = (\mathsf{P}, \mathsf{V})$ be a seed-friendly Sigma protocol for relation $\mathcal{R}$, with challenge space $\mathsf{ChSpace}$ and special challenge $\tilde{\mathsf{ch}} = 0$. Let $(\mathsf{Prove}_{\blacklozenge}, \mathsf{Verify}_{\blacklozenge})$ be the non-interactive proof system obtained by applying the Coll-GAO transform (Transformation 3) to Σ, parametrized by $(L, b, \rho, \tilde{\mathsf{ch}})$, with L and ρ non-zero natural numbers, and b a positive rational less than λ. Then $(\mathsf{Prove}_{\blacklozenge}, \mathsf{Verify}_{\blacklozenge})$ is a straight-line extractable NIZKP for the relation $\mathcal{R}$ (in the random oracle model) with:*

1. *completeness error $\epsilon_c^{\mathsf{Prove}_{\blacklozenge}}(L, b, \rho, \ell) < \epsilon_c^{\mathsf{Prove}_{\blacktriangle}}(L, b, \rho, \ell)$;*
2. *soundness error $\epsilon_s^{\mathsf{Ext}_{\blacklozenge}} \leq 2^{-b\rho}$;*
3. *query complexity $Q^{\mathsf{Prove}_{\blacklozenge}} \leq L \cdot \ell$.*

where $\ell = |\mathsf{ChSpace}| - 1$.

The techniques used to prove this theorem are analogous to the ones used to prove Theorem 1, using some additional observations from [54].

About the straight-line extraction, note that the collision predicate of Definition 5 guarantees that an adversary can guess a pair of target components and the associated challenges yielding a collision with probability $\leq 2^{-2b}$, and to generate a valid proof the adversary must guess $\frac{\rho}{2}$ pairs. The rest of the proof follows the proof of Theorem 1.

For a proof sketch see Section B.3.

6.3 Applying the Stretch-and-Compress Technique to Coll-GAO

In this section we describe how to apply the Stretch and Compress technique to the Coll-GAO transform, and this allows us to define the SC-Coll-GAO transform.

Definition 7 (SC-Coll-GAO Transform)**.** *Let λ be a security parameter and let $\Sigma = (\mathsf{P}, \mathsf{V})$ be a seed-friendly Sigma protocol for a binary relation $\mathcal{R}$. Let $k_{max}, L, \rho \in \mathbb{N}, b \in \mathbb{Q}$ with $0 < b < \lambda$, let $\tilde{\mathsf{ch}}$ be the special challenge of Σ, and let $\mathsf{H}_1, \ldots, \mathsf{H}_{k_{max}} \colon \{0,1\}^* \to \{0,1\}^\lambda$ be a collection of k_{max} independent random oracles. The* Stretch-and-Compress Coll-GAO *transform, parametrized by $(L, b, \rho, \tilde{\mathsf{ch}}, k_{max})$, applies the algorithms $(\mathsf{Prove}_\blacklozenge, \mathsf{Verify}_\blacklozenge)$ from Definition 6 with parameter $(L, b, \rho, \tilde{\mathsf{ch}})$ to Σ. Then, produces a NIZKP $(\mathsf{Prove}_\bigstar, \mathsf{Verify}_\bigstar)$ as detailed in Transformation 4.*

Transformation 4: SC-Coll-GAO$[L, b, \rho, \tilde{\mathsf{ch}}, k_{\max}]$

Let $\Sigma = (\mathsf{P}, \mathsf{V})$ be a seed-friendly Sigma protocol for $\mathcal{R}$ and let $(x, w) \in \mathcal{R}$.

$\mathsf{Prove}_\bigstar^{\{\mathsf{H}_i\}}(x, w)$:
1: $\overline{f} \leftarrow\$ \mathsf{Prove}_{\blacklozenge,1}(x, w)$
2: **for** $i \leftarrow 1, \ldots, k_{\max}$ **do**
3: $\quad \pi' \leftarrow\$ \mathsf{Prove}_{\blacklozenge,2}^{\mathsf{H}_i}(x, w, \overline{f})$
4: $\quad$ **if** $\pi' \neq \perp$ **then**
5: $\quad\quad$ **return** $\pi \leftarrow (i, \pi')$
6: **return** $\perp$

$\mathsf{Verify}_\bigstar^{\{\mathsf{H}_i\}}(x, \pi = (i, \pi'))$:
1: **if** $i > k_{\max}$ **then**
2: $\quad$ **return** reject
3: **return** $\mathsf{Verify}_\blacklozenge^{\mathsf{H}_i}(x, \pi')$

Theorem 4. *Let $\Sigma = (\mathsf{P}, \mathsf{V})$ be a seed-friendly Sigma protocol for $\mathcal{R}$, and $(\mathsf{Prove}_\bigstar, \mathsf{Verify}_\bigstar)$ be the algorithms obtained by applying the SC-Coll-GAO transform (Transformation 4) with parameters $(L, b, \rho, \tilde{\mathsf{ch}}, k_{max})$ to Σ. Let $(\mathsf{Prove}_\blacklozenge, \mathsf{Verify}_\blacklozenge)$ be the subroutines obtained by applying the Coll-GAO transform with parameters $(L, b, \rho, \tilde{\mathsf{ch}})$ to Σ, and $\epsilon_c^{\mathsf{Prove}_\blacklozenge}, \epsilon_s^{\mathsf{Ext}_\blacklozenge}$ the corresponding completeness and soundness error determined according to Theorem 3. Then $(\mathsf{Prove}_\bigstar, \mathsf{Verify}_\bigstar)$ is a straight-line extractable NIZKP for the relation $\mathcal{R}$ with:*

1. *completeness error $\epsilon_c^{\mathsf{Prove}_\bigstar} = \left(\epsilon_c^{\mathsf{Prove}_\blacklozenge}\right)^{k_{max}}$;*
2. *soundness error $\epsilon_s^{\mathsf{Ext}_\bigstar} = 1 - (1 - \epsilon_s^{\mathsf{Ext}_\blacklozenge})^{k_{max}}$ $(> \epsilon_s^{\mathsf{Ext}_\blacklozenge})$;*
3. *expected query complexity $\mathbb{E}[Q^{\mathsf{Prove}_\bigstar}] \leq \frac{1}{1-\epsilon_c^{\mathsf{Prove}_\blacklozenge}} \cdot L \cdot \ell$, where $\ell = |\mathit{ChSpace}| - 1$.*

The proof can be obtained by using similar arguments as the ones in the proof of Theorem 2. The complete proof is given in Section B.4.

The generation of parameters for the SC-Coll-GAO transform is performed as it is described in Figure 1. However, instead of computing L via Equation (3) and binary search, L must be found empirically simulating the execution of the $\mathsf{Prove}_{\blacklozenge}$ algorithm. The simulations allow to estimate $\epsilon_c^{\mathsf{Prove}_{\blacklozenge}}$ using the same methodology described in Section 3.1. The empirical estimation is considerably more efficient and straightforward compared to the Fixed-Weight Fischlin transform. This allows us to increase accuracy by raising the target number of successes to $N = 7000$, ensuring the normalized mean absolute error of the estimated completeness error remains below 1%.

Table 4: Optimal parameters for SC-Coll-GAO (Definition 7) to minimize the computational complexity, for a given choice of ℓ and ρ. For ease of comparison, the corresponding parameters for SC-GAO (Table 3) are also shown. The computational complexities and signature sizes are compared in gray with the LESS parameterizations (Table 1).

Transform	ℓ	ρ	k	b	L	$\bar{Q}$	CC	**sig (B)**
SC-GAO	1	36	301	3.78	392	17.2K	254M (×2.3)	3233 (×1.2)
SC-GAO-Coll			196	3.77	279	8K	177M (×1.6)	2977 (×1.1)
SC-GAO	3	42	92	3.20	131	5.39K	86M (×2.1)	2529 (×1.4)
SC-GAO-Coll			57	3.19	97	2.53K	62.2M (×1.5)	2241 (×1.2)
SC-GAO	7	34	62	3.94	85	5.58K	59.6M (×2.2)	1905 (×1.4)
SC-GAO-Coll			33	3.91	65	2.37K	43.6M (×1.6)	1649 (×1.2)

Table 4 presents optimal parametrizations for the SC-Coll-GAO transform across the challenge space sizes and target weights corresponding to the official LESS parameterizations shown in Table 1. The collision-based predicate consistently improves upon SC-GAO in both computational complexity and signature size, with SC-Coll-GAO reducing computational overhead by approximately 25–30% relative to SC-GAO. When compared to the Fiat-Shamir-based LESS baseline, SC-Coll-GAO incurs modest penalties: computational costs increase by 50–60%, while signature sizes grow by only 10–20%. Additional parameterizations are shown in Appendix E.

References

1. Abdalla, M., An, J.H., Bellare, M., Namprempre, C.: From identification to signatures via the fiat-shamir transform: Minimizing assumptions for security and forward-security. In: Advances in Cryptology—EUROCRYPT 2002: International Conference on the Theory and Applications of Cryptographic Techniques Amsterdam, The Netherlands, April 28–May 2, 2002 Proceedings 21. pp. 418–433. Springer (2002)

2. Alamati, N., De Feo, L., Montgomery, H., Patranabis, S.: Cryptographic group actions and applications. In: Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II 26. pp. 411–439. Springer (2020)
3. Ambainis, A., Rosmanis, A., Unruh, D.: Quantum attacks on classical proof systems: The hardness of quantum rewinding. In: 2014 IEEE 55th Annual Symposium on Foundations of Computer Science. p. 474–483. IEEE (Oct 2014), http://dx.doi.org/10.1109/FOCS.2014.57
4. Atapoor, S., Baghery, K., Cozzo, D., Pedersen, R.: CSI-SharK: CSI-FiSh with Sharing-friendly Keys, p. 471–502. Springer Nature Switzerland (2023), http://dx.doi.org/10.1007/978-3-031-35486-1_21
5. Baldi, M., Barenghi, A., Beckwith, L., Biasse, J., Chou, T., Esser, A., Gaj, K., Karl, P., Mohajerani, K., Pelosi, G., Persichetti, E., Saarinen, M.O., Santini, P., Wallace, R., Zweydinger, F.: LESS — Linear Equivalence Signature Scheme. Tech. rep., National Institute of Standards and Technology (2024), available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures
6. Baldi, M., Bitzer, S., Pavoni, A., Santini, P., Wachter-Zeh, A., Weger, V.: Zero Knowledge Protocols and Signatures from the Restricted Syndrome Decoding Problem, p. 243–274. Springer Nature Switzerland (2024)
7. Battagliola, M., Borin, G., Di Crescenzo, G., Meneghetti, A., Persichetti, E.: Enhancing threshold group action signature schemes: Adaptive security and scalability improvements. In: International Conference on Post-Quantum Cryptography. pp. 129–161. Springer (2025)
8. Battagliola, M., Longo, R., Pintore, F., Signorini, E., Tognolini, G.: A revision of cross security: Proofs and attacks for multi-round fiat–shamir signatures. Mediterranean Journal of Mathematics **22**(5) (Jul 2025)
9. Battagliola, M., Longo, R., Pintore, F., Signorini, E., Tognolini, G.: Security of fixed-weight repetitions of special-sound multi-round interactive proofs. Designs, Codes and Cryptography (Jul 2025)
10. Bellare, M., Davis, H., Günther, F.: Separate Your Domains: NIST PQC KEMs, Oracle Cloning and Read-Only Indifferentiability, p. 3–32. Springer International Publishing (2020)
11. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Proceedings of the 13th ACM conference on Computer and communications security. pp. 390–399 (2006)
12. Beullens, W., Dobson, S., Katsumata, S., Lai, Y.F., Pintore, F.: Group signatures and more from isogenies and lattices: Generic, simple, and efficient. In: Designs, Codes and Cryptography,. pp. 91(6), 2141–2200. Springer (2023)
13. Beullens, W., Katsumata, S., Pintore, F.: Calamari and falafl: logarithmic (linkable) ring signatures from isogenies and lattices. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 464–492. Springer (2020)
14. Beullens, W., Kleinjung, T., Vercauteren, F.: Csi-fish: efficient isogeny based signatures through class group computations. In: International conference on the theory and application of cryptology and information security. pp. 227–247. Springer (2019)
15. Biasse, J.F., Micheli, G., Persichetti, E., Santini, P.: Less is more: code-based signatures without syndromes. In: Progress in Cryptology-AFRICACRYPT 2020: 12th International Conference on Cryptology in Africa, Cairo, Egypt, July 20–22, 2020, Proceedings 12. pp. 45–65. Springer (2020)

16. Borin, G., Persichetti, E., Pintore, F., Reijnders, K., Santini, P.: A guide to the design of digital signatures based on cryptographic group actions: A guide to the design of digital signatures. J. Cryptol. **38**(3) (Jun 2025)
17. Brassard, G., Yung, M.: One-way group actions. In: Advances in Cryptology-CRYPTO'90: Proceedings 10. pp. 94–107. Springer (1991)
18. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: International Conference on Security in Communication Networks. pp. 268–289. Springer (2002)
19. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Annual international cryptology conference. pp. 56–72. Springer (2004)
20. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Proceedings 42nd IEEE Symposium on Foundations of Computer Science. pp. 136–145. IEEE (2001)
21. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: Csidh: an efficient post-quantum commutative group action. In: Advances in Cryptology–ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part III 24. pp. 395–427. Springer (2018)
22. Chairattana-Apirom, R., Harding, F., Lysyanskaya, A., Tessaro, S.: Server-aided anonymous credentials. In: Annual International Cryptology Conference. pp. 291–324. Springer (2025)
23. Chen, M., Dey, P., Ganesh, C., Mukherjee, P., Sarkar, P., Sasmal, S.: Universally composable non-interactive zero-knowledge from sigma protocols via a new straight-line compiler. In: IACR International Conference on Public-Key Cryptography. pp. 381–417. Springer (2025)
24. Chen, Y.H., Lindell, Y.: Optimizing and implementing fischlin's transform for uc-secure zero knowledge. IACR Communications in Cryptology (Jul 2024)
25. Chou, T., Niederhagen, R., Persichetti, E., Randrianarisoa, T.H., Reijnders, K., Samardjiska, S., Trimoska, M.: Take your meds: digital signatures from matrix code equivalence. In: International conference on cryptology in Africa. pp. 28–52. Springer (2023)
26. Chou, T., Persichetti, E., Santini, P.: On linear equivalence, canonical forms, and digital signatures. Designs, Codes and Cryptography pp. 1–43 (2025)
27. Chou, T., Persichetti, E., Santini, P.: On linear equivalence, canonical forms, and digital signatures. Designs, Codes and Cryptography **93**(7), 2415–2457 (Mar 2025)
28. Dagdelen, Ö., Venturi, D.: A second look at fischlin's transformation. In: Progress in Cryptology–AFRICACRYPT 2014: 7th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 28-30, 2014. Proceedings 7. pp. 356–376. Springer (2014)
29. Damgård, I.: On σ-protocols. Lecture Notes, University of Aarhus, Department for Computer Science **84** (2002)
30. Dartois, P., Eriksen, J.K., Fouotsa, T.B., Herlédan Le Merdy, A., Invernizzi, R., Robert, D., Rueger, R., Vercauteren, F., Wesolowski, B.: PEGASIS: Practical Effective Class Group Action using 4-Dimensional Isogenies, p. 67–99. Springer Nature Switzerland (2025), http://dx.doi.org/10.1007/978-3-032-01855-7_3
31. De Feo, L., Galbraith, S.D.: Seasign: compact isogeny signatures from class group actions. In: Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III 38. pp. 759–789. Springer (2019)

32. Doerner, J., Kondi, Y., Lee, E., Shelat, A.: Secure two-party threshold ecdsa from ecdsa assumptions. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 980–997. IEEE (2018)
33. Doerner, J., Kondi, Y., Lee, E., Shelat, A., Tyner, L.: Threshold bbs+ signatures for distributed anonymous credential issuance. In: 2023 IEEE Symposium on Security and Privacy (SP). pp. 773–789. IEEE (2023)
34. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Efficient nizks and signatures from commit-and-open protocols in the qrom. In: Annual International Cryptology Conference. pp. 729–757. Springer (2022)
35. Ducas, L., van Woerden, W.: On the lattice isomorphism problem, quadratic forms, remarkable lattices, and cryptography. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 643–673. Springer (2022)
36. D'Alconzo, G., Flamini, A., Gangemi, A.: Non-interactive commitment from non-transitive group actions. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 222–252. Springer (2023)
37. D'Alconzo, G., Flamini, A., Meneghetti, A., Signorini, E.: A framework for group action-based multi-signatures and applications to less, meds, and alteq. In: Public-Key Cryptography – PKC 2025. p. 99–133. Springer Nature Switzerland (2025)
38. El Kaafarani, A., Katsumata, S., Pintore, F.: Lossy CSI-FiSh: Efficient Signature Scheme with Tight Reduction to Decisional CSIDH-512, p. 157–186. Springer International Publishing (2020), http://dx.doi.org/10.1007/978-3-030-45388-6_6
39. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Conference on the theory and application of cryptographic techniques. pp. 186–194. Springer (1986)
40. Fischlin, M.: Communication-efficient non-interactive proofs of knowledge with online extractors. In: Annual International Cryptology Conference. pp. 152–168. Springer (2005)
41. Flamini, A., Lee, E., Lysyanskaya, A.: Multi-holder anonymous credentials from bbs signatures. In: Advances in Cryptology – CRYPTO 2025. p. 325–357. Springer Nature Switzerland (2025)
42. Fujisaki, E., Suzuki, K.: Traceable ring signature. In: International Workshop on Public Key Cryptography. pp. 181–200. Springer (2007)
43. Ghantous, W., Pintore, F., Veroni, M.: Efficiency of sidh-based signatures (yes, sidh). Journal of Mathematical Cryptology **18.1**, 20230023 (2024)
44. Gilchrist, V., Marco, L., Petit, C., Tang, G.: Solving the tensor isomorphism problem for special orbits with low rank points: Cryptanalysis and repair of an asiacrypt 2023 commitment scheme. In: Annual International Cryptology Conference. pp. 141–173. Springer (2024)
45. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. Journal of the ACM (JACM) **38**(3), 690–728 (1991)
46. Hanzlik, L., Lai, Y.F., Mula, M., Paracucchi, E., Slamanig, D., Tang, G.: Tanuki: New frameworks for (concurrently secure) blind signatures from post-quantum groups actions. Cryptology ePrint Archive, Paper 2025/1100 (2025), https://eprint.iacr.org/2025/1100
47. Hesse, J., Singh, N., Sorniotti, A.: How to bind anonymous credentials to humans. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 3047–3064 (2023)

48. Ji, Z., Qiao, Y., Song, F., Yun, A.: General linear group action on tensors: A candidate for post-quantum cryptography. In: Theory of cryptography conference. pp. 251–281. Springer (2019)
49. Jiang, K., Wang, A., Luo, H., Liu, G., Tang, G., Pan, Y., Wang, X.: Re-randomize and extract: A novel commitment construction framework based on group actions. In: Advances in Cryptology – EUROCRYPT 2025. p. 124–153. Springer Nature Switzerland (2025)
50. Katsumata, S.: A new simple technique to bootstrap various lattice zero-knowledge proofs to qrom secure nizks. In: Annual International Cryptology Conference. pp. 580–610. Springer (2021)
51. Katsumata, S., Lai, Y.F., LeGrow, J.T., Qin, L.: CSI-Otter: Isogeny-Based (Partially) Blind Signatures from the Class Group Action with a Twist, p. 729–761. Springer Nature Switzerland (2023)
52. Katsumata, S., Reichle, M., Sakai, Y.: Practical round-optimal blind signatures in the rom from standard assumptions. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 383–417. Springer (2023)
53. Klooß, M., Reichle, M., Wagner, B.: Practical blind signatures in pairing-free groups. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 363–395. Springer (2024)
54. Kondi, Y., Shelat, A.: Improved straight-line extraction in the random oracle model with applications to signature aggregation. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 279–309. Springer (2022)
55. Leroux, A., Roméas, M.: Updatable encryption from group actions. In: International Conference on Post-Quantum Cryptography. pp. 20–53. Springer (2024)
56. Lindell, Y.: Simple three-round multiparty schnorr signing with full simulatability. IACR Communications in Cryptology (Apr 2024)
57. Lysyanskaya, A., Rosenbloom, L.N.: Universally composable σ-protocols in the global random-oracle model. In: Theory of Cryptography Conference. pp. 203–233. Springer (2022)
58. Lyubashevsky, V.: Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In: International conference on the theory and application of cryptology and information security. pp. 598–616. Springer (2009)
59. Meers, J., Riepel, D.: Cca secure updatable encryption from non-mappable group actions. In: International Conference on Post-Quantum Cryptography. pp. 137–169. Springer (2024)
60. Mendo, L.: Estimation of a probability with guaranteed normalized mean absolute error. IEEE Communications Letters **13**(11), 817–819 (Nov 2009)
61. NIST: Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process (2023), uRL: https://csrc.nist.gov/projects/pqc-dig-sig/standardization/call-for-proposals
62. Pass, R.: On deniability in the common reference string and random oracle model. In: Advances in Cryptology-CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings 23. pp. 316–337. Springer (2003)
63. Pointcheval, D., Stern, J.: Provably secure blind signature schemes. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 252–265. Springer (1996)

64. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: International conference on the theory and applications of cryptographic techniques. pp. 387–398. Springer (1996)
65. Prasad, G.: Sharper variance upper bound for unbiased estimation in inverse sampling. Trabajos de Estadistica y de Investigacion Operativa **33**(3), 130–132 (Oct 1982)
66. Rotem, L., Tessaro, S.: Straight-line knowledge extraction for multi-round protocols. In: Annual International Cryptology Conference. pp. 95–127. Springer (2025)
67. Scheurer, T.: Universally composable verifiable random oracles. Ph.D. thesis, Karlsruher Institut für Technologie (KIT) (2022)
68. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of cryptology **4**, 161–174 (1991)
69. Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 1–16. Springer (1998)
70. Stern, J.: A new identification scheme based on syndrome decoding. In: Annual International Cryptology Conference. pp. 13–21. Springer (1993)
71. Tang, G., Duong, D.H., Joux, A., Plantard, T., Qiao, Y., Susilo, W.: Practical post-quantum signature schemes from isomorphism problems of trilinear forms. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 582–612. Springer (2022)
72. Unruh, D.: Non-interactive zero-knowledge proofs in the quantum random oracle model. In: Advances in Cryptology-EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II 34. pp. 755–784. Springer (2015)

A Extended Preliminaries

Notation. Let A be an event parametrized by the security parameter λ and P_λ its probability. We say that A occurs with negligible probability if, for every polynomial $p \in \mathbb{N}[x]$, $P_\lambda < \frac{1}{p(\lambda)}$ for infinitely-many values of λ in $\mathbb{N}$. Conversely, A occurs with overwhelming probability if $1 - P_\lambda$ is negligible.

We denote by $\mathbb{N}^*$ the set of non-zero natural numbers. Given $M \in \mathbb{N}$ and $N \in \mathbb{N}^*, N > M$, we denote by $[N]$ the set $\{1, \ldots, N\}$, and with $[M, N]$ the set $\{M, M+1, \ldots, N\}$.

Where not otherwise specified, each algorithm is probabilistic polynomial-time (PPT). For a deterministic algorithm A, we write $y \leftarrow \mathsf{A}(x)$ to denote the assignment to y of the output of A on input x. If A is probabilistic, we write $y \leftarrow\!\!\$\ \mathsf{A}(x)$. To make the algorithm's use of random coins r explicit, we write $\mathsf{A}(x; r)$. In a pseudocode, each variable assignment is done by either deterministic assignment ($\leftarrow$) or probabilistic assignment ($\leftarrow\!\!\$$), while the symbol $=$ is reserved for equality testing. Furthermore, we use the symbol $\perp$ to denote a failure, e.g. $\perp \leftarrow \mathsf{A}(x)$. For an algorithm A and an arbitrary function F, we write A^{F} to denote the execution of A with access to F. For a set S, we write $s \leftarrow\!\!\$\ S$ to denote sampling from the uniform distribution over S.

When two quantities a and b differ only of a negligible value in the security parameter λ, we write $a \approx b$.

Game 1: Zero-Knowledge

$\mathsf{Exp}^{\text{ZK-REAL}}_{(\mathsf{Prove},\mathsf{Verify}),\mathsf{H}}$:
1: $(x, w, \mathsf{st}_{\mathcal{D}}) \leftarrow\!\$\ \mathcal{D}_0^{\mathsf{H}}(\lambda)$
2: **if** $(x, w) \in \mathcal{R}$ **then**
3: $\quad \pi \leftarrow\!\$\ \mathsf{Prove}^{\mathsf{H}}(x, w)$
4: **else**
5: $\quad \pi \leftarrow \perp$
6: **return** $\mathcal{D}_1^{\mathsf{H}}(\pi, \mathsf{st}_{\mathcal{D}})$

$\mathsf{Exp}^{\text{ZK-IDEAL}}_{(\mathsf{Prove},\mathsf{Verify}),\mathsf{Sim}}$:
1: $(\mathsf{H}^{(0)}, \mathsf{st}_S) \leftarrow\!\$\ \mathsf{Sim}_0(\lambda)$
2: $(x, w, \mathsf{st}_{\mathcal{D}}) \leftarrow\!\$\ \mathcal{D}_0^{\mathsf{H}^{(0)}}(\lambda)$
3: **if** $(x, w) \in \mathcal{R}$ **then**
4: $\quad (\mathsf{H}^{(1)}, \pi) \leftarrow\!\$\ \mathsf{Sim}_1(\mathsf{st}_S, x, 1)$
5: **else**
6: $\quad (\mathsf{H}^{(1)}, \pi = \perp) \leftarrow\!\$\ \mathsf{Sim}_1(\mathsf{st}_S, x, 0)$
7: **return** $\mathcal{D}_1^{\mathsf{H}^{(1)}}(\pi, \mathsf{st}_{\mathcal{D}})$

Let $\{\mathcal{R}_\lambda \subseteq X_\lambda \times W_\lambda\}_{\lambda\in\mathbb{N}}$ be a set of NP binary relations, where X_λ, W_λ are finite subsets of $\{0,1\}^*$ for each value of λ. Elements of X_λ are called statements; if $(x, w) \in \mathcal{R}_\lambda$, then w is said to be a witness for x. To ease the notation, we will use $\mathcal{R} \subset X \times W$ to denote both a family of binary relations $\{R_\lambda\}_{\lambda\in\mathbb{N}}$ and any of its elements.

A.1 Straight-Line Extractable NIKPs and Fischlin Transform

Definition 8 (NIZKP with straight-line extractor [40])**.** *An straight-line extractable NIZKP for a binary relation $\mathcal{R}$, with completeness error $\epsilon_c^{\mathsf{Prove}}$ and soundness error $\epsilon_s^{\mathsf{Ext}}$, is a pair of oracle-calling PPT algorithms* $(\mathsf{Prove}, \mathsf{Verify})$ *satisfying the following properties:*

1. Completeness*: Let* H *be a random oracle.* Prove*, on input a statement-witness pair $(x, w) \in \mathcal{R}$, generates an invalid proof $\pi = \perp$ for x with probability $\epsilon_c^{\mathsf{Prove}}$, i.e.*
$$\epsilon_c^{\mathsf{Prove}} = \Pr[\perp \leftarrow\!\$\ \mathsf{Prove}(x, w) \mid (x, w) \in \mathcal{R}].$$
2. Zero-Knowledge*: Let* H *be a random oracle. For any PPT distinguisher $\mathcal{D} = (\mathcal{D}_0, \mathcal{D}_1)$, there exists a simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ such that the quantity*
$$|\Pr\left[\mathsf{Exp}^{\text{ZK-REAL}}_{(\mathsf{Prove},\mathsf{Verify}),\mathsf{H}}(\mathcal{D}) = 1\right] - \Pr\left[\mathsf{Exp}^{\text{ZK-IDEAL}}_{(\mathsf{Prove},\mathsf{Verify}),\mathsf{Sim}}(\mathcal{D}) = 1\right]|$$
is negligible in λ, where $\mathsf{Exp}^{\text{ZK-REAL}}_{(\mathsf{Prove},\mathsf{Verify}),\mathsf{H}}(\mathcal{D})$ and $\mathsf{Exp}^{\text{ZK-IDEAL}}_{(\mathsf{Prove},\mathsf{Verify}),\mathsf{Sim}}(\mathcal{D})$ are defined in Game 1.
3. Straight-Line Extractor*: There exists a PPT algorithm* Ext *— called straight-line extractor — such that the following holds for any adversary $\mathcal{A}$. Let* H *be a random oracle and $(x, \pi, Q_H(\mathcal{A})) \leftarrow\!\$\ \mathcal{A}^{\mathsf{H}}(\lambda)$, where $Q_H(\mathcal{A})$ is the set of queries made by $\mathcal{A}$ to* H*, together with its answers. For $w \leftarrow\!\$\ \mathsf{Ext}(x, \pi, Q_H(\mathcal{A}))$, the probability $\epsilon_s^{\mathsf{Ext}}$ — called soundness error — that $(x, w) \notin \mathcal{R}$ given $\mathsf{Verify}^{\mathsf{H}}(x, \pi) = \mathsf{accept}$, i.e.*
$$\epsilon_s^{\mathsf{Ext}} = \Pr\left[(x, \mathsf{Ext}(x, \pi, Q_H(\mathcal{A}))) \notin \mathcal{R} \mid (x, \pi, Q_H(\mathcal{A})) \leftarrow \mathcal{A}^{\mathsf{H}}(\lambda) \wedge \mathsf{Verify}^{\mathsf{H}}(x, \pi) = \mathsf{accept}\right],$$
is negligible in λ.

A.2 Sigma Protocols

Loosely speaking, a Sigma protocol for a binary relation $\mathcal{R} \subset X \times W$ is a 3-step interactive protocol between a prover and a verifier. The goal of the prover is proving knowledge of a witness $w \in W$ for a public statement $x \in X$ without revealing any information about w.

Definition 9 (Sigma Protocol [29])**.** *Let $\mathcal{R} \subseteq X \times W$ be a binary relation. A Sigma protocol Σ for $\mathcal{R}$ consists of PPT algorithms $(\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$, where V_2 is deterministic and we assume $\mathsf{P}_1, \mathsf{P}_2$ share states. Let $\mathit{ChSpace}$ be the (finite) challenge space. Then, the interaction between a prover and a verifier by means of Σ has the following three-move flow. The prover, on input $(x, w) \in \mathcal{R}$, runs $f \leftarrow\!\!\$\ \mathsf{P}_1(x, w)$ to compute a message f, called first message, which is sent to the verifier. The verifier runs $\mathsf{ch} \leftarrow\!\!\$\ \mathsf{V}_1(1^\lambda)$ to obtain a random challenge ch from $\mathit{ChSpace}$, and sends it to the prover. Finally, the prover computes a response $z \leftarrow\!\!\$\ \mathsf{P}_2(x, w, f, \mathsf{ch})$ and sends it to the verifier, who outputs $\mathsf{accept/reject} \leftarrow \mathsf{V}_2(x, f, \mathsf{ch}, z)$.*

We require Σ to satisfy the following properties.

1. Completeness: *We say that Σ is complete if, for all $(x, w) \in \mathcal{R}$, when prover and verifier follow the protocol, the verifier always accepts.*
2. 2-Special Soundness: *We say that Σ is 2-special sound if there exists a PPT algorithm Ext (called witness extractor) such that, given a statement $x \in X$ and two accepting transcripts (f, ch, z) and (f, ch', z') relative to x, with $\mathsf{ch} \neq \mathsf{ch}'$, Ext always outputs $w \in W$ such that $(x, w) \in \mathcal{R}$ (i.e., w is a witness for x). 2-special soundness implies that, if $\mathcal{R}$ is a hard relation, a dishonest prover can convince the verifier to accept with probability at most $1/|\mathit{ChSpace}|$, which is called soundness error.*
3. Honest-Verifier Zero-Knowledge: *Σ is said to be honest-verifier zero-knowledge (HVZK) if there exists a PPT algorithm Sim (called simulator) which, on input $(x, \mathsf{ch}) \in X \times \mathit{ChSpace}$, outputs a pair (f, z) such that (f, ch, z) is an accepting transcript for x. In addition, the output distribution of Sim on input (x, ch) is equal to the distribution of those outputs generated via an honest execution of Σ with the prover knowing a witness w for x and the verifier conditioned on using ch as the challenge.*

To securely apply the Fiat-Shamir transform to a Sigma protocol Σ, to construct a NIZKP or a signature scheme [68,19,15,71,31,70], Σ is usually required to satisfy some additional properties beyond the completeness, honest-verifier zero-knowledge and special soundness.

Definition 10 (Effective Sigma protocol[18])**.** *We say that a Sigma protocol $\Sigma = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ for a binary relation $\mathcal{R}$ is effective if it satisfies the following properties.*

[18] In literature these Sigma protocols are also referred to as Fiat-Shamir proofs of knowledge [40].

1. First-Message Entropy: *Being λ the security parameter, the min-entropy of $f \leftarrow\!\$\ \mathsf{P}_1(x, w)$ is super-logarithmic in λ, for $(x, w) \in \mathcal{R}$.*
2. Unique Responses: *On input the security parameter λ, any PPT algorithm $\mathcal{A}$ produces a tuple $(x, f, \mathsf{ch}, z, z') \leftarrow\!\$\ \mathcal{A}(\lambda)$ such that $\mathsf{V}_2(x, f, \mathsf{ch}, z) = \mathsf{V}_2(x, f, \mathsf{ch}, z') = \mathsf{accept}$ and $z \neq z'$, with no more than negligible probability in λ.*

Another property that characterize the Sigma protocols that we are interested into is the *first-message retrievability*.

Definition 11 (First-Message-Retrievable Sigma Protocol)**.** *We say that a Sigma protocol $\Sigma = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ for a binary relation $\mathcal{R} \subset X \times W$ is first-message retrievable if, for any statement $x \in X$, challenge ch and response z, there exists a unique first message f such that (f, ch, z) is a valid transcript for x and a PPT algorithm F such that $f \leftarrow\!\$\ F(x, \mathsf{ch}, z)$.*

Parallel repetition of Sigma protocols [29]. Reducing the soundness error of a Sigma protocol Σ is sometimes necessary to design cryptographic primitives using Σ as a building block, as for example digital signatures and NIZKPs using the Fiat-Shamir transform [64,1]. A common solution to decrease the soundness error of a Sigma protocol is to repeat the protocol in parallel multiple times.

Definition 12 (Parallel Repetition)**.** *Let $\Sigma = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ be a Sigma protocol for a binary relation $\mathcal{R}$ with challenge space $\mathsf{ChSpace}$. Given $L \in \mathbb{N}^*$, the L-fold* parallel repetition *of Σ is the Sigma protocol $\Sigma^L = (\mathsf{P}^L = (\mathsf{P}_1^L, \mathsf{P}_2^L), \mathsf{V}^L = (\mathsf{V}_1^L, \mathsf{V}_2^L))$ for the same relation $\mathcal{R}$, where P^L (resp. V^L) are obtained by executing $\mathsf{P}_1, \mathsf{P}_2$ (resp. $\mathsf{V}_1, \mathsf{V}_2$) each L times* in parallel. *Therefore, a transcript for Σ^L is of the form $\big((f_i)_{i\in[L]}, (\mathsf{ch}_i)_{i\in[L]}, (z_i)_{i\in[L]}\big)$, where each $(f_i, \mathsf{ch}_i, z_i)$ is a transcript for Σ. As a result, the challenge space of Σ^L is $\mathsf{ChSpace}^L$.*

We notice that Σ^L has soundness error equal to $1/|\mathsf{ChSpace}|^L$.

A.3 Examples of Seed-Friendly Sigma Protocols

Examples of seed friendly Sigma protocols are the Schnorr Sigma protocol (with challenge space restricted to $\mathsf{ChSpace} \subset \mathbb{Z}_p, |\mathsf{ChSpace}| \leq \mathsf{poly}(\lambda)$) and the Sigma protocol for graph isomorphism [45].

Let us recall how these Sigma protocols are defined.

Schnorr Let G be a prime order p group, and (x, w) a statement-witness pair with $x = (h, g), h, g \in G$, and $w \in \mathbb{Z}_p$ with $h = g^w$. The Schnorr Sigma protocol defines the following algorithms: P_1 samples $r \leftarrow\!\$\ \mathbb{Z}_p$ and to computes $f = g^r$; V_1 samples a random challenge $\mathsf{ch} \in \mathsf{ChSpace} \subseteq \mathbb{Z}_p$; P_2, given $r, \mathsf{ch} \in \mathsf{ChSpace}, w$ in input, instructs the prover to compute the response $z = f + \mathsf{ch}w$; V_2 checks that $f(h)^{\mathsf{ch}} = g^z$.

GMW Let $\mathbf{G}$ be the set of isomorphic graphs, $\mathbf{I}$ the group of graphs isomorphisms, and (x, w) a statement-witness pair, with $x = (G_1, G_0), G_0, G_1 \in \mathbf{G}$

Transformation 5: Fiat-Shamir

Let $\Sigma = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ be an effective Sigma protocol.

$\mathsf{Prove}^{\mathsf{H}}_{\mathsf{FS}}(x, w)$:
1: $f \leftarrow\!\$\ \mathsf{P}_1(x, w)$
2: $\mathsf{ch} \leftarrow \mathsf{H}(x, f)$
3: $z \leftarrow\!\$\ \mathsf{P}_2(x, w, f, \mathsf{ch})$
4: **return** $\pi \leftarrow (f, z)$

$\mathsf{Verify}^{\mathsf{H}}_{\mathsf{FS}}(x, \pi = (f, z))$:
1: $\mathsf{ch} \leftarrow \mathsf{H}(x, f)$
2: **return** $\mathsf{V}_2(x, f, \mathsf{ch}, z)$

and $w \in \mathbf{I}$ with $G_1 = w \star G_0$. The Sigma protocol for graph isomorphism defines the following algorithms: P_1 samples $r \leftarrow\!\$\ \mathbf{I}$, and computes $f \leftarrow r \star G_1$; V_1 samples $\mathsf{ch} \leftarrow\!\$\ \{0,1\}$; P_2, given in input r, ch, w returns $z = w^{-\mathsf{ch}} \cdot r$; V_2 checks that $f = z \star G_{\mathsf{ch}}$.

In both cases the special challenge is $\tilde{\mathsf{ch}} = 0$ and the public function is $Z(x, r) = r$.

A.4 Fixed-Weight Technique for Fiat-Shamir

The Fiat-Shamir transform [39,64] is the most common (non-straight-line extractable) compiler used to turn Sigma protocols in NIZKP because of its simplicity and efficiency. Intuitively, the idea of the transform is to replace the challenge computation from the verifier to the output of a random oracle, taking as input the first message produced by the prover.

Definition 13. *Let λ be a security parameter and let $\Sigma = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ be an effective Sigma protocol for a binary relation $\mathcal{R}$ with challenge space $\mathit{ChSpace}$ and soundness error $\mathsf{negl}(\lambda)$. Let $\mathsf{H}\colon \{0,1\}^* \to \mathit{ChSpace}$ be a random oracle. The Fiat-Shamir transform applied on Σ produces a NIZKP $(\mathsf{Prove}_{\mathsf{FS}}, \mathsf{Verify}_{\mathsf{FS}})$ as defined in Transformation 5.*

When the Fiat-Shamir transform needs to be applied to a Sigma protocol Σ with challenge space with small cardinality ℓ, Σ must be repeated in parallel $L(\lambda) = \lambda / \log(\ell)$ times so to generate the Sigma protocol $\Sigma^{L(\lambda)}$ with a challenge space of sufficiently large cardinality 2^λ. It is well-known that from Sigma protocols with high min-entropy and negligible soundness error, the Fiat-Shamir transform can be employed to obtain a secure digital signature in the random oracle model [1].

Fixed-Wight Technique When the Sigma protocol is seed-friendly (Definition 1), it is possible to apply the fixed-weight techniques used for some post quantum signature schemes based on cryptographic group actions [15,25,71]. Since the Fiat-Shamir transform produces NIZKPs whose proofs resemble a transcript of an honest execution of the compiled Sigma protocol, this technique directly applies to the Sigma protocol that later will be compiled using the Fiat-Shamir transform.

The Fiat-Shamir transform requires the size of the challenge space of the compiled Sigma protocol to be exponential in the security parameter. Therefore,

a seed-friendly Sigma protocol Σ must be repeated in parallel to enlarge the challenge space, and reduce the soundness error. Instead of using the standard parallel repetition of Σ, a modified repetition technique where challenges are sampled from a non-uniform distribution is employed. Specifically, the number of challenges equal to $\tilde{\mathsf{ch}}$ in a proof of the fixed-weight repetition is fixed, allowing responses for these challenges to be compressed using short seeds.

More precisely, consider the L-fold parallel repetition Σ^L where challenges $\mathsf{ch} = (\mathsf{ch}_1, \ldots, \mathsf{ch}_L) \in \mathsf{ChSpace}^L$ are restricted to have exactly ρ positions where $\mathsf{ch}_i \neq \tilde{\mathsf{ch}}$. We call ρ the *weight* of the challenge vector, and this technique the (L, ρ)-fixed-weight repetition. Under the hypothesis that the challenge space $\mathsf{ChSpace}$ of Σ is identified with $\{0, 1, \ldots, |\mathsf{ChSpace}| - 1\}$ and $\tilde{\mathsf{ch}} = 0$ (see Remark 1), the weight of a challenge vector ch with respect to $\tilde{\mathsf{ch}}$ is simply its Hamming weight $\mathsf{wt}(\mathsf{ch})$.

Definition 14 (Fixed-weight Repetition)**.** *Let $\Sigma = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ be a* seed-friendly *Sigma protocol for a binary relation $\mathcal{R}$ with challenge space $\mathsf{ChSpace}$. Given $L, \rho \in \mathbb{N}^*, \rho \leq L$, the (L, ρ)-fixed-weight repetition of Σ, which we denote with $\Sigma^{L,\rho}$, is the L-fold parallel repetition with challenge space $\mathsf{ChSpace}^{L,\rho} = \{c \in \mathsf{ChSpace}^L \mid \mathsf{wt}(c) = \rho\}$. Therefore, a transcript for $\Sigma^{L,\rho}$ is of the form $\big((f_i)_{i\in[L]}, (\mathsf{ch}_i)_{i\in[L]}, (z_i)_{i\in[L]}\big)$, where for exactly $L - \rho$ repetitions, i.e. for $L - \rho$ indices, $(f_i, \mathsf{ch}_i, z_i)$ is a transcript for Σ where $\mathsf{ch}_i = 0$ and z_i is a λ bit seed.*

The soundness error of the fixed-weight repetition $\Sigma^{L,\rho}$ is equal to:

$$\frac{1}{|\mathsf{ChSpace}^{L,\rho}|} = \binom{L}{\rho}^{-1} \frac{1}{\big(|\mathsf{ChSpace}| - 1\big)^{\rho}}.$$

Notice that this technique presents a trade-off. In fact, while it reduces the proof size by allowing $L - \rho$ responses to be encoded as short seeds, it increases the computational overhead since the prover, compared to the standard parallel repetition, must compute more first messages in order to achieve the same soundness error. This is particularly noticeable for very unbalanced parameters (e.g. $\rho \ll L/2$), where L significantly grows to make the soundness error suitably small.

The presented technique is particularly effective when coupled with the Fiat-Shamir transform, which is designed to simulate an honest execution of the underlying Sigma protocol, inheriting any optimizations. However, its application to other transforms, like the Fischlin's one, requires a more careful analysis. For an extensive discussion of this, we refer the reader to Section 3.

A.5 Cryptographic Group Actions

Definition 15 (Group Action)**.** *A group $(G, \cdot, e)$, with e the identity element, is said to act on a set X if there exists a map*

$$\begin{aligned} \star \colon G \times X &\longrightarrow X \\ (g, x) &\longmapsto g \star x \end{aligned}$$

such that $e \star x = x$ *for every* $x \in X$ *and* $g \star (h \star x) = (g \cdot h) \star x$ *for every* $x \in X$ *and* $g, h \in G$*. In this case, we say that the triple* $(G, X, \star)$ *is a group action.*

To build cryptographic schemes, only group actions $(G, X, \star)$ for which G and X are both finite are considered. In addition, it must be possible to efficiently perform some operations, such as: (1) computing the group operation and the inverse of a group element; (2) performing (uniform) random sampling in X and G, (3) deciding equality and validity of a representation of elements in X; (4) computing the group action $\star$. Group actions for which these operations are efficient are called *effective group actions* [2].

Besides efficiency requirements, group actions for cryptographic applications must satisfy security properties. In particular, a group action must be based on one-way functions, or equivalently-hard problems. An effective group action that satisfies this further requirement is called *cryptographic group action*. A (typically hard) problem for group actions is the Group Action Inversion Problem (GAIP), defined below.

Definition 16 (Group Action Inversion Problem (GAIP))**.** *The GAIP on a group action* $(G, X, \star)$ *is defined as follows. Let* $x_0 \in X$*. Given as input a uniformly-random element* $x_1 \in \{g \star x_0 | g \in G\}$*, output a group element* $g \in G$ *such that* $x_1 = g \star x_0$*.*

Another well-known problem is the Stabilizer Computation Problem (SCP), defined as follows.

Definition 17 (Stabilizer Computation Problem (SCP))**.** *On input a uniformly-random* $x_0 \in X$*, the SCP over a group action* $(G, X, \star)$ *consists in finding an element* $g \in G \setminus \{e\}$ *such that* $g \star x_0 = x_0$*.*

Examples of effective group actions for which these problems are believed to be hard are based on isogenies [21][19], tensors [48,71] and linear codes [25,15].

About the seed-friendlyness of $\Sigma_\star$ It can be easily proved that $\Sigma_\star$ is complete, 2-special sound and honest-verifier zero knowledge, as per Definition 9. The simulator Sim, used to prove the HVZK property, takes in input a statement (x_0, x_1) and a challenge $\mathsf{ch} \in \{0, 1\}$, samples uniformly at random a group element $z \in G$, computes the first message $f = z \star x_{\mathsf{ch}}$ and returns the transcript (f, ch, z). The witness extractor Ext considered to prove the 2-special-soundness property takes two transcripts $(f, 0, z_0)$ and $(f, 1, z_1)$ for the same statement (x_0, x_1), and outputs a witness $w = z_1^{-1} z_0$.

When we instantiate the Sigma protocol $\Sigma_\star$ using a group action $(G, X, \star)$ for which SCP is hard, then $\Sigma_\star$ has unique responses. In fact, given a first message f and a challenge ch, it is hard for any prover to compute two distinct valid responses. In this case, if the group G and the set X are sufficiently large, then the min-entropy of the first message of the Sigma protocol is super-logarithmic in

[19] For the original CSIDH group action, only the action of group elements with a specific form can be effectively computed.

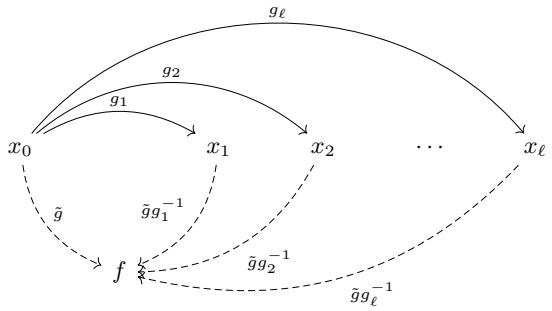


Fig. 2: Graphical representation of $\Sigma_\star$ when ℓ public keys (i.e. statements) are used.

λ, i.e. $\Sigma_\star$ is an effective Sigma protocol according to Definition 10. In addition, the Sigma protocol $\Sigma_\star$ is seed-friendly because it is first-message retrievable, has constant challenge space and the $\tilde{\mathsf{ch}} = 0$ is the special challenge (see Definition 1).

A.6 Multiple Public Keys Optimization

Given a Sigma protocol $\Sigma_\star$ for a group action $(G, X, \star)$, the idea behind the technique is to consider a list $(x_1, \ldots, x_\ell) \in X^\ell$ of statements obtained from secret group elements $g_1, \ldots, g_\ell$ by computing $x_i = g_i \star x_0$ for some $x_0 \in X$. In this way, given a first message $f \leftarrow \tilde{g} \star x_0$, the possible challenges are given by the union of all the paths $\tilde{g}g_i^{-1}$ between x_i and the first message, together with the path $\tilde{g}$ which sends x_0 to f. Therefore, the challenge space then grows from $\{0, 1\}$ to a larger set $\{0, \ldots, \ell\}$. A schematic representation of the optimization is provided in Figure 2.

Notice that, differently from the previously-presented optimizations, the resulting protocol is for a relation different from that of $\Sigma_\star$. As a consequence, for its security a stronger hardness assumption is necessary, relative to the following problem.

Definition 18 (Multiple Group Action Inverse Problem (MGAIP))**.** *Given a group action $(G, X, \star)$, the Multiple Group Action Inverse Problem (MGAIP) takes as input $\ell + 1$ random set elements $x_0, \ldots, x_\ell$ ($\ell \geq 1$) in the orbit $G \star x_0$, and asks to find $g \in G$ such that $x_i = g \star x_j$, for some $i \neq j$.*

A.7 Fischlin Transform

In the following, we formally define the Fischlin transform with the inversion predicate of Definition 2.

Definition 19 (Fischlin Transform)**.** *Let λ be a security parameter and let $\Sigma = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ be an effective Sigma protocol for a binary relation $\mathcal{R}$ with challenge space $\mathsf{ChSpace}$. Let $b \in \mathbb{Q}, r \in \mathbb{N}^*$, and let $\mathsf{H} \colon \{0,1\}^* \to \{0,1\}^\lambda$*

Transformation 6: Fischlin$[b, r]$

Let $\Sigma = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2))$ be an effective Sigma protocol.

$\mathsf{Prove}^{\mathsf{H}}_{\mathtt{Fi}}(x, w)$:
1: **for** $i \leftarrow 1, \ldots, r$ **do**
2: $\quad f_i \leftarrow\!\!\$\ \mathsf{P}_1(x, w)$
3: $\overline{f} \leftarrow (f_1, \ldots, f_r)$
4: **for** $i \leftarrow 1, \ldots, r$ **do**
5: $\quad \mathsf{ch}_i \leftarrow \bot$
6: $\quad$ **for** $\mathsf{ch}_j \in \mathsf{ChSpace}$ **do**
7: $\quad\quad z_i \leftarrow\!\!\$\ \mathsf{P}_2(x, w, f_i, \mathsf{ch}_j)$
8: $\quad\quad$ **if** $\mathsf{H}(x, \overline{f}, i, \mathsf{ch}_j, z_i) < \lfloor 2^{\lambda - b} \rfloor$ **then**
9: $\quad\quad\quad \mathsf{ch}_i \leftarrow \mathsf{ch}_j$
10: $\quad\quad\quad$ **break**
11: $\quad$ **if** $\mathsf{ch}_i = \bot$ **then return** $\bot$
12: **return** $\pi \leftarrow (\{f_i\}_{i\in[r]}, \{\mathsf{ch}_i\}_{i\in[r]}, \{z_i\}_{i\in[r]})$

$\mathsf{Verify}^{\mathsf{H}}_{\mathtt{Fi}}(x, \pi = (\{f_i\}_{i\in[r]}, \{\mathsf{ch}_i\}_{i\in[r]}, \{z_i\}_{i\in[r]}))$:
1: $\overline{f} \leftarrow (f_1, \ldots, f_r)$
2: **for** $i \leftarrow 1, \ldots, r$ **do**
3: $\quad$ **if** $\mathsf{V}_2(x, f_i, \mathsf{ch}_i, z_i) = \mathsf{reject}$ **then**
4: $\quad\quad$ **return** reject
5: $\quad$ **if** $\mathsf{H}(x, \overline{f}, i, \mathsf{ch}_i, z_i) \geq \lceil 2^{\lambda - b} \rceil$ **then**
6: $\quad\quad$ **return** reject
7: **return** accept

be a random oracle. The Fischlin transform, parametrized by (b, r)*, applied to* Σ *produces a NIZKP* $(\mathsf{Prove}_{\mathtt{Fi}}, \mathsf{Verify}_{\mathtt{Fi}})$*, as detailed in Transformation 6.*

A.8 LESS Signature Scheme

Group Action. LESS is constructed from a group action $\star : \mathrm{Mon}_q(n, k) \times X \to X$, where X represents the set of $[n, k]$ linear codes over the finite field $\mathbb{F}_q$ and $\mathrm{Mon}_q(n, k)$ is the group of monomial transformations. The underlying Sigma protocol $\Sigma_\star$ is seed-friendly with special challenge $\tilde{\mathsf{ch}} = 0$.

For security parameter $\lambda = 128$, LESS uses parameters $q = 127$, $n = 252$, and $k = 126$. Following various optimizations and the use of canonical forms [27], the size of a monomial transformation can be minimized to n bits, nearly reaching the optimal theoretical lower bound of 2λ bits. Official parameterizations involving the (L, ρ)-fixed-weight optimization with seed trees and ℓ public-keys are shown in Table 1.

Performance Baseline. In the last column of Table 1, we include the performance of LESS (using the Fiat-Shamir transform) as a reference point for the efficiency comparison we will make with the newly-proposed generic transforms. However, this comparison has inherent limitations. In fact, contrarily to our transforms, the Fiat-Shamir transform incurs a security loss due to rewinding in its security reduction, which is typically ignored in practical parameter selection [28]. Additionally, since our transforms target straight-line extractability while Fiat-Shamir does not, the security models differ fundamentally. Straight-line extractable transforms typically incur additional computational overhead and larger proof sizes compared to their rewinding-based counterparts.

Generalizability. LESS achieves near-optimal compactness for its group elements, with bit-length 2λ. Sigma protocols based on group actions with less compact representations would see proportionally larger improvements when applying our transforms, since the proof size would benefit further from the use of *highly-unbalanced* fixed-weight distributions.

B Missing Proofs

B.1 Proof of Theorem 1

By construction, an honest execution of $\mathsf{Prove}_{\blacktriangle}$ computes at most ℓ random-oracle queries for each of the L first messages of the underlying Sigma protocol Σ, so that $Q^{\mathsf{Prove}_{\blacktriangle}} \leq L \cdot \ell$.

Completeness error The probability that an index $i \in [L]$ is a target component, i.e. there exists $\mathsf{ch}_i \in S$ s.t. $\mathsf{H}(d, i, \mathsf{ch}_i, z_i) < \lfloor 2^{\lambda - b} \rfloor$ is given by $p \approx 1 - (1 - 2^{-b})^{\ell}$, since ℓ different challenges can be tested to obtain a transcript which satisfies the inversion predicate[20]. On the other hand, the completeness error $\epsilon_c^{\mathsf{Prove}_{\blacktriangle}}$ (i.e. the probability that the algorithm $\mathsf{Prove}_{\blacktriangle}$ fails in producing a proof and outputs $\perp$) measures the probability that the number of obtained target components is less than ρ over all the L components. Formally,

$$\epsilon_c^{\mathsf{Prove}_{\blacktriangle}} = \sum_{i=0}^{\rho-1} \binom{L}{i} p^i (1-p)^{L-i}.$$

Zero-Knowledge We provide the description of a simulator $(\mathsf{Sim}_{\blacktriangle,0}, \mathsf{Sim}_{\blacktriangle,1})$ that makes the two experiments in Definition 8 (Zero-Knowledge property) indistinguishable for any distinguisher $(\mathcal{D}_0, \mathcal{D}_1)$. The distinguisher interacts with one of the two experiments, where one captures the interaction with a real prover and a real random oracle while the second captures the interaction with the simulator $(\mathsf{Sim}_{\blacktriangle,0}, \mathsf{Sim}_{\blacktriangle,1})$, which we now define.

When $\mathcal{D}_0$ interacts with the simulator, $\mathsf{Sim}_{\blacktriangle,0}$ defines $\mathsf{H}^{(0)}$ as an empty hash table which it programs to simulate a random oracle. In particular, $\mathsf{Sim}_{\blacktriangle,0}$ gives $\mathcal{D}_0$ access to $\mathsf{H}^{(0)}$, i.e. $\mathcal{D}_0$ can send $\mathsf{Sim}_{\blacktriangle,0}$ random-oracle queries for some input m. If m was not previously queried, $\mathsf{Sim}_{\blacktriangle,0}$ programs the random oracle $\mathsf{H}^{(0)}$ by sampling uniformly at random $d_m \leftarrow\!\!\$\ \{0,1\}^{\lambda}$ and adding to the hash table the pair (m, d_m). If m was previously queried, $\mathsf{Sim}_{\blacktriangle,0}$ returns d_m.

$\mathcal{D}_0$ eventually outputs a triple $(x, w, \mathsf{st}_{\mathcal{D}})$: when $(x, w) \notin \mathcal{R}$, $\mathsf{Sim}_{\blacktriangle,1}$ is given in input $(\mathsf{st}_S, x, 0)$, where st_S is the state information received from $\mathsf{Sim}_{\blacktriangle,0}$, and returns $(\mathsf{H}^{(1)}, \perp)$ to the distinguisher. Otherwise, if $(x, w) \in \mathcal{R}$, $\mathsf{Sim}_{\blacktriangle,1}$ simulates the generation of a proof (and the random oracle $\mathsf{H}^{(1)}$), taking as input only the triple $(\mathsf{st}_S, x, 1)$, as detailed below:

1. $\mathsf{Sim}_{\blacktriangle,1}$ denotes the elements of S as $\{c_j\}_{j \in [\ell]}$ and initializes and empty set $\mathcal{T} = \emptyset$;
2. for $i \in [L]$, and while $|\mathcal{T}| < \rho$, $\mathsf{Sim}_{\blacktriangle,1}$ performs the following operations:
   - $\forall j \in [\ell]$, $\mathsf{Sim}_{\blacktriangle,1}$ samples $s_{i,j} \leftarrow\!\!\$\ \{0,1\}^{\lambda}$ and defines the map

$$\tau_i : S \to \{0,1\}^{\lambda} \text{ mapping } c_j \mapsto s_{i,j};$$

[20] More precisely, the probability that a component is not target is $\geq (1 - 2^{-b})^{\ell}$.

- by interpreting the outputs of τ_i as integers in $[0, 2^\lambda - 1]$, $\mathsf{Sim}_{\blacktriangle,1}$ checks whether $\tau_i^{-1}([0, \lfloor 2^{\lambda-b} \rfloor]) \neq \emptyset$. If so, $\mathsf{Sim}_{\blacktriangle,1}$ denotes with c_k the smallest value (in increasing order[21]) in $\tau_i^{-1}([0, \lfloor 2^{\lambda-b} \rfloor]) \subseteq \{1, \ldots, \ell\}$, sets $\mathsf{ch}_i = c_k$, runs the simulator Sim of the HVZK property of the underlying Sigma protocol Σ obtaining $(f_i, z_i) \leftarrow\!\$\ \mathsf{Sim}(x, \mathsf{ch}_i)$, and adds i to $\mathcal{T}$;

3. If $|\mathcal{T}| < \rho$, $\mathsf{Sim}_{\blacktriangle,1}$ aborts and outputs $\perp$.
4. for all $i \in [L] \setminus \mathcal{T}$, $\mathsf{Sim}_{\blacktriangle,1}$ sets $\mathsf{ch}_i \leftarrow \tilde{\mathsf{ch}}$, then runs the simulator Sim of the HVZK property of the underlying Sigma protocol Σ obtaining $(f_i, z_i) \leftarrow\!\$\ \mathsf{Sim}(x, \mathsf{ch}_i)$, and sets $\pi \leftarrow (\{f_i\}_{i\in[L]}, \{\mathsf{ch}_i\}_{i\in[L]}, \{z_i\}_{i\in[L]})$;
5. $\mathsf{Sim}_{\blacktriangle,1}$ must program the random oracle $\mathsf{H}^{(1)}$ and to do so defines $\overline{f} = (f_1, \ldots, f_L)$, samples uniformly at random $d \leftarrow \{0,1\}^\lambda$ and sets $\mathsf{H}^{(1)}(x, \overline{f}) = d$ and for any transcript $(f_i, \mathsf{ch}_i, z_i)$ in π, with $\mathsf{ch}_i \neq \tilde{\mathsf{ch}}$, $\mathsf{Sim}_{\blacktriangle,1}$ sets $\mathsf{H}^{(1)}(d, i, \mathsf{ch}_i, z_i) = \tau_i(\mathsf{ch}_i) < \lfloor 2^{\lambda-b} \rfloor$.
   In order to make the simulation work, $\mathsf{Sim}_{\blacktriangle,1}$ must "implicitly program" $\mathsf{H}^{(1)}$ on inputs (d, i, ch, z) where $i \in \{i \in [L], i \leq \max(\mathcal{T})\}$, $\mathsf{ch} \in S, \mathsf{ch} \neq \mathsf{ch}_i$ and $\mathsf{V}_2(x, f_i, \mathsf{ch}, z) = \mathsf{accept}$ to $\tau_i(\mathsf{ch})$. By "implicitly program" we mean that $\mathsf{Sim}_{\blacktriangle,1}$ can not add the corresponding row to the hash table underlying $\mathsf{H}^{(1)}$ because it does not know the response z corresponding to f_i and $\mathsf{ch} \neq \mathsf{ch}_i$. However, the distinguisher can send valid queries associated to such transcripts using its knowledge of the witness. In such cases $\mathsf{Sim}_{\blacktriangle}$ checks whether $\mathsf{V}_2(x, f_i, \mathsf{ch}, z) = 1$ and, if so, returns the previously generated random digest specified by the function τ_i.
6. $\mathsf{Sim}_{\blacktriangle,1}$ outputs $(\mathsf{H}^{(1)}, \pi)$.

Remark 7. The random oracle $\mathsf{H}^{(1)}$ does not need to be programmed for inputs corresponding to transcripts with indices $i \in [L]$ where $i > \max(\mathcal{T})$, since the verification algorithm does not determines hash queries for these components, and they do not affect the distribution of real proofs. However, it is crucial that the simulator generates proofs by selecting ρ target components in sequential order and choosing the smallest challenge in S that satisfies the inversion predicate. This deterministic selection strategy is essential because the distinguisher, possessing the witness w, can verify whether the simulator followed this prescribed order by computing the missing transcripts and checking their hash values.

$\mathsf{Sim}_{\blacktriangle}$ produces a proof π which is indistinguishable from a real proof because the outputs of the hash function $\mathsf{H}^{(1)}$ are sampled uniformly at random. From these, the components for which $\mathsf{ch}_i \in S$ are selected as in a real-protocol execution. Also, the issue in the definition of the simulator highlighted in [67, A.5] (and remarked in [53]) is not a concern for our security proof because no random oracle input is arbitrarily assigned a digest $\leq \lfloor 2^{\lambda-b} \rfloor$.
The only case in which the simulation fails is when $\mathsf{Sim}_{\blacktriangle,1}$ has to program the hash table $\mathsf{H}^{(1)}$ adding the queries used to generate the proof π, mapping each

[21] We are working under the assumption that the challenge space of Σ is identified with $\{0, 1, \ldots, \ell\}$, as per Remark 1.

transcript $(d, i, \mathsf{ch}_i, z_i)$ to the associated digest $s_{i,j}$, but the input $(d, i, \mathsf{ch}_i, z_i)$ was already queried by $\mathcal{D}_0$. This happens with negligible probability because d has super-logarithmic min-entropy (it is chosen at random) and since the underlying Sigma protocol is effective, the min-entropy of the single first messages (and therefore of $\overline{f}$) is super-logarithmic. So, when $\mathsf{Sim}_{\blacktriangle,1}$ programs $d \leftarrow \mathsf{H}^{(1)}(x, \overline{f})$ the hash table of $\mathsf{H}^{(0)}$ is overwritten only with negligible probability since $\mathcal{D}_0$ can query $\mathsf{H}^{(0)}$ only on a polynomial number of inputs.

Straight-Line Extraction According to Transformation 1, the non-interactive protocol obtained in Definition 3 is straight-line extractable if there exists an algorithm $\mathsf{Ext}_{\blacktriangle}$ which, on input a valid proof $\pi = (\{f_i\}_{i\in[L]}, \{\mathsf{ch}\}_{i\in[L]}, \{z_i\}_{i\in[L]})$ for a statement x and the set of random-oracle queries $Q_H(\mathcal{A})$ made by the adversary $\mathcal{A}$ who produced the proof, outputs, except with negligible probability, a witness w for x. The extractor $\mathsf{Ext}_{\blacktriangle}$ we consider is defined as follows. It loops through the list $Q_H(A)$ and the transcripts $\{(f_i, \mathsf{ch}_i, z_i)_{i\in[L]}\}$ included in π, looking for two accepting transcripts $(f, \mathsf{ch}, z), (f, \mathsf{ch}', z')$ with $\mathsf{ch} \neq \mathsf{ch}'$. If the search is successful, $\mathsf{Ext}_{\blacktriangle}$ calls the witness extractor Ext of the underlying Sigma protocol Σ on the two triples, and outputs the extracted value. If the search is unsuccessful, $\mathsf{Ext}_{\blacktriangle}$ outputs $\perp$. Thus, in order to compute the failure probability of $\mathsf{Ext}_{\blacktriangle}$, it is sufficient to bound the probability that $\mathcal{A}$ does not provide $\mathsf{Ext}_{\blacktriangle}$ with two such transcripts, while the proof π is still correctly verified.

Without loss of generality, we can restrict our analysis to adversaries $\mathcal{A}$ who only send "meaningful random-oracle queries", which we define as queries of the form (d, i, ch, z), with $d = \mathsf{H}(x, \overline{f}), \overline{f} = (f_1, \ldots, f_L), \mathsf{ch} \in S$ and $\mathsf{V}_2(x, f_i, \mathsf{ch}, z) = \mathsf{accept}$. Otherwise, we could build an algorithm $\mathcal{A}'$ which uses $\mathcal{A}$ as a subroutine and answers to non-meaningful queries programming an *internal* (to $\mathcal{A}'$) random oracle, and answers to meaningful queries by forwarding the queries to the *real* random oracle. The interaction with $\mathcal{A}'$ would be indistinguishable for $\mathcal{A}$ from the interaction with a random oracle, therefore the probability that $\mathcal{A}$ outputs a valid proof π is unchanged, and if $\mathcal{A}'$ outputs the same proof π, the success probability of $\mathcal{A}$ and $\mathcal{A}'$ is the same. We also ask that the adversary $\mathcal{A}$ *always* sends a meaningful random-oracle query for each of the non-target components of the proof π. This can be assumed because if $\mathcal{A}$ does not do that, we could instruct the wrapper adversary $\mathcal{A}'$ to send the queries associated to those transcripts just before sending to $\mathsf{Ext}_{\blacktriangle}$ the proof π.

Let us assume that $\mathcal{A}$ outputs a valid proof π without providing the extractor $\mathsf{Ext}_{\blacktriangle}$ with two accepting transcripts for the same first message and different challenges. Intuitively, this implies that, for all $i \in [L]\setminus\mathcal{T}$, $\mathcal{A}$ has never queried the random oracle for (d, i, ch, z) for some challenge $\mathsf{ch} \in S$. This means that $\mathcal{A}$ has not checked if these components could be target components. Similarly, for all $i \in \mathcal{T}$, $\mathcal{A}$ has never queried the random oracle for (d, i, ch, z_i) for another challenge $\mathsf{ch} \in S \setminus \{\mathsf{ch}_i\}$. This means that $\mathcal{A}$ must have guessed the right challenge for each target component in $\mathcal{T}$.

Therefore, $\mathcal{A}$ has sent only ρ meaningful queries $d_{i_1}, \ldots, d_{i_\rho}$ for the ρ components in $\mathcal{T}$. In order for the proof to be successfully verified, the transcripts

associated to $d_{i_1}, \ldots, d_{i_\rho}$ must satisfy the inversion predicate, so the output of the random oracle on such queries $d_{i_1}, \ldots, d_{i_\rho}$ has to be smaller than $\lfloor 2^{\lambda-b} \rfloor$. The random oracle queries are distinct, therefore the returned outputs — which are all smaller than $\lfloor 2^{\lambda-b} \rfloor$ — have been sampled uniformly at random and independently by the random oracle. Therefore the probability that these ρ digests are all smaller than $\lfloor 2^{\lambda-b} \rfloor$ is approximately but less than $(2^{-b})^\rho$. As a consequence,

$$\epsilon_s^{\mathsf{Ext}_\blacktriangle} \leq 2^{-b\rho}.$$

Remark 8. Thanks to Eq. (3), given the parameters b, ρ, ℓ and a target probability 2^{-k}, it is easy to determine a value L that guarantees $\epsilon_c^{\mathsf{Prove}_\blacktriangle} \leq 2^{-k}$ (e.g. via binary search). This is immediate because $\epsilon_c^{\mathsf{Prove}_\blacktriangle}$ is a strictly decreasing function in the undetermined L.

B.2 Proof of Theorem 2

We show that $(\mathsf{Prove}_\blacksquare, \mathsf{Verify}_\blacksquare)$ is a NIZKP and quantify its completeness and soundness error. Then we determine its expected query complexity.

Completeness Being $\mathcal{T}_i \subseteq [L]$ the set of target components associated to the execution of $\mathsf{Prove}_\blacktriangle$ with random oracle H_i, the completeness error of $\mathsf{Prove}_\blacksquare$ is given by

$$\begin{aligned}\epsilon_c^{\mathsf{Prove}_\blacksquare} &= \Pr[\perp \leftarrow\!\$\ \mathsf{Prove}_\blacksquare(x,y) \mid (x,y) \in \mathcal{R}] = \Pr[\ |\mathcal{T}_i| < \tau,\ \forall i \in [k_{\max}]] \\ &= \Pr[|\mathcal{T}_i| < \tau]^{k_{\max}} = \left(\epsilon_c^{\mathsf{Prove}_\blacktriangle}\right)^{k_{\max}}.\end{aligned}$$

Zero-Knowledge We design a simulator $\mathsf{Sim}_\blacksquare = (\mathsf{Sim}_{\blacksquare,0}, \mathsf{Sim}_{\blacksquare,1})$ that employs the simulator $\mathsf{Sim}_\blacktriangle$ from Theorem 1 as a subroutine. For any query directed to oracle H_i before receiving a statement from the distinguisher, $\mathsf{Sim}_{\blacksquare,0}$ responds identically to $\mathsf{Sim}_{\blacktriangle,0}$ but maintains separate state for each of the $k_{\max}$ oracles by programming a corresponding table[22].
Upon receiving a statement-witness pair $(x, w) \in \mathcal{R}$ from the distinguisher $\mathcal{D}_0$, $\mathsf{Sim}_{\blacksquare,1}$ executes the following procedure:

1. $\mathsf{Sim}_{\blacksquare,1}$ sets $k = 1$;
2. if $k > k_{\max}$, $\mathsf{Sim}_{\blacksquare,1}$ returns $\perp$. Otherwise $\mathsf{Sim}_{\blacksquare,1}$ executes the same operations prescribed by $\mathsf{Sim}_{\blacktriangle,1}$ in the proof of the Zero-Knowledge property in Theorem 1 when $\mathsf{Sim}_{\blacktriangle,1}$ receives from the distinguisher a statement-witness pair in $\mathcal{R}$. The only difference is that it programs and gives the distinguisher access to $k_{\max}$ random oracles instead of one;
3. if $\mathsf{Sim}_{\blacksquare,1}$ fails in generating a proof (i.e. the execution of $\mathsf{Sim}_{\blacktriangle,1}$ outputs $\perp$), then $\mathsf{Sim}_{\blacksquare,1}$ increases k by 1 and starts again from Item 2;

[22] An alternative approach to constructing the simulator could involve programming a single hash table. For each input message m, the simulator would populate the hash table with the message prefixed by $i \leq k_{\max}$, i.e. $\langle i \rangle \| m$: this delineates the random oracle H_i that the input m was intended to be delivered to.

4. if $\mathsf{Sim}_{\blacksquare,1}$ succeeds in generating a proof π ($\mathsf{Sim}_{\blacktriangle,1}$ outputs a proof π), interacting with the random oracle H_k, then $\mathsf{Sim}_{\blacksquare}$ outputs the proof (k, π).

This simulator $\mathsf{Sim}_{\blacksquare}$ perfectly simulates the queries made by the distinguisher to the random oracles. The only case where the simulation fails is when the simulator has built the value $\overline{f}$ and needs to update the hash tables used to simulate the random oracles. In fact these values might be previously queried by the distinguisher. However, this happens with negligible probability since the distinguisher is a PPT algorithm and can query the random oracles a polynomial (in λ) number of times, whereas the number of possible $\overline{f}$ is super-polynomial in λ.

Straight-Line Extraction The extractor $\mathsf{Ext}_{\blacksquare}$ has access to the proof π produced in output by the algorithm A and all the random oracle queries sent to all the $k_{\max}$ oracles. The way $\mathsf{Ext}_{\blacksquare}$ works is the same as $\mathsf{Ext}_{\blacktriangle}$ (introduced in the analysis of the Straight-Line Extraction property in the proof of Theorem 1): it looks for two valid transcripts (f, ch_1, z_1) and (f, ch_2, z_2) for the statement x, with the same first message f and different challenges $\mathsf{ch}_1 \neq \mathsf{ch}_2$. Then it uses the witness extractor of the underlying Sigma protocol (P, V) to extract a witness w for x.

Again, without loss of generality, we can assume that A only performs meaningful queries of the form (d, i, ch, z) to $\mathsf{H}_k, k \in [k_{\max}]$, with d being $\mathsf{H}_k(x, \overline{f})$, $i \in [L]$ being the component of the vector $\overline{f} = (f_1, \ldots, f_L)$, $\mathsf{ch} \in S$ and $V(x, f_i, \mathsf{ch}, z) = \mathsf{accept}$. Also we assume that A makes a meaningful random oracle query for every transcript included in the proof π associated to $\mathsf{ch}_i \in S$. We can restrict to this kind of algorithm for the same reasons described in the proof of Theorem 1.

According to the proof of Theorem 1, A can create a valid proof without letting $\mathsf{Ext}_{\blacksquare}$ extract a witness only if it has managed to guess ρ target components $\mathcal{T}' = \{i_1, \ldots, i_\rho\} \subseteq [L]$ and ρ associated challenges $\{\mathsf{ch}_{i_1}, \ldots, \mathsf{ch}_{i_\rho}\} \in S$ such that, for some $k \in [k_{\max}]$, $\mathsf{H}_k(d, i_j, \mathsf{ch}_{i_j}, z_{i_j}) < \lfloor 2^{\lambda - b} \rfloor$, $\forall j \in [\rho]$, where $d = \mathsf{H}_k(x, \overline{f})$. This means that, once the prover has sent ρ queries of the form $\mathsf{H}_k(i_j, \mathsf{ch}_{i_j}, z_{i_j})$, for the same prefix $d = \mathsf{H}_k(x, \overline{f})$ and $\mathsf{ch}_{i_j} \in S$, if the associated digests are not all smaller than $\lfloor 2^{\lambda - b} \rfloor$, the only thing the prover can do to prevent $\mathsf{Ext}_{\blacksquare}$ from extracting the witness is to change the value k. In fact, changing the value k in the query does not let the extractor collect any other transcript useful for the witness extraction.

The probability of succeeding in creating a proof without letting the $\mathsf{Ext}_{\blacksquare}$ extract the witness is the probability that, given a prefix $(x, \overline{f})$, a set of target components $\mathcal{T} \subsetneq [L]$ and the associated challenges $\{\mathsf{ch}_{i_j}, i_j \in [\mathcal{T}]\}$, the prover generates a valid proof with at least one of the $k_{\max}$ random oracles. Since the output of the different random oracles are independent, and noting that the success probability of the prover interacting with a single random oracle is indeed $\epsilon_s^{\mathsf{Ext}_{\blacktriangle}}$, then

$$\epsilon_s^{\mathsf{Ext}_{\blacksquare}} = 1 - (1 - \epsilon_s^{\mathsf{Ext}_{\blacktriangle}})^{k_{\max}} = 1 - (1 - 2^{-b\rho})^{k_{\max}} \tag{9}$$

Clearly, $\epsilon_s^{\mathsf{Ext}_{\blacksquare}} > \epsilon_s^{\mathsf{Ext}_{\blacktriangle}}$, because $1 - \epsilon_s^{\mathsf{Ext}_{\blacksquare}} = (1 - \epsilon_s^{\mathsf{Ext}_{\blacktriangle}})^{k_{\max}} < (1 - \epsilon_s^{\mathsf{Ext}_{\blacktriangle}})$.

Query complexity To evaluate the query complexity, it is sufficient to compute the expected number of oracles a honest prover must interact with, which is $\frac{1}{1-\epsilon_c^{\mathsf{Prove}_\blacktriangle}}$ during the proof generation algorithm, and multiply it by $L \cdot \ell$, which is the number of hash queries the prover must compute every time it interacts with one of the $k_{\max}$ random oracles that does not let the prover generate a proof. In the last interaction the prover might compute less than $L \cdot \ell$ hash queries, so the following inequality holds:

$$\mathbb{E}[Q^{\mathsf{Prove}_\blacksquare}] \leq \frac{1}{1-\epsilon_c^{\mathsf{Prove}_\blacktriangle}} \cdot L \cdot \ell \tag{10}$$

B.3 Proof of Theorem 3

Completeness In [54] the authors empirically show that the query complexity in finding a collision of $2b$-bit long digests associated to two distinct first messages is lower than the query complexity of finding two transcripts that invert the digest 0^b (i.e. that are associated to the digest 0^b). In our construction, where the number ℓ of challenges that can be used to find the collision is small, the query complexity of finding a collision is even lower. See Remark 6. However, the value of L needed to obtain the desired completeness error must be determined empirically.

Straight-Line Extraction The arguments used to prove that the soundness error $\epsilon_s^{\mathsf{Ext}_\blacklozenge} \leq 2^{-b\rho}$ are similar to the ones used in Theorem 1 and in [54, Section 5.1]. To prevent the extractor from learning a witness for the proven statement, the adversary must not send a random oracle query for any transcript different from the ones included in the proof and one of the first messages in $\overline{f}$. This means that the adversary must guess components in $\mathcal{T}$ and challenges in S so that the collision predicate is satisfied. This happens if for every $\frac{\rho}{2}$ pairs of elements in $\mathcal{T}$ the digest collide pairwise. Two digests collide, according to our predicate, with probability $\leq \frac{1}{2^{2b}}$, and since there are $\frac{\rho}{2}$ pairs, the probability of producing a proof π without allowing the extractor $\mathsf{Ext}_\blacklozenge$ (which works like the extractor $\mathsf{Ext}_\blacktriangle$ introduced to prove Theorem 1) to extract the witness is $\leq 2^{-2b \cdot \frac{\rho}{2}} = 2^{-b\rho}$.

Zero-Knowledge The Simulator $(\mathsf{Sim}_{\blacklozenge,0}, \mathsf{Sim}_{\blacklozenge,1})$ is defined as the simulator $(\mathsf{Sim}_{\blacktriangle,0}, \mathsf{Sim}_{\blacktriangle,1})$ for the NIZKP obtained by applying the GAO transform, until it receives from the distinguisher a pair $(x, w) \in \mathcal{R}$ for which $\mathsf{Sim}_{\blacklozenge,1}$ must produce a simulated proof. As $\mathsf{Sim}_{\blacktriangle,1}$, $\mathsf{Sim}_{\blacklozenge,1}$ denotes the elements of S as $\{c_j\}_{j\in[\ell]}$ and initializes an empty set $T = \emptyset$. In the same way, it defines the maps τ_i, for $i \in [L]$. The only difference between the two simulators is that $\mathsf{Sim}_{\blacklozenge,1}$, instead of looking for the preimage of $[0, 2^{\lambda-b}]$ via the maps τ_i, looks for collisions among the images of pairs of maps τ_i, τ_j according to our collision predicate presented in Section 6. Of course, the pairs of colliding maps must be non-crossing. Apart from the way target components are chosen by $\mathsf{Sim}_{\blacklozenge,1}$, the simulation follows the one described in the proof of Theorem 1.

Query complexity The prover sends at most $L \cdot \ell$ random oracle queries, where L is the number of first messages, and $\ell + 1$ is the size of the challenge space.

B.4 Proof of Theorem 4

We show that $(\mathsf{Prove}_\bigstar, \mathsf{Verify}_\bigstar)$ is a NIZKP and quantify its completeness and soundness error. Then we determine its expected query complexity.

Completeness Being $\mathcal{T}_i \subseteq [L]$ the set of target components associated to the execution of $\mathsf{Prove}_\blacklozenge$ with random oracle H_i, the completeness error of $\mathsf{Prove}_\bigstar$ is given by

$$\begin{aligned}\epsilon_c^{\mathsf{Prove}_\bigstar} &= \Pr[\perp \leftarrow\!\$\ \mathsf{Prove}_\bigstar(x,y) \mid (x,y) \in \mathcal{R}] = \Pr[\ |\mathcal{T}_i| < \tau,\ \forall i \in [k_{\max}]] \\ &= \Pr[|\mathcal{T}_i| < \tau]^{k_{\max}} = \left(\epsilon_c^{\mathsf{Prove}_\blacklozenge}\right)^{k_{\max}}.\end{aligned} \tag{11}$$

Zero-Knowledge We design a simulator $\mathsf{Sim}_\bigstar = (\mathsf{Sim}_{\bigstar,0}, \mathsf{Sim}_{\bigstar,1})$ that employs the simulator $\mathsf{Sim}_\blacklozenge$ from Theorem 1 as a subroutine. For any query directed to oracle H_i before receiving a statement from the distinguisher, $\mathsf{Sim}_{\bigstar,0}$ responds identically to $\mathsf{Sim}_{\blacklozenge,0}$ but maintains separate state for each of the $k_{\max}$ oracles by programming a corresponding table.
Upon receiving a statement-witness pair $(x,w) \in \mathcal{R}$ from the distinguisher $\mathcal{D}_0$, $\mathsf{Sim}_{\bigstar,1}$ executes the following procedure:

1. $\mathsf{Sim}_{\bigstar,1}$ sets $k = 1$;
2. if $k > k_{\max}$, $\mathsf{Sim}_{\bigstar,1}$ returns $\perp$. Otherwise $\mathsf{Sim}_{\bigstar,1}$ executes the same operations prescribed by $\mathsf{Sim}_{\blacklozenge,1}$ in the proof of the Zero-Knowledge property in Theorem 3 when $\mathsf{Sim}_{\blacklozenge,1}$ receives from the distinguisher a statement-witness pair in $\mathcal{R}$. The only difference is that it programs and gives the distinguisher access to $k_{\max}$ random oracles instead of one;
3. if $\mathsf{Sim}_{\bigstar,1}$ fails in generating a proof (i.e. the execution of $\mathsf{Sim}_{\blacklozenge,1}$ outputs $\perp$), then $\mathsf{Sim}_{\bigstar,1}$ increases k by 1 and starts again from Item 2;
4. if $\mathsf{Sim}_{\bigstar,1}$ succeeds in generating a proof π ($\mathsf{Sim}_{\blacklozenge,1}$ outputs a proof π), interacting with the random oracle H_k, then $\mathsf{Sim}_\bigstar$ outputs the proof (k,π).

This simulator $\mathsf{Sim}_\bigstar$ perfectly simulates the queries made by the distinguisher to the random oracles. The only case where the simulation fails is when the simulator has built the value $\overline{f}$ and needs to update the hash tables used to simulate the random oracles. In fact these values might be previously queried by the distinguisher. However, this happens with negligible probability since the distinguisher is a PPT algorithm and can query the random oracles a polynomial (in λ) number of times, whereas the number of possible $\overline{f}$ is super-polynomial in λ.

Straight-Line Extraction The extractor $\mathsf{Ext}_\bigstar$ has access to the proof π produced in output by the algorithm A and all the random oracle queries sent to all the $k_{\max}$ oracles. The way $\mathsf{Ext}_\bigstar$ works is the same as $\mathsf{Ext}_\blacklozenge$ (introduced in the analysis of the Straight-Line Extraction property in the proof of Theorem 3): it looks for two valid transcripts (f, ch_1, z_1) and (f, ch_2, z_2) for the statement x, with the same first message f and different challenges $\mathsf{ch}_1 \neq \mathsf{ch}_2$. Then it uses the witness extractor of the underlying Sigma protocol (P, V) to extract a witness w for x.

Again, without loss of generality, we can assume that A only performs meaningful queries of the form (d, i, ch, z) to $\mathsf{H}_k, k \in [k_{\max}]$ with $d \leftarrow \mathsf{H}_k(x, \overline{f})$, $i \in [L]$ being the component of the vector $\overline{f} = (f_1, \ldots, f_L)$, $\mathsf{ch} \in S$ and $V(x, f_i, \mathsf{ch}, z) = \mathsf{accept}$. Also we assume that A makes a meaningful random oracle query for every transcript included in the proof π associated to $\mathsf{ch}_i \in S$. We can restrict to this kind of algorithm for the same reasons described in the proof of Theorem 1.

According to the proof of Theorem 3, A can create a valid proof without letting $\mathsf{Ext}_\bigstar$ extract a witness only if it has managed to guess ρ target components $\mathcal{T}' = \{i_1, \ldots, i_\rho\} \subseteq [L]$ and ρ associated challenges $\{\mathsf{ch}_{i_1}, \ldots, \mathsf{ch}_{i_\rho}\} \in S$ such that, for some $k \in [k_{\max}]$, the following holds: for every pair $(i_{j_1}, i_{j_2}) \leftarrow (\mathcal{T}[2j-1], \mathcal{T}[2j]), j \in [1, \frac{\rho}{2}]$, $\mathsf{H}_k(d, i_{j_1}, \mathsf{ch}_{i_{j_1}}, z_{i_{j_1}}) = \mathsf{H}_k(d, i_{j_2}, \mathsf{ch}_{i_{j_2}}, z_{i_{j_2}})$, where the equality in this case represent the membership of the two digests to the same interval, according to the collision predicate introduced in Section 6. This means that, once the prover has sent ρ queries of the form $\mathsf{H}_k(d, i_j, \mathsf{ch}_{i_j}, z_{i_j})$, for the same prefix $d = \mathsf{H}_k(x, \overline{f})$ and $\mathsf{ch}_{i_j} \in S$, if the digests associated to non-crossing target components do not collide, the only thing the prover can do to prevent $\mathsf{Ext}_\bigstar$ from extracting the witness is to change the value k. In fact, changing the value k in the query does not let the extractor collect any other transcript useful for the witness extraction.

The probability of succeeding in creating a proof without letting the $\mathsf{Ext}_\bigstar$ extract the witness is the probability that, given a prefix $(x, \overline{f})$, a set of target components $\mathcal{T} \subsetneq [L]$ and the associated challenges $\{\mathsf{ch}_{i_j}, i_j \in [\mathcal{T}]\}$, the prover generates a valid proof with at least one of the $k_{\max}$ random oracles. Since the output of the different random oracles are independent, and noting that the success probability of the prover interacting with a single random oracle is indeed $\epsilon_s^{\mathsf{Ext}_\blacklozenge}$, then

$$\epsilon_s^{\mathsf{Ext}_\bigstar} = 1 - (1 - \epsilon_s^{\mathsf{Ext}_\blacklozenge})^{k_{\max}} = 1 - (1 - 2^{-b\rho})^{k_{\max}} \tag{12}$$

Clearly, $\epsilon_s^{\mathsf{Ext}_\bigstar} > \epsilon_s^{\mathsf{Ext}_\blacktriangle}$, because $1 - \epsilon_s^{\mathsf{Ext}_\bigstar} = (1 - \epsilon_s^{\mathsf{Ext}_\blacktriangle})^{k_{\max}} < (1 - \epsilon_s^{\mathsf{Ext}_\blacktriangle})$.

Query complexity To evaluate the query complexity, it is sufficient to compute the expected number of oracles a honest prover must interact with, which is $\frac{1}{1-\epsilon_c^{\mathsf{Prove}_\blacktriangle}}$ during the proof generation algorithm, and multiply it by $L \cdot \ell$, which is the number of hash queries the prover must compute every time it interacts with one of the $k_{\max}$ random oracles that does not let the prover generate a proof. In the last interaction the prover might compute less than $L \cdot \ell$ hash queries, so the following inequality holds:

$$\mathbb{E}[Q^{\mathsf{Prove}_\bigstar}] \leq \frac{1}{1 - \epsilon_c^{\mathsf{Prove}_\blacktriangle}} \cdot L \cdot \ell \tag{13}$$

C Evaluating the Fischlin Transform

C.1 Naive Application of Fischlin Transform to $\Sigma_\star$

Since $\Sigma_\star$ has a binary challenge space, to reduce the completeness error it is necessary to apply the Fischlin transform to $\Sigma_\star^\alpha$ rather than $\Sigma_\star$, for a suitable

Table 5: Given the security parameter $\lambda = 128$, as parameters $b, r = \lceil 128/b \rceil$ vary we analyse the efficiency of the NIZKP obtained by applying the Fischlin transform to $\Sigma_\star^\alpha$, where $\alpha = \lceil b + 5 \rceil$ is the minimum value of α that guarantees a completeness error no greater than 2^{-40}. In particular, the table lists the number L of group action operations that must be computed ($r \cdot \alpha$), the expected overall weight ρ of the challenge ch (via Equation (15)) and the expected query complexity $\mathbb{E}[Q]$ (computed as $2^b r$).

r	b	L	$\mathbb{E}[\rho]$	$\mathbb{E}[Q]$
64	2	448	54	256
43	2.97	344	50	337
32	4	288	48	512
26	4.92	260	47	787
22	5.82	242	46	1242
19	6.74	228	45	2030
16	8	208	46	4096
15	8.53	210	45	5544
13	9.84	195	45	11915
12	10.67	192	45	19551

number α of parallel repetitions. It is therefore convenient, in this case, to consider α as a third parameter — in addition to b and r — of the Fischlin transform.

According to Equation (1), to guarantee a completeness error $\epsilon_c^{\mathsf{Prove}_{\mathtt{Fi}}} \leq 2^{-40}$, it is necessary to set $\alpha \geq b+5$, so that the challenge space of $\Sigma_\star^\alpha$ has size $2^\alpha \geq 2^{b+5}$. So, as a consequence of Equations (1) and (2), the number of first messages L of $\Sigma_\star$ to be computed by $\mathsf{Prove}_{\mathtt{Fi}}$ is $L = (\alpha)r = br + (\alpha - b)r = \lambda + (\alpha - b)r \geq \lambda + 5r$. This means that a proof π, produced by $\mathsf{Prove}_{\mathtt{Fi}}$ on input a pair $(x, w) \in R_\star$, is of the form $(\{f_i\}_{i\in[r]}, \{\mathsf{ch}_i\}_{i\in[r]}, \{z_i\}_{i\in[r]})$ such that, for each $i \in [r]$, the first message f_i is of the kind $(f_{i,1}, \ldots, f_{i,\alpha})$, where $f_{i,j}$ is a first message of $\Sigma_\star$. Moreover, the challenge ch_i is a bit string $(\mathsf{ch}_{i,1}, \ldots, \mathsf{ch}_{i,\alpha}) \in \{0,1\}^\alpha$, and the response z_i is of the form $(z_{i,1}, \ldots, z_{i,\alpha})$, where $z_{i,j}$ is the response (with respect to $\Sigma_\star$) corresponding to $f_{i,j}$ and $\mathsf{ch}_{i,j}$.

We observe that, since the Sigma protocol $\Sigma_\star$ is first-message retrievable (see Definition 11), then the size of π can be reduced by omitting the first messages, i.e. we can assume $\pi = (\{\mathsf{ch}_i\}_{i\in[r]}, \{z_i\}_{i\in[r]})$. Since $\Sigma_\star$ is also seed-friendly (see Definition 1), the first messages $f_{i,j}$ can be generated starting from seeds $\mathsf{seed}_{i,j}$ so that the responses $z_{i,j}$ corresponding to the challenge $\mathsf{ch}_{i,j} = 0$ can be encoded as seeds (then the *complete* response is obtained by computing $Z(x, \mathsf{PRF}(\mathsf{seed}_{i,j}))$).

Below, we discuss how $\mathsf{Prove}_{\mathtt{Fi}}$ should proceed in creating a the proof π which keeps the weight of the challenges ch_i included in π minimal and therefore maximizes the number of challenges 0 which allow to encode the corresponding response with a compact λ-bit seed.

Choosing Lighter Challenges First. To reduce the weight of the challenges $\mathsf{ch}_1, \ldots, \mathsf{ch}_r \in \{0,1\}^\alpha$ found while creating a proof, the proof-generation algorithm must try the challenges in increasing order of weight. In particular, for each $i \in [r]$, the unique challenge of weight 0 (which is 0^α) must be tested first, then

the $\alpha = \binom{\alpha}{1}$ challenges of weight 1, the $\binom{\alpha}{2}$ challenges of weight 2, etc., up to the unique challenge of weight α. As a result, the algorithm will find the lighter challenge ch_i satisfying the predicate $\mathsf{H}(x, \overline{f}, i, \mathsf{ch}_i, z_i) < \lfloor 2^{\lambda - b} \rfloor$. While this specification does not alter the completeness error $\epsilon_c^{\mathsf{Prove}_{\mathtt{Fi}}}$, it allows us to determine the expected overall weight of the challenge $\mathsf{ch} = (\mathsf{ch}_1, \ldots, \mathsf{ch}_r)$ in a proof π, as follows. For every $j \in \{0\} \cup [\alpha]$, we denote by $k_j = \sum_{i=0}^{j} \binom{\alpha}{i}$ the number of challenges of weight no greater than j. For $i \in [r]$, let $Q_i(b)$ be the random variable that describes the number of attempts to find a transcript which satisfies the required predicated in the i-th iteration of $\Sigma_\star$, and let $W_i(b, \alpha)$ be the random variable that represents the weight of the lighter challenge satisfying $\mathsf{H}(x, \overline{f}, i, \mathsf{ch}_i, z_i) < \lfloor 2^{\lambda - b} \rfloor$. Then the random variable representing the overall weight of the proof π is $W(r, b, \alpha) = \sum_{i=1}^{r} W_i(b, \alpha)$.

It is possible to compute the expected value of the random variable $W(r, b, \alpha)$ by means of the below equation:

$$\mathbb{E}[W(r, b, \alpha)] = \mathbb{E}[\sum_{i=1}^{r} W_i(b, \alpha)] = \sum_{i=1}^{r} \mathbb{E}[W_i(b, \alpha)] = r \cdot \mathbb{E}[W_i(b, \alpha)] \tag{14}$$

where the expected value of $W_i(b, \alpha), \forall i \in [r]$, is approximated by

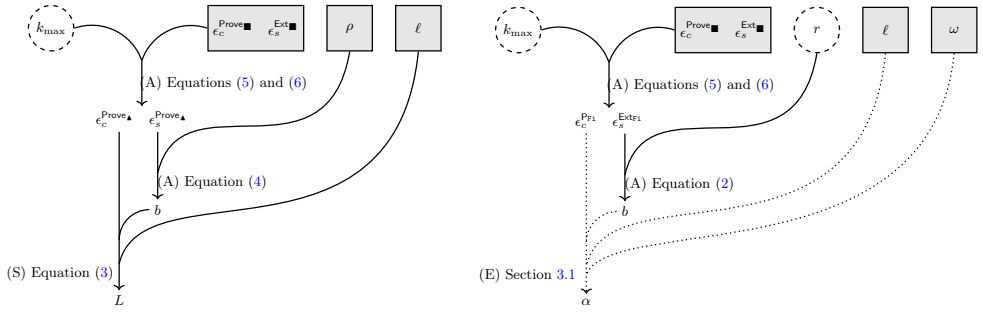
$$\begin{aligned} \mathbb{E}[W_i(b, \alpha)] &= \sum_{j=1}^{\alpha} j \cdot \Pr[W_i(b, \alpha) = j] = \sum_{j=1}^{\alpha} j \cdot \Pr[k_{j-1} < Q_{\mathtt{Fi}}(b) \leq k_j] \\ &\approx \sum_{j=1}^{\alpha} j \cdot (1 - 2^{-b})^{k_{j-1}} (1 - (1 - 2^{-b})^{k_j - k_{j-1}}). \end{aligned} \tag{15}$$

C.2 Combining Stretch-and-Compress with Fischlin Transform

It is straightforward to see that the same ideas behind the Stretch-and-Compress technique presented in Definition 4 can be applied using as a subroutine the NIZKP obtained using the Fischlin transform instead of the GAO transform. This would surely improve the performance of the Fischlin transform when applied to a Sigma protocol like the one from cryptographic group actions.

However, unlike when we apply as a subroutine $(\mathsf{Prove}_{\blacktriangle}, \mathsf{Verify}_{\blacktriangle})$ (the NIZKP obtained using the GAO transform), if we use as a subroutine the NIZKP $(\mathsf{P}_{\mathtt{Fi}}, \mathsf{V}_{\mathtt{Fi}})$ derived using the Fischlin transform, we encounter two difficulties in choosing a parameterization:

- given the completeness and soundness errors $\epsilon_c^{\mathsf{Prove}_{\blacksquare}}, \epsilon_s^{\mathsf{Ext}_{\blacksquare}}$, a target weight ω, the number of challenges in $\mathsf{ChSpace} \setminus \{\tilde{\mathsf{ch}}\}$, we have two degrees of freedom given by the number of random oracle $k_{\max}$ but also the number of repetitions r of the Fischlin transform;
- finding an optimal parameterization is made even more challenging by our incapability to find an efficiently computable analytical equations that enables finding the optimal value of α given $\epsilon_c^{\mathsf{P}_{\mathtt{Fi}}}$.



(a) Given the values $\epsilon_c^{\mathsf{Prove}\blacksquare}, \epsilon_s^{\mathsf{Ext}\blacksquare}, \rho, \ell$ (in boxes filled in gray), express the values $\epsilon_c^{\mathsf{Prove}\blacktriangle}, \epsilon_s^{\mathsf{Ext}\blacktriangle}, b, L$ as functions of the maximum number $k_{\max}$ of random oracles that the prover can use (in the dashed circle).

(b) Given the values $\epsilon_c^{\mathsf{Prove}\blacksquare}, \epsilon_s^{\mathsf{Ext}\blacksquare}, \ell, \omega$ (in boxes filled in gray), express the values $\epsilon_c^{\mathsf{P_{Fi}}}, \epsilon_s^{\mathsf{Ext_{Fi}}}, b, \alpha$ as functions of $k_{\max}$ and the number of repetitions r.

Fig. 3: Describe the dependency of parameters in Definition 4 (a) and Definition 19 (b). The arrows describe the nature of the dependency: the parameters at the tail of the arrows determine the value of the parameters at the head. This dependency is determined by the equation specified alongside the arrow. If the relation is analytical, i.e. we have an exact formula for the parameters at the head, we write (A) next to the equation, if the parameter at the head of the arrow must be found using binary search, we write (S) and if the parameter at the head of the arrow must be found empirically (simulating the prover generation algorithm), we write (E) and the corresponding arrows are dotted.

The former difficulty requires to loop over the relevant values of r to find the optimal parameterization. The latter difficulty, which is more problematic, is an obvious consequence of the discussion in Section 3.1, in fact, it would be impractical to find the optimal parameter α so that the completeness error of the underlying subroutine $(\mathsf{P_{Fi}}, \mathsf{V_{Fi}})$ is below the desired threshold $\epsilon_c^{\mathsf{P_{Fi}}}$.

In Figure 1 we highlight the relationship among the parameters that must be determined to instantiate a NIZKP using the Stretch-and-Compress technique both using the GAO transform (on the left) and using the Fischlin transform (on the right). Figure 1 describes how the parameter choices must be made and highlights how the application with the GAO transform is better even in terms of simplicity of the relations among the parameters.

D Technical Details

D.1 Inversion Predicate and Completeness Error in Theorem 1

In this section we argue that $\epsilon_c^{\mathsf{Prove}\blacktriangle}$ is negligibly close to $\sum_{i=0}^{\rho-1} \binom{L}{i} p^i (1-p)^{L-i}$, with $p = 1 - (1-2^{-b})^{\ell}$ as described in Theorem 1.

The point is that the GAO predicate is satisfied if $\mathsf{H}(d, i, \mathsf{ch}_i, z_i) < \lfloor 2^{\lambda - b} \rfloor$ and this happens with probability 2^{-b} only if $2^{\lambda-b}$ is integer, otherwise the probability is negligibly smaller than that, due to the loss of precision in the

computation of the integer part of $2^{\lambda-b}$. This slightly reduces the probability that a digest falls in the interval that makes the predicate being satisfied, and therefore the probability that a component is a target component.

Given λ, b, it is possible to determine $\bar{b} = -\log_2\left(\frac{\lfloor 2^{\lambda-b}\rfloor}{2^\lambda}\right)$ (i.e. $2^{-\bar{b}} = \frac{\lfloor 2^{\lambda-b}\rfloor}{2^\lambda}$) which is the irrational number such that $\Pr\left[\mathsf{H}(x) < \lfloor 2^{\lambda-b}\rfloor\right] = 2^{-\bar{b}}$.

Let us define $\bar{p}$ as the probability that a component is a target component, meaning that there is a challenge that makes the predicate $\mathsf{H}(d, i, \mathsf{ch}_i, z_i) < \lfloor 2^{\lambda-b}\rfloor$ hold. This is the real probability and is computed as

$$\bar{p} = 1 - (1 - 2^{-\bar{b}})^\ell,$$

which is smaller than the probability p introduced in Theorem 1

$$p = 1 - (1 - 2^{-b})^\ell.$$

However, these two values are negligibly close, and setting $\delta = \mid p - \bar{p} \mid$ we can write

$$\begin{aligned}
\delta &= 1 - (1 - 2^{-b})^\ell - (1 - (1 - 2^{-\bar{b}})^\ell) \\
&= (1 - 2^{-\bar{b}})^\ell - (1 - 2^{-b})^\ell \\
&= (1 - 2^{-\bar{b}} - 1 + 2^{-b}) \left(\sum_{i+j=\ell-1, i,j\in\mathbb{N}} (1 - 2^{-\bar{b}})^i (1 - 2^{-b})^j \right) \\
&\leq (2^{-b} - 2^{-\bar{b}}) \underbrace{\ell}_{\text{small}} \underbrace{(1 - 2^{-\bar{b}})^{\ell-1}}_{<1}
\end{aligned}$$

where

$$2^{-b} - 2^{-\bar{b}} = \frac{2^{\lambda-b} - \lfloor 2^{\lambda-b}\rfloor}{2^\lambda} \leq 2^{-\lambda}$$

So, to conclude,

$$\delta \leq 2^{-\lambda} \underbrace{\ell}_{\text{small}} \underbrace{(1 - 2^{-\bar{b}})^{\ell-1}}_{<1}$$

Now that we have upper-bounded the difference between the probability p that we use in our description and the real probability $\bar{p}$ corresponding to our construction, we can upper-bound also the real completeness error $\bar{\epsilon}_c^{\mathsf{Prove}_\blacktriangle}$ and $\epsilon_c^{\mathsf{Prove}_\blacktriangle}$ that we describe in Theorem 1.

Again, $\bar{\epsilon}_c^{\mathsf{Prove}_\blacktriangle} \geq \epsilon_c^{\mathsf{Prove}_\blacktriangle}$ because the floor might reduce the successful interval in the space of digests when the predicate is verified. We can compute

$$\begin{aligned}
\bar{\epsilon}_c^{\mathsf{Prove}_\blacktriangle} - \epsilon_c^{\mathsf{Prove}_\blacktriangle} &= \sum_{i=0}^{\rho-1} \binom{L}{i} \left((\bar{p})^i(1-\bar{p})^{L-i} - (p)^i(1-p)^{L-i}\right) = \\
&= \sum_{i=0}^{\rho-1} \binom{L}{i} \left((p-\delta)^i(1-p+\delta)^{L-i} - (p)^i(1-p)^{L-i}\right) \leq \\
&\leq \sum_{i=0}^{\rho-1} \binom{L}{i} (p)^i \left((1-p+\delta)^{L-i} - (1-p)^{L-i}\right) = \\
&= \sum_{i=0}^{\rho-1} \binom{L}{i} (p)^i \left((1-p+\delta-1+p)\left(\sum_{i+j=L-i-1, i,j\in\mathbb{N}} (1-p+\delta)^i(1-p)^j\right)\right) \leq \\
&\leq \sum_{i=0}^{\rho-1} \binom{L}{i} (p)^i \delta \left((L-i)(1-p+\delta)^{L-i-1}\right) \leq \\
&\leq \delta \underbrace{L}_{\mathrm{poly}(\lambda)} \underbrace{\sum_{i=0}^{\rho-1} \binom{L}{i} (p)^i (1-p+\delta)^{L-1-i}}_{\leq 1+\mathsf{negl}(\lambda)} \leq 2^{-\lambda+\log(\mathsf{poly}(\lambda))}
\end{aligned}$$

We consider values of $L \leq 2^{13} \approx 8000$ since the first messages are usually expensive to compute, so it is not realistic to consider the application of the GAO transform with such a great value of L.

This allows us to state that the value $\epsilon_c^{\mathsf{Prove}_\blacktriangle}$ described in Theorem 1 is negligibly close to the real completeness error that considers the fact that $2^{\lambda-b}$ might not be integer, so the prover must consider its floor in the verification of the GAO predicate.

D.2 Collision Predicate Used in Definition 6

In this section, we formally show that the partition of the interval $[0, 2^\lambda - 1]$ in k intervals of size q, and an interval of size $r = n - qk$, as it is done in Section 6 combined with the ability to randomly sample an element in $[0, 2^\lambda - 1]$ allows to define random variables with a probability to have a collision which is negligibly close to 2^{-b} for an arbitrary rational value b.

Our Approach. Our approach is the following: being $N = 2^\lambda$ we divide in $k+1$ intervals the set $\{0, \ldots, N-1\}$ creating k intervals of cardinality q and the last interval of cardinality r, with $r = N - kq$. Then the i-th interval is defined as $[q(i-1), qi - 1], \forall i \in \{1, \ldots, k\}$ and the $k+1$-th interval is $[qk - 1, N - 1]$.

In this way, if we consider the random variable $X_{q,k,r} \in \{1, 2, \ldots, k+1\}$ associated to the random sampling of a random element in $[0, N-1]$ and which returns the associated interval, as defined above, then it is possible to describe the probability distribution of $X_{q,k,r} \in \{1, 2, \ldots, k+1\}$, as follows:

$$\Pr[X_{q,k,r} = x] = \begin{cases} \frac{q}{n} & \text{if } 1 \leq x \leq k \\ \frac{r}{n} & \text{if } x = k+1 \end{cases}$$

This probability distribution allows us to compute the probability that two independent, identically distributed random variables $X_1 = X_2$ distributed as $X_{q,k,r}$ have a collision in output:

$$\Pr[X_1 = X_2] = \sum_{i=1}^{k+1} \Pr[X_1 = X_2 = i] = \sum_{i=1}^{k+1} \Pr[X_1 = i] \Pr[X_2 = i] = \tag{16}$$

$$= \sum_{i=1}^{k+1} \Pr[X_{q,k,r} = x]^2 = \frac{kq^2 + r^2}{n^2}. \tag{17}$$

We want to choose the values k and q (that given N determine the value r) so that the collision probability described above $\mathrm{CP}(k,q) = \Pr[X_1 = X_2] = 2^{-2b}$ for some rational b.

To simplify the notation, we set $\beta = 2^{-2b}$, then we must find the solutions to $\mathrm{CP}(k,q) = \beta$ i.e.:

$$\frac{kq^2 + (N - kq)^2}{N^2} = \beta$$

That can be rewritten as

$$\frac{kq^2 + (N^2 - 2Nkq + k^2q^2) - N^2\beta}{N^2} = 0$$

$$\implies (k^2 + k)q^2 - 2Nkq + N^2(1 - \beta) = 0$$

$$\implies q_0 = \frac{Nk \pm N\sqrt{k(\beta k - 1 + \beta)}}{k^2 + k}.$$

so, the admissible integers $k > 0$ for which at least a solution exist are $k \geq \left\lceil \frac{1-\beta}{\beta} \right\rceil$. We will describe how to choose the value of k once we have identified how the collision probability depends on k.

If we consider k as a determined value, we can find one of the values of q satisfying the equation which is:

$$q_0 = \frac{Nk - N\sqrt{k(\beta k - 1 + \beta)}}{k^2 + k}.$$

This value q_0 is real, instead we want to find a natural number which determines the cardinality of the first k intervals which partition $[0, N-1]$. We consider the natural number $\bar{q} = \lceil q_0 \rceil$ and we show that $\mid \mathrm{CP}(\bar{q}) - \mathrm{CP}(q_0) \mid$ is negligible. We recall that

$$\mathrm{CP}(q) = \frac{kq^2 + r^2}{N^2} = \frac{kq^2 + (N - kq)^2}{N^2} = \frac{(k^2 + k)}{N^2}q^2 - \frac{2k}{N}q + 1$$

so it is possible to evaluate the gradient of the parabola at q_0, which is

$$\begin{aligned}\mathrm{CP}'(q_0) =& \frac{2(k^2+k)}{N^2}q_0 - \frac{2k}{N} = \frac{2(k^2+k)}{N^2}\cdot\frac{Nk - N\sqrt{k(\beta k-1+\beta)}}{k^2+k} - \frac{2k}{N} = \\ =& \frac{2(k-\sqrt{k(\beta k-1+\beta)})}{N} - \frac{2k}{N} = \frac{2}{N}\left(\left(k-\sqrt{k(\beta k-1+\beta)}\right)-k\right) = \\ =& -\frac{2}{N}\sqrt{k(\beta k-1+\beta)}\end{aligned}$$

Since $\beta < 1$, then

$$\mid \mathrm{CP}'(q_0) \mid \leq \frac{2k}{N}.$$

We have defined $\bar{q} = \lceil q_0 \rceil \geq q_0$, so $\bar{q} - q_0 < 1$. Since the gradient of the parabola increases as q increases (the coefficient of degree 2 is positive), we can state that, if $\mathrm{CP}(\bar{q}) \leq \mathrm{CP}(q)$ then

$$\mid \mathrm{CP}(q_0) - \mathrm{CP}(\bar{q}) \mid \leq \mid \mathrm{CP}'(q_0) \mid \leq \frac{2k}{N}.$$

Instead if $\mathrm{CP}(\bar{q}) > \mathrm{CP}(q)$, then it means that $\bar{q}$ is after the vertex of the parabola, so

$$\mid \mathrm{CP}(\bar{q}) - \mathrm{CP}(q_0) \mid \leq \mid \mathrm{CP}(q_0+1) - \mathrm{CP}(q_0) \mid,$$

which can be proven to be $\leq \frac{2k}{N} + \frac{k^2+k}{N^2}$.

This allows us to conclude that

$$\mid \mathrm{CP}(\bar{q}) - \mathrm{CP}(q_0) \mid \leq \max\{\frac{2k}{N}, \frac{2k}{N} + \frac{k^2+k}{N^2}\}$$

At this point it becomes clear that the smaller is k, the better will be the approximation, so we choose $k = \left\lceil \frac{1-\beta}{\beta} \right\rceil$ which is the minimum admissible value.

Since $\frac{1}{\beta}$ must be polynomial in λ, as the prover must be able to find collisions in polynomial time (to generate NIZKP), and $k = \left\lceil \frac{1}{\beta}(1-\beta) \right\rceil$, we can state that $\mid \mathrm{CP}(\bar{q}) - \mathrm{CP}(q_0) \mid$ is negligible.

E Additional Parametrizations for SC-GAO and SC-Coll-GAO

This appendix provides comprehensive parametrizations for the SC-GAO transform (Table 6) and SC-Coll-GAO transform (Table 7) across extended ranges of challenge space sizes ℓ and target weights ρ. While Tables 3 and 4 presented selected optimal parameters to demonstrate the optimization methodology, the tables below offer complete parametrization data for practitioners implementing these transforms in different cryptographic contexts. Each table reports the optimal values of $k_{\max}$, b, and L that minimize computational complexity subject

to the security constraints $\epsilon_{\mathsf{C}} \leq 2^{-40}$ and $\epsilon_{\mathsf{S}} \leq 2^{-128}$. The parametrizations follow the optimization methodology described in Section 5.3, with computational costs measured according to the framework established in Section 2. Furthermore, each parameterization includes a direct comparison with the equivalent version of LESS Fiat-Shamir (i.e., with the same target weight).

Table 6: Optimal parameters for SC-GAO (Definition 4) to minimize the computational complexity, for a given choice of ℓ and ρ. The computational complexities and signature sizes are compared in gray with equivalent LESS parameterizations (same weight ρ).

ℓ	ρ	k	b	L	$\mathbb{E}[Q]$	$\bar{Q}$	CC	**sig (B)**
	36	301	3.78	392	4.44K	17.2K	254M (×2.3)	3233 (×1.2)
	32	282	4.25	477	5.09K	19.6K	308M (×2.3)	3169 (×1.2)
	28	294	4.86	621	6.88K	26.6K	402M (×2.4)	3041 (×1.2)
1	24	358	5.69	896	12K	46.6K	589M (×2.4)	2865 (×1.2)
	20	358	6.82	1584	21.2K	82.4K	1.04B (×2.4)	2769 (×1.1)
	18	360	7.58	2357	31.7K	123K	1.55B (×2.5)	2737 (×1.1)
	16	381	8.54	3916	55.7K	217K	2.59B (×2.5)	2689 (×1.1)
	36	158	3.76	214	2.65K	9.93K	140M (×2.3)	2753 (×1.3)
	32	157	4.23	257	3.17K	11.9K	169M (×2.4)	2641 (×1.2)
	28	181	4.84	329	4.62K	17.5K	218M (×2.4)	2593 (×1.2)
2	24	198	5.65	475	7.26K	27.6K	317M (×2.5)	2577 (×1.2)
	20	212	6.79	832	13.6K	51.6K	559M (×2.6)	2449 (×1.2)
	18	220	7.54	1233	20.8K	79.4K	831M (×2.6)	2449 (×1.2)
	16	212	8.48	2066	33.6K	128K	1.39B (×2.6)	2401 (×1.1)
	36	100	3.74	154	1.9K	6.85K	102M (×2.2)	2465 (×1.3)
	32	124	4.22	180	2.69K	9.89K	121M (×2.4)	2401 (×1.3)
	28	130	4.82	230	3.59K	13.3K	155M (×2.5)	2385 (×1.3)
3	24	163	5.64	326	6.22K	23.4K	224M (×2.6)	2321 (×1.3)
	20	161	6.77	572	10.8K	40.6K	392M (×2.7)	2305 (×1.2)
	18	154	7.51	853	15.5K	58K	583M (×2.7)	2321 (×1.2)
	16	162	8.46	1416	27K	101K	972M (×2.8)	2273 (×1.2)
	36	83	3.73	122	1.71K	6.06K	81.7M (×2.2)	2289 (×1.4)
	32	89	4.20	143	2.12K	7.56K	96.2M (×2.3)	2257 (×1.4)
	28	101	4.81	180	2.99K	10.8K	122M (×2.5)	2209 (×1.3)
4	24	107	5.61	257	4.5K	16.3K	176M (×2.6)	2161 (×1.2)
	20	117	6.74	445	8.42K	30.9K	307M (×2.8)	2225 (×1.2)
	18	127	7.50	655	13.3K	49K	455M (×2.8)	2209 (×1.2)
	16	132	8.44	1087	22.9K	84.9K	759M (×2.9)	2225 (×1.2)
	36	78	3.73	102	1.7K	5.98K	69.7M (×2.2)	2129 (×1.4)
	32	78	4.20	119	1.99K	6.97K	81.3M (×2.3)	2129 (×1.4)
	28	80	4.80	150	2.55K	8.99K	103M (×2.5)	2081 (×1.4)
5	24	107	5.61	208	4.54K	16.5K	147M (×2.7)	2049 (×1.3)
	20	100	6.73	364	7.48K	27K	255M (×2.8)	2097 (×1.2)
	18	108	7.49	535	11.8K	42.7K	377M (×2.9)	2113 (×1.2)
	16	109	8.42	890	19.8K	72K	628M (×2.9)	2145 (×1.2)
	36	62	3.72	90	1.49K	5.05K	61.7M (×2.1)	2017 (×1.4)
	32	63	4.19	104	1.74K	5.92K	71.4M (×2.3)	1985 (×1.4)
	28	76	4.79	128	2.51K	8.78K	89.4M (×2.5)	1937 (×1.3)
6	24	82	5.60	180	3.75K	13.3K	127M (×2.7)	1985 (×1.3)
	20	79	6.72	313	6.31K	22.2K	219M (×2.9)	2033 (×1.2)
	18	94	7.48	454	10.6K	38.1K	324M (×2.9)	2033 (×1.2)
	16	100	8.42	750	18.6K	67.1K	540M (×3.0)	2081 (×1.2)
	36	46	3.71	82	1.26K	4.06K	56.1M (×2.1)	1921 (×1.4)
	32	68	4.19	91	1.9K	6.53K	64.3M (×2.3)	1921 (×1.4)
	28	62	4.78	114	2.21K	7.5K	79.9M (×2.5)	1905 (×1.4)
7	24	68	5.59	160	3.32K	11.4K	113M (×2.7)	1889 (×1.3)
	20	89	6.72	266	6.94K	24.7K	194M (×2.9)	1921 (×1.2)
	18	82	7.46	397	9.6K	33.9K	286M (×3.0)	1985 (×1.2)
	16	91	8.41	652	17.3K	61.9K	476M (×3.1)	1985 (×1.2)

Table 7: Optimal parameters for SC-Coll-GAO (Definition 7) to minimize the computational complexity, for a given choice of ℓ and ρ. The computational complexities and signature sizes are compared in gray with equivalent LESS parameterizations (same weight ρ).

ℓ	ρ	k	b	L	$\mathbb{E}[Q]$	$\bar{Q}$	CC	**sig (B)**
	36	197	3.77	274	2.08K	7.89K	174M (×1.6)	2929 (×1.1)
	32	163	4.23	343	2.18K	8.19K	216M (×1.6)	2913 (×1.1)
	28	199	4.84	427	3.28K	12.5K	271M (×1.6)	2801 (×1.1)
1	24	270	5.67	595	6.09K	23.4K	383M (×1.6)	2673 (×1.1)
	20	102	6.73	1176	4.94K	17.9K	729M (×1.7)	2625 (×1.1)
	18	162	7.52	1730	10.9K	41.1K	1.09B (×1.7)	2593 (×1.1)
	16	355	8.53	2738	36.4K	141K	1.8B (×1.7)	2545 (×1.1)
	36	69	3.73	156	942	3.25K	98.4M (×1.6)	2481 (×1.2)
	32	86	4.20	185	1.34K	4.77K	118M (×1.6)	2449 (×1.2)
	28	95	4.81	235	1.84K	6.61K	150M (×1.7)	2401 (×1.2)
2	24	143	5.63	329	3.72K	13.8K	215M (×1.7)	2321 (×1.1)
	20	174	6.77	587	7.95K	30K	388M (×1.8)	2321 (×1.1)
	18	164	7.52	839	10.8K	40.5K	552M (×1.7)	2305 (×1.1)
	16	508	8.56	1382	51.9K	204K	1.05B (×2.0)	2289 (×1.1)
	36	67	3.72	112	981	3.37K	72.2M (×1.6)	2177 (×1.2)
	32	64	4.19	133	1.13K	3.85K	85.6M (×1.7)	2161 (×1.2)
	28	105	4.81	160	2.07K	7.5K	106M (×1.7)	2097 (×1.1)
3	24	67	5.59	230	2.02K	6.93K	148M (×1.7)	2145 (×1.2)
	20	85	6.72	403	4.31K	15.3K	263M (×1.8)	2145 (×1.1)
	18	91	7.47	592	6.74K	24.1K	388M (×1.8)	2113 (×1.1)
	16	59	8.37	1031	8.16K	27.4K	662M (×1.9)	2161 (×1.1)
	36	65	3.72	87	1K	3.42K	57.4M (×1.6)	2001 (×1.2)
	32	45	4.17	105	912	2.92K	68M (×1.7)	2001 (×1.2)
	28	86	4.80	124	1.79K	6.34K	83.2M (×1.7)	1985 (×1.2)
4	24	98	5.61	175	2.82K	10.2K	119M (×1.8)	2001 (×1.1)
	20	51	6.68	319	3.02K	9.89K	208M (×1.9)	2065 (×1.2)
	18	89	7.47	455	6.76K	24.1K	306M (×1.9)	2049 (×1.1)
	16	92	8.41	745	11.3K	40.6K	502M (×1.9)	2065 (×1.1)
	36	52	3.71	75	899	2.96K	49.8M (×1.6)	1841 (×1.2)
	32	44	4.17	89	952	3.03K	58.7M (×1.7)	1889 (×1.3)
	28	52	4.78	107	1.28K	4.2K	71.1M (×1.7)	1889 (×1.2)
5	24	36	5.55	152	1.39K	4.21K	99.2M (×1.8)	1873 (×1.1)
	20	58	6.69	254	3.31K	11.1K	170M (×1.9)	1969 (×1.2)
	18	78	7.46	364	6.06K	21.3K	249M (×1.9)	1953 (×1.1)
	16	91	8.41	586	11.1K	39.6K	406M (×1.9)	1953 (×1.1)
	36	32	3.69	69	709	2.08K	45.5M (×1.6)	1745 (×1.2)
	32	45	4.17	75	965	3.08K	50.3M (×1.6)	1761 (×1.3)
	28	64	4.79	91	1.55K	5.28K	62.6M (×1.7)	1777 (×1.2)
6	24	40	5.56	127	1.51K	4.7K	84.7M (×1.8)	1841 (×1.2)
	20	92	6.73	207	4.74K	16.9K	147M (×1.9)	1857 (×1.1)
	18	52	7.43	318	4.55K	15K	215M (×2.0)	1905 (×1.1)
	16	43	8.34	539	6.76K	21.4K	361M (×2.0)	1937 (×1.1)
	36	33	3.70	62	763	2.27K	41.6M (×1.6)	1633 (×1.2)
	32	29	4.15	69	770	2.18K	46M (×1.6)	1681 (×1.3)
	28	47	4.77	80	1.24K	4K	54.8M (×1.7)	1649 (×1.2)
7	24	54	5.57	110	1.92K	6.36K	76.2M (×1.8)	1729 (×1.2)
	20	72	6.71	176	3.83K	13.3K	125M (×1.9)	1729 (×1.1)
	18	64	7.44	272	5.39K	18.4K	191M (×2.0)	1777 (×1.1)
	16	62	8.37	439	8.43K	28.6K	307M (×2.0)	1889 (×1.2)