Improved Straight-Line Extraction in the Random Oracle Model With Applications to Signature Aggregation

No Institute Given

Abstract. The goal of this paper is to *improve the efficiency and applicability* of straightline extraction techniques in the random oracle model. *Straightline extraction in the random oracle model* refers to the existence of an extractor, which given the random oracle queries made by a prover $P^*(x)$ on some theorem x, is able to produce a witness w for x with roughly the same probability that P^* produces a verifying proof. This notion applies to both zero-knowledge protocols and verifiable computation where the goal is *compressing* a proof.

Pass (CRYPTO '03) first showed how to achieve this property for NP using a *cut-and-choose* technique which incurred a λ^2-bit overhead in communication where λ is a security parameter. Fischlin (CRYPTO '05) presented a more efficient technique based on "proofs of work" that sheds this λ^2 cost, but only applies to a limited class of Sigma Protocols with a "quasi-unique response" property, which for example, does not necessarily include the standard OR composition for Sigma protocols.

With *Schnorr/EdDSA signature aggregation* as a motivating application, we develop new techniques to improve the computation cost of straight-line extractable proofs. Our improvements to the state of the art range from *70×–200×* for the best compression parameters. This is due to a uniquely suited polynomial evaluation algorithm, and the insight that a proof-of-work that relies on multicollisions and the birthday paradox is faster to solve than inverting a fixed target.

Our collision based proof-of-work more generally improves the Prover's random oracle query complexity when applied in the NIZK setting as well. In addition to reducing the query complexity of Fischlin's Prover, for a special class of Sigma protocols we can for the first time closely match a new lower bound we present.

Finally we extend Fischlin's technique so that it applies to a more general class of *strongly-sound* Sigma protocols, which includes the OR composition. We achieve this by carefully randomizing Fischlin's technique—we show that its current deterministic nature prevents its application to certain multi-witness languages.

1 Introduction

A Sigma protocol is a three move public coin proof for a language L that allows for efficient sampling of transcripts without a witness (honest-verifier zero-knowledge), and has the property that any pair of accepting conversations that share the same first message will yield a witness for the statement (two-special soundness). Sigma protocols are a useful abstraction in multiple regards, as many algebraic languages admit highly efficient sigma protocols [Sch91], compilers for more complex languages have been constructed [CDS94], and analysis of whether a protocol does indeed meet the definition of a Sigma protocol is usually straightforward.

In the many settings where a non-interactive zero-knowledge proof (NIZK) suits the network constraints, a Sigma protocol can be efficiently compiled to a NIZK in the Random Oracle model [FS87, Pas03, Fis05]. The Fiat-Shamir compiler [FS87] is the most efficient with essentially no overhead in computation or communication, however the extractor induced for the proof-of-knowledge property requires rewinding a malicious prover in order to extract a witness. This extraction technique known as "forking" the adversary is due to Pointcheval and Stern [PS96] and incurs a substantial penalty in the *tightness* of the security reduction.

Moreover while a rewinding extractor is conducive to proving sequential composition, when arbitrary concurrent composition is desired, an *online* or *straightline* extractor vastly simplifies matters. Straightline extraction refers to the notion of soundness by which the witness for a theorem can be extracted from a prover without rewinding. Early work in this area [SG02, CF01] established its benefits for composition and tight security, and that protocols which support straightline extraction require some setup such as a common random string or a random oracle. The later choice is particularly useful in more practical protocols.

Signature Aggregation. A recent application of straight-line extraction techniques is in the aggregation of Schnorr/EdDSA signatures [CGKN21]. Signature schemes based on the discrete logarithm problem alone have not traditionally been known to support aggregation methods, unlike say pairing based constructions [BLS01]. Chalkias et al. [CGKN21] construct a Sigma protocol by which one can prove knowledge of a collection of Schnorr signatures rather than transmit them naively. The Sigma protocol is compressing, as its transcript is only half the size of a naive concatenation of the signatures. Compiling this Sigma protocol to a non-interactive proof (i.e. an *aggregate signature*) via the Fiat-Shamir transformation is efficient but problematic as it incurs a quadratic security loss due to the forking lemma—doubling the size of the underlying elliptic curve (to retain the same security level as the original signature) entirely erases the compression due to aggregation. Using a straight-line extractable compiler to produce a non-interactive proof yields a tight reduction, and therefore has the scope to retain the compression of the Sigma protocol while maintaining the same security level as the signature itself.

1.1 Existing Approaches to Straight Line Extraction

Pass [Pas03] showed that the random oracle model could be used to achieve efficient and easily implementable protocols that were *straightline extractable*, deniable, and concurrently secure. The main idea in Pass is to apply a *cut and* *choose* technique to a Sigma protocol wherein a Prover commits to the transcripts of 2^ℓ invocations of the protocol with the same first message but different challenges. These commitments are implemented using a Merkle tree consisting of random oracle evaluations. The Merkle tree root is itself used as a random oracle query, and the result determines the index of the transcript that is to be decommitted to the verifier. Intuitively a prover that succeeds in this protocol must have committed to at least two accepting transcripts with probability greater than $2^{-\ell}$; these two transcripts can then be used by the extractor (without rewinding) to extract a witness due to the two-special soundness property of the original Sigma protocol. This basic unit is repeated $r = \lambda/\ell$ times to amplify the soundness to a λ-bit security level. This technique applies to any two-special sound Sigma protocol, and thus shows the universal straightline extractability for any language in NP via Blum's Hamiltonicity protocol. Unruh [Unr15] shows how to adapt this technique to construct a non-interactive zero-knowledge proof of knowledge that is secure against polynomial-time quantum adversaries[1].

The drawbacks of this approach are two-fold: first, the Prover must compute $r \cdot 2^\ell$ protocol transcripts and hash them, and second, there is large overhead in opening the leaves of the Merkle tree in each repetition of the basic unit. Concretely revealing a single leaf costs $\ell\lambda$ bits, and r leaves have to be revealed, bringing the total overhead to $r\ell\lambda = \lambda^2$ bits for the openings alone.

To partially address this inefficiency, Fischlin [Fis05] suggested a different method for achieving straightline extraction that relies on the Prover using a *proof of work* to find a suitable protocol transcript. Intuitively, the Prover must compute a protocol transcript that, for example, hashes to zero for a suitably chosen hash function. This is equivalent to 'inverting' the hash function at a fixed target, i.e. finding a pre-image x so that $H(x) = 0$. The proof of work intuitively forces the Prover compute several valid protocol transcripts (all starting with the same first message), and thus allows an extractor to find a witness simply by reading the different queries to the random oracle. This method avoids the overhead of having to commit to many protocol instances and opening only one. The main advantage of this approach is an asymptotically smaller transcript because it entirely sheds the λ^2 bits required for the Merkle tree openings, which in many situations could be the dominant asymptotic term[2].

Inadequacies in the state of the art. While the method of Fischlin achieves a lower communication complexity, it also has two drawbacks.

- **Prover Computation Overhead.** The prover must hash roughly the same number of transcripts in expectation as Pass in order to find a proof. Fischlin provides some justification as to why the Prover of any NIZKPoK with a straight-line extractor that does not program the random oracle must incur a

[1]The Unruh transformation removes the Merkle tree alltogether and thus incurs a large overhead penalty; however the aim in that work is security against quantum adversaries (which, e.g., cannot be rewound).

[2]If a single Sigma protocol transcript is of size S, then a proof by Pas03 is of size $S \cdot \frac{\lambda}{\log \lambda} + \lambda^2$. Assuming $S \in O(\lambda)$, the λ^2 Merkle opening cost dominates asymptotically

cost of $\omega(\log \lambda)$ queries made to the random oracle [Fis05, Proposition 2] however the gap between *optimal* performance and the performance of Fischlin's scheme (if there is one) remains unexplored. This aspect is particularly evident in the signature aggregation application, as the construction that Chalkias et al. obtained upon applying Fischlin's transformation suffered from a high computation cost for the prover/aggregator.

– **Limited Applicability Due To Quasi-unique Responses.** For technical reasons in their proof, Fischlin's method only applies to a subset of three-move protocols which satisfy a "quasi unique responses" property. Roughly this means that no efficient prover can output a theorem x and a, e, z, z' such that (a, e, z) and (a, e, z') are both accepting transcripts for x. This excludes Sigma protocols such as logical compositions and proof of knowledge of Pedersen commitment openings. While it is folklore that this property is not necessary for the extractor to succeed, to our knowledge it is unknown at present if this property is strictly necessary for zero-knowledge.

1.2 This Work

We advance the study of straight-line extraction in the random oracle model on the fronts of *computation cost*, as well as the *applicability of Fischlin's transform*. We make orthogonal but compatible improvements in both dimensions.

Computation Cost of Straight-Line Extraction. Our motivating application in which to improve computation cost is signature aggregation, and so we first develop our new techniques in this context and subsequently examine implications that are of more general interest. Roughly, the prover/aggregator in Chalkias et al's construction evaluates a polynomial f that encodes the signatures, in order to find points $x_i, f(x_i)$ such that $H(x_i, f(x_i)) = 0$. The computation cost can be broken into two components: the cost $\mathsf{C}_{\mathsf{qry}}$ per evaluation of f, and the *prover query complexity*, i.e. number $\mathsf{T}_{\mathsf{Agg}}$ of evaluations of f that must be hashed before a solution is found—we improve both components in this work.

– **Better $\mathsf{C}_{\mathsf{qry}}$ via Improved Polynomial Evaluation.** We make use of an $O(n^{1.5})$ polynomial evaluation algorithm that performs over an order of magnitude better than the $O(n^2)$ naive method for practically relevant parameters. After diligently searching the literature for this simple technique, we are unaware of any previous application of this observation—perhaps because it was already folklore. Nonetheless, we are the first to discover its unique suitability to straight-line extraction especially for the parameters and elliptic curve groups relevant to signature aggregation.

Theorem 1. *(Informal) For $\mathbb{Z}_q$ such that $q-1$ has a few small factors, there is an algorithm to evaluate a degree n polynomial at n points using $2n^{1.5}$ multiplications in $\mathbb{Z}_q$.*

Polynomial evaluation algorithms with significantly better asymptotic costs are known [vzGG13, BCKL21], however they are either concretely inferior in the relevant parameter ranges, or outright incompatible with commonly used signing curve groups.

– **Collision Predicates Improve Prover Query Complexity.** We replace the inversion based proof-of-work predicate with a *collision* based one. In particular the prover must now find $x_i, f(x_i)$ values such that $H(x_1, f(x_1)) = \cdots = H(x_r, f(x_r))$, which is significantly faster (up to $2\times$) than finding inversions at the same security level.

  Theorem 2. *(Informal) Let r be an integer, and H_1 and H_2 be random oracles with output lengths ℓ_1 and ℓ_2 bits respectively. Let inv and col be predicates such that $\mathsf{inv}^{H_1}(x_1, \cdots, x_r) = 1$ iff $H_1(x_1) = \cdots = H_1(x_r) = 0^{\ell_1}$, and $\mathsf{col}^{H_2}(x_1, \cdots, x_r) = 1$ iff $H_2(x_1) = \cdots = H_2(x_r)$. If r, ℓ_1, ℓ_2 are constrained so that $\Pr[\mathsf{inv}^{H_1}(1, \cdots, r)] = \Pr[\mathsf{col}^{H_2}(1, \cdots, r)]$, then finding a satisfying assignment for col^{H_2} is faster than finding one for inv^{H_1}.*

  We find that the principle of collision finding having superior combinatorics as compared to inversions more generally improves prover query complexity—Fischlin's NIZKPoK construction is sped up by $10-15\%$ by directly applying this insight. For a special class of Sigma protocols, the prover query complexity improvement due to the collision predicate idea is up to $2\times$.

– **Lower Bound on Query Complexity.** We tighten Fischlin's asymptotic lower bound on prover query complexity to obtain a concrete one under certain conditions.

  Lemma 1. *(Informal) If a NIZKPoK scheme for a hard relation with a straight-line extractor (in the non-programmable ROM) induces a verifier to make V queries to the RO for a λ-bit security level, then the prover must on average make at least $P_{\mathsf{OPT}}[V, \lambda] = (V! \cdot 2^{\lambda})^{\frac{1}{V}}$ queries in generating a proof.*

  This bound is not met by any existing constructions for non-trivial parameters. However the special class of Sigma protocols mentioned above with the collision predicate idea achieves the optimal query complexity for a range of non-trivial parameters—this also serves to inspire confidence in the tightness of the bound.

  Lemma 2. *(Informal) There is a NIZPoK for the DLog relation with a straight-line extractor (in the non-programmable ROM) where the prover makes roughly $P_{\mathsf{OPT}}[V, \lambda]$ queries on average for V up to 5, and $\lambda = 128$ onwards.*

We tighten the parameters and benchmark our improved aggregation construction, the result of which report in Table 1. We obtain up to a $200\times$ improvement in prover computation over Chalkias et al. [CGKN21] for practically relevant parameters, at the same compression rate. This makes provably secure parameters for signature aggregation accessible in many real-world settings.

Applicability of Fischlin's Transform. We revisit (and eliminate) the role of quasi-unique responses in Fischlin's transform. To our knowledge, it is folklore that the extractor does not strictly need this property, and it is unclear as to whether it is really necessary for zero-knowledge. In fact, Fischlin even suggested informally [Fis05, pg. 13] that their construction works for Sigma protocols for languages with multiple witnesses (such as logical combinations [CDS94]) where achieving quasi-unique responses appears to be simply a matter of adjusting syntax. We find this intuition to be false; in particular we show by means of an attack that *witness indistinguishability* is not preserved upon applying Fischlin's transformation to a natural Sigma protocol (i.e. logical OR composition [CDS94]) in a context that appears to be conducive to quasi-unique responses. Intuitively this stems from the deterministic nature of Fischlin's Prover which leads to a subtle trace of the witness in compiled proofs.

Theorem 3. *(Informal) Fischlin's transformation does not preserve Witness Indistinguishability when applied to the Sigma protocol to prove knowledge of one of two Discrete Logarithms.*

Through a new proof, we show how a simple randomization of Fischlin's method allows it to be safely applied to any *strong* special sound Sigma protocol, where strong special soundness—which we introduce—is a simpler property of a Sigma protocol and does not require context-specific reasoning (i.e. dependence on setup parameters) like quasi-unique responses. Requiring strong special soundness rather than quasi-unique responses strictly increases the applicability of Fischlin's transform.

Theorem 4. *(Informal) Any Strong Special Sound Sigma protocol can be compiled to a straight-line extractable NIZKPoK in the ROM, with the same computation and bandwidth efficiency as applying Fischlin's transformation.*

Our attack on WI appears to uncover an interesting aspect of the role of randomness in straight-line extractable zero-knowledge proofs. Pass' transformation is randomized (due to its use of a commitment scheme), and naively derandomizing it would result in a similar attack. An interesting and natural question for future work would be to identify the class of languages for which "well-behaved" transforms that make black-box use of an underlying zero-knowledge protocol and compile them into a straightline extractable one in the random oracle model *must* be randomized.

We therefore demonstrate conclusively that one can do better than generic cut-and-choose (i.e. Pass [Pas03]) for straight-line extractable NIZKs for many algebraic languages in the random oracle model. Such languages include logical combinations [CDS94], openings to Pedersen commitments, among many others that are used in non-trivial cryptographic systems such as the anonymous survey protocol [HMPs14].

2 Our Techniques

We first recall Fischlin's transformation in order to build intuition for our techniques. The base unit of the transformation is the following: for the instance x, the Prover computes a first message a of the Sigma protocol, and finds second

and third messages e, z such that $V_x(a, e, z) = 1$ and $H(a, e, z) = 0$[3] for some ℓ-bit hash function H, where $\ell \in O(\log \lambda)$. This is done by starting with $e = 0$ (and the corresponding response z) and computing $H(a, e, z)$, iteratively stepping through e, z candidates which verify until the first e, z pair is found such that $H(a, e, z)$ evaluates to the all-zero string 0. An adversarial prover is able to produce (a, e, z) such that $H(a, e, z) = 0$ without querying more than one transcript to H only if it gets lucky with its first query, which happens with probability $2^{-\ell}$. This base unit is therefore repeated $r = \lambda/\ell$ times to achieve λ bits of soundness; specifically, to bind these instances together and prevent independent grinding, all of the a messages for the repeated instances are incorporated into the input to the hash function. For example, for 2 repetitions, the Prover must produce $a_1, a_2, e_1, e_2, z_2, z_2$ such that $H(a_1, a_2, e_1, z_1) = 0$ and $H(a_1, a_2, e_2, z_2) = 0$ and of course $V_x(a_1, e_1, z_1) = 1$ and $V_x(a_2, e_2, z_2) = 1$.
Prover Query Complexity. We refer to the (expected) number of queries that the prover makes to the random oracle as the *prover query complexity.* For instance, the Prover query complexity of Fischlin's construction as described above is $r \cdot 2^\ell = r \cdot 2^{\frac{\lambda}{r}}$, which implies a tradeoff between r (which governs proof size and verification cost) and the query complexity. We develop the study of prover query complexity in this work, as part of our study on the computation cost of straight-line extraction.

Fischlin presents a variant of their transformation where the verifier accepts 'near' inversions. This is is not relevant for our work, as discussed in Appendix K.

2.1 Schnorr/EdDSA Signature Aggregation and Computation Cost

Our motivating practical application is that of aggregating Schnorr/EdDSA signatures with tight security. Chalkias et al. construct a compressing Sigma protocol to prove knowledge of n Schnorr signatures, to which they apply Fischlin's transformation to obtain a non-interactive proof. As mentioned earlier, their scheme is roughly to have the prover encode the n signatures as the coefficients of a degree $n - 1$ polynomial f, and output a proof consisting of $(x_1, f(x_1)), \cdots, (x_r, f(x_r))$ such that each $H(x_i, f(x_i)) = 0$. They find producing such a proof to be computationally intensive, for instance over a minute to aggregate even hundreds of signatures at a 53% compression ratio[4] which induces a prohibitively high latency for many applications.

Faster Polynomial Evaluation with Curve25519. If we denote the prover query complexity as $\mathsf{T}_{\mathsf{Agg}}$, the prover must evaluate f at $\mathsf{T}_{\mathsf{Agg}}$ points. The first aspect of the prover's computation cost that we improve is the cost of producing $\mathsf{T}_{\mathsf{Agg}}$ evaluations of f. The naive method to evaluate a degree n polynomial costs n multiplications in $\mathbb{Z}_q$, meaning that the prover performs $n\mathsf{T}_{\mathsf{Agg}}$ multiplications. The Fast Fourier Transform (FFT) is a well-known method to speed up polynomial evaluation to $O(\mathsf{T}_{\mathsf{Agg}} \log n)$, and is used in straight-line extractable proofs

[3]The instance x is also included in the hash, but omitted for clarity.

[4]The r parameter governs a tradeoff between query complexity and compression ratio—a lower ratio is better compression, and 50% is the lowest possible [CGKN21]

for general statements [AHIV17, BCR$^+$19]. Unfortunately the most common variant of Schnorr in practice—EdDSA—uses Curve25519, whose corresponding base field does not have a sufficiently large multiplicative subgroup to support the FFT.

We instead make use of a method (Theorem 5) by which we can derive a randomly chosen polynomial h of degree $k < n$, such that it agrees with f on k points. Deriving h costs n multiplications, and evaluating h at each point costs k multiplications, which means that we can obtain k evaluations of f at roughly $n + k^2$ cost rather than the naive nk—a substantial improvement when $k \approx \sqrt{n}$. A prerequisite to use this method is that $\mathbb{Z}_q$ must have a multiplicative subgroup of size k, however unlike the FFT this method is *randomized* and can be invoked multiple times using the same subgroup, with negligible probability of producing redundant evaluations (Corollary 5). Curve25519 has multiplicative subgroups of size up to 132, which provides nearly optimal values of $k \approx \sqrt{n}$ for the parameters relevant to signature aggregation (n up to 2^{12} or so).

The intuition for the method is as follows: we decompose f into k different degree n/k polynomials f_i such that $f(x) = \sum\limits_{i\in[k]} x^i \cdot f_i(x^k)$. We then sample $\alpha \leftarrow \mathbb{Z}_q$, and derive $h(x) = \sum_{i\in[k]} x^i \cdot f_i(\alpha^k)$. Observe that for any primitive k^{th} root of unity $\omega \in \mathbb{Z}_q$ and for any $j \in [k]$, it holds that $f_i((\alpha\omega^j)^k) = f_i(\alpha^k)$ for every f_i. Consequently, h agrees with f on the points $\{\alpha \cdot \omega^j\}_{j\in[k]}$.

Better Prover Query Complexity via Collisions. We change the underlying proof of work predicate to that of finding collisions rather than inversions of the hash function. In particular, the prover outputs a proof consisting of $(x_1, f(x_1)), \cdots, (x_r, f(x_r))$ such that $H(x_1, f(x_1)) = \cdots = H(x_r, f(x_r))$. For the same r and soundness level (note that ℓ has to be adjusted), analytical estimates on multicollision running times [vM39, Pre93] place the query complexity $\mathsf{T}_{\mathsf{Agg}}$ induced by this collision predicate at up to $2\times$ better than that of inversions.

Combining these improvements (along with a tighter analysis that makes the proof of work easier by 2–8$\times$) yields an improvement of a *factor of 70$\times$–200$\times$* for the most aggressive compression settings reported in prior work (see Table 1).

Collisions Improve Fischlin's NIZK. We generalize this principle and apply it to Fischlin's transform for NIZKPoKs as well, by using a collision pair base unit as a drop-in replacement for inversion base units. In particular, a collision pair base unit instructs the prover to find pairs of accepting Sigma protocol transcripts (a, e, z) and (a', e', z') such that $H((a, a'), e, z) = H((a, a'), e', z')$. A forgery requires a collision within the first two queries to the random oracle, which happens with probability $2^{-\ell}$ for an ℓ-bit hash function. This serves as a drop-in replacement for a pair of inversion base units that achieve a combined ℓ bits of soundness. Analyzing the query complexity is difficult as this is a *chosen prefix* collision [SLdW07], and so we test the new proof-of-work problem empirically and observe an $11\% - 15\%$ improvement for common practical parameters.

A Query Complexity Lower Bound. We tighten Fischlin's asymptotic lower bound on hash queries for a NIZK with a non-programming extractor [Fis05, Proposition 2] to derive Lemma 3 and subsequently Corollary 1, which characterizes the optimal prover query complexity $P_{\mathsf{OPT}}[V]$ for a given verifier query complexity V. Intuitively if the prover makes P queries of which V are checked by the verifier, $\binom{P}{V}$ must be at least 2^λ to achieve a $2^{-\lambda}$ soundness error. We note that this bound applies to schemes with perfect completeness, and while Lemma 3 is sufficiently general to derive a strict bound for probabilistic schemes, P_{OPT} serves as a useful reference point, and will be the quantity that we refer to as 'optimal' prover query complexity.

We show via Claim 6 that the expected query complexity of Fischlin's construction is never better than $\sqrt{2}P_{\mathsf{OPT}}$ in any non-trivial parameter regime.

We note that Pass' transform (and equivalently Unruh's transform[5] [Unr15]) has a (strict) query complexity that is twice that of the expected prover complexity of Fischlin in any non-trivial parameter regime, and so we do not consider Pass/Unruh going forward.

Achieving P_{OPT}. For a special class of r-simulatable Sigma protocols (i.e. r transcripts are simulatable at once) we show that a NIZKPoK with prover query complexity P_{OPT} can be achieved for a range of non-trivial parameters. We construct this NIZK by applying a multicollision predicate akin to our signature aggregation construction, where the prover must produce transcripts $(\boldsymbol{a}, e_1, z_1), \cdots, (\boldsymbol{a}, e_r, z_r)$ such that $H(\boldsymbol{a}, e_1, z_1) = \cdots = H(\boldsymbol{a}, e_r, z_r)$. We make use of classic results on multicollision complexities [vM39, Pre93] to analyze the expected prover query complexities. Note that this transform is limited in applicability—we show how Schnorr's proof of knowledge of discrete logarithm can be made r-simulatable, but leave it as an interesting problem for future work to expand the scope of this transform.

Wider Application of Our Techniques. Our techniques for improving the computation cost of Signature Aggregation can be applied directly to the threshold cryptography context for the same signature schemes. For example, the most expensive component of Distributed Key Generation (DKG) for the canonical (t, n) threshold Schnorr scheme [Lin22, Protocol 6.1] is the NIZKPoK to prove knowledge of a polynomial that is committed in the curve group. The instantiation for this NIZKPoK suggested by Lindell [Lin22] is the batch PoK of Discrete Log [GLSY04] compiled to a NIZK using Fischlin's transform—i.e. exactly the same as EdDSA signature aggregation (with an extra blinding factor). Consequently, DKG for (t, n) EdDSA can benefit from roughly the same speedup that we report for signature aggregation. We briefly discuss other applications related to threshold cryptography with Curve25519 and secp256k1 in Appendix I.

[5] For the purpose of prover query complexity, Unruh's transform can be seen as Pass' transform without the Merkle trees to reduce the number of repetitions of the base Sigma protocol.

Is better (eg. sublinear) aggregation possible? Unfortunately, any aggregation technique that is blackbox in the Schnorr hash function (such as ours) is inherently limited to a 50% aggregation rate [CGKN21, Theorem 9]. The only known aggregation methods that are non-blackbox in the hash function involve expressing the hash function as an arithmetic circuit and invoking a generic SNARK, which is much too slow for standard hash functions like SHA2—on the order of 10s–100s of milliseconds per signature being aggregated, as opposed to our technique which can process each signature in a fraction of a millisecond.

2.2 Extending the Applicability of Fischlin's Transform

A technicality in Fischlin's transformation arises when it is possible for the Prover to iterate through verifying transcripts *without* having to change the challenge message e. Consider a Sigma protocol that permits an adversary without a witness to sample $(a, e), z_1, z_2, \cdots z_n$ such that each (a, e, z_i) is a valid transcript. Applying Fischlin's transformation will not produce a sound NIZK because an adversary can simply step through $H(a, e, z_1), \cdots, H(a, e, z_n)$ to find a pre-image of 0 whereas an extractor may not be able to extract a witness from this sequence of queries because they do not satisfy the requirements for 2-special soundness.

Although it is folklore that many Sigma protocols allow for extraction even given accepting transcripts $(a, e, z_1), (a, e, z_2)$ (examples include the famous logical OR composition [CDS94], opening of a Pedersen commitment, etc. for which this is simply a matter of adjusting syntax), Fischlin's transform only applies to protocols that support a *quasi-unique response* property, given below.

Definition 1. *[Fis05, Definition 1] A Sigma protocol has quasi-unique responses if for every PPT algorithm $\mathcal{A}$, for system parameter k and $(x, a, e, z_1, z_2) \leftarrow \mathcal{A}(k)$, we have as a function of k that the following probability is negligible:*

$$\Pr\left[V_x(a, e, z_1) = V_x(a, e, z_2) = 1 \wedge z_1 \neq z_2\right]$$

Here the system parameter k can be an arbitrarily structured object sampled according to some distribution, for eg. an RSA modulus or $h \in \mathbb{G}$ such that $\mathsf{DLog}_g(h)$ is unknown, as required in Okamoto's identification protocols [Oka93].

Interestingly, Fischlin's proof also uses this property to argue *zero-knowledge*. It is less obvious as to why quasi-unique responses is relevant for this purpose. In the absence of an explicit attack on the zero-knowledge property when quasi-unique responses does not hold, one may even conclude that it is simply an artefact leveraged to prove the simulation secure.

We show this intuition to be *false*. In particular, we construct an explicit attack on *Witness Indistinguishability* when Fischlin's transformation is applied to a common Sigma protocol for a language with two witnesses. This attack is the result of combining two facts:

– **Fischlin's Transformation is Deterministic.** Once the Sigma protocol first messages have been sampled, the prover's algorithm is deterministic.

– **Some Sigma Protocols Reveal the Prover's Randomness.** In particular Schnorr's proof of knowledge of discrete logarithm reveals a linear combination of the witness and the prover's randomness—knowledge of the witness therefore allows an attacker to reconstruct the prover's randomness.

It is therefore possible for an attacker to *retrieve* the prover's random tape when given a Fischlin-compiled Schnorr proof, and *replay* the prover's steps and reconstruct the proof string. To demonstrate why this is problematic, we examine the effect of this retrieve-and-replay strategy given a Fischlin-compiled proof of knowledge of one-out-of-two discrete logarithms [CDS94]. In particular if a prover uses one of x_0, x_1 to prove knowledge of $x_0 \cdot G \vee x_1 \cdot G$, an attacker with knowledge of say x_0 can execute the retrieve-and-replay strategy to test if x_0 was indeed used in producing the proof string. We show that if the attacker uses x_0 to execute this strategy on a proof that was actually produced using x_1, there is a non-negligible chance that the proof string that the attacker reconstructs will be *different* from the given one (as opposed to a proof string produced using x_0, which always matches the reconstruction). Intuitively, this is because the proof string serves as a record of how many Sigma protocol transcripts had to be hashed before a solution to the proof of work was found—recomputing the proof using a different witness might result in finding a solution by hashing fewer transcripts.

We note that our attack runs entirely in the random oracle model and does not exploit concrete instantiations of the hash function, unlike previous work that studies the concrete instantiability of Fischlin's transform [ABGR13].

Randomization Fixes the Problem. We formalize a notion of *strong special soundness* to capture the folklore notion that accepting transcripts of the form (a, e, z_1),(a, e, z_2) yield a witness. This is a subtle change in the definition of special soundness; luckily many natural Sigma protocols (including those with multiple witnesses for which Fischlin's transformation is shown not to work as above) satisfy this property, including every regular special sound Sigma protocol that supports quasi-unique responses.

We then show how to randomize Fischlin's transformation to erase all traces of the witness from the compiled proof strings, and prove that zero-knowledge is guaranteed unconditionally for any strong special sound Sigma protocol. Intuitively this is achieved by having the prover step randomly through the challenge space to find a solution to the proof of work, and this form of randomization is directly compatible with a collision-based proof of work.

3 Preliminaries

A Sigma protocol is a three move public coin protocol between a prover $P_\Sigma(x, w)$ and a verifier $V_\Sigma(x)$. We further use $(\mathsf{state}, a) \leftarrow P_{\Sigma,a}(x, w)$ to denote the internal state and first message output by P_Σ respectively. Subsequently $z \leftarrow P_{\Sigma,z}(\mathsf{state}, e)$ denotes the response of P_Σ upon being given the previously produced internal state, and the verifier's challenge respectively. We recall the exact properties (Completeness, two-special soundness, and honest verifier ZK) in Appendix A. We also recall the definition of straightline extraction as given by Pass in Appendix A. As we have already recalled Fischlin's transform informally earlier, we defer a formal description to Figure 8 in Appendix A.

4 Signature Aggregation With a Tight Reduction

We first explore aggregating EdDSA signatures as a motivating practical application. In particular, we are focused on obtaining a tight reduction for the unforgeability of the aggregate signature to that of the underlying signatures, which at its core is a problem of straight-line extraction. We briefly recap the work of Chalkias et al. [CGKN21] who recently constructed an aggregation scheme for Schnorr (of which EdDSA is a widely used instantiation) that achieves factor 2 compression in the random oracle model.

Sigma Protocol and Non-Interactive Compilation. Their first step is to construct an n-special sound Sigma protocol to prove knowledge of n Schnorr signatures. For signatures instantiated over a field of order q, the transcript of the Sigma protocol is of size $(n+1)|q|$ bits, as opposed to naive transmission of n signatures which would require $2n|q|$ bits.

They subsequently apply Fischlin's transformation to their Sigma protocol in order to construct a non-interactive proof of knowledge that enjoys a tight reduction (yielding *provably secure* parameters, unlike Fiat-Shamir) while achieving a compression rate that can be arbitrarily close to 2. However the proximity to factor 2 compression comes at the expense of prover computation.

Concretely as per [CGKN21, Figure 2] aggregating EdDSA[6] signatures with Fischlin's transformation incurs an amortized cost of 4.2ms per signature when compressing by a factor of 1.33, and 39.7ms for factor 1.81 compression. This is multiple orders of magnitude slower than the Fiat-Shamir compiled proof (which incurs a fraction of a microsecond per signature on the same hardware) and processing even hundreds of signatures at once becomes prohibitively expensive.

Faster Straight-Line Extraction. In this section we will develop the tools to substantially speed up the aggregation of EdDSA signatures with straight-line extraction in the random oracle model. Our improved aggregation algorithm is up to 200× faster for practically relevant parameters, and potentially within the performance envelope of real-world applications.

4.1 Recap of [CGKN21] Construction

Schnorr Compression Sigma Protocol [CGKN21]. Recall that a Schnorr signature on a message $m \in \{0,1\}^*$ under a public key $\mathsf{pk} \in \mathbb{G}$ consists of a nonce $R \in \mathbb{G}$ and a scalar $s \in \mathbb{Z}_q$ such that $z \cdot G = H_{\mathsf{Sch}}(\mathsf{pk}, R, m) \cdot \mathsf{pk} + R$. Informally the Sigma protocol is the combination of two ideas:

[6]We use EdDSA to refer to Ed25519 [BDL+12] in particular, which is believed to instantiate a 128-bit security level.

1. Once m, pk, R are determined there is a unique $s \in \mathbb{Z}_q$ that 'completes' the signature, and this is the discrete logarithm of the publicly computable group element $S = H_{\mathsf{Sch}}(\mathsf{pk}, R, m) \cdot \mathsf{pk} + R$. Proving knowledge of the discrete logarithm of S is therefore equivalent to proving knowledge of the missing component of the signature.
2. There is an n-special sound Sigma protocol to simultaneously prove knowledge of the discrete logarithms of n public group elements at the same bandwidth cost of a single PoK of DLog [GLSY04].

Upon fixing n messages m_i and signatures $(R_i, s_i)_{i \in [n]}$ under respective public keys pk_i, the prover is given a challenge $e \in \mathbb{Z}_q$, to which it computes the response $z = \sum_{i \in [n]} s_i \cdot e^i$. The verifier is given the statement $(\mathsf{pk}_i, R_i, m_i)_{i \in [n]}$, challenge e, and the putative Prover's response z, and validates them by verifying that $z \cdot G = \sum_{i \in [n]} e^i \cdot (H_{\mathsf{Sch}}(\mathsf{pk}_i, R_i, m_i) \cdot \mathsf{pk} + R_i)$.

Applying Fischlin's Transformation. Chalkias et al. directly apply Fischlin's transformation to the above Sigma protocol to obtain a non-interactive proof. In particular, a 'base unit' of the proof is a challenge-response pair (e_j, z_j) such that $H(\mathsf{prefix}, e_j, z_j) = 0$ where H is an ℓ-bit random oracle, and this unit is repeated r times in order to achieve a λ-bit soundness level. These parameters are set so that a successful prover must query the random oracle with at least n accepting transcripts except with probability $2^{-\lambda}$.

Breaking down the cost. We can express the prover's computation cost in producing a proof as $\mathsf{T}_{\mathsf{Agg}} \cdot \mathsf{C}_{\mathsf{qry}}$, where $\mathsf{T}_{\mathsf{Agg}}$ is the prover query complexity, i.e. the number of (e, z) values the prover queries to the random oracle, and $\mathsf{C}_{\mathsf{qry}}$ is the cost of generating each (e, z) value. We discuss below how to improve on both of these dimensions.

4.2 Reducing $\mathsf{C}_{\mathsf{qry}}$ via Improved Polynomial Evaluation

The efficiency of polynomial evaluation algorithms is usually tied to the degree of the polynomial being evaluated. In our case, the degree of the polynomial corresponds to the number of signatures being aggregated. As the signature batch size can be small in practice (eg. number of transactions in a block, which is around 2000 for Bitcoin [Blo]) asymptotically efficient polynomial evaluation algorithms [vzGG13, BCKL21] may not be relevant to our setting.

Theorem 5. *Given a prime q, degree n polynomial $f \in \mathbb{Z}_q[X]$, and primitive k^{th} root of unity $\omega \in \mathbb{Z}_q$, Algorithm* PolyEval *outputs a list of k distinct points that lie on f at a cost of $k^2 + n + 2\log k$ multiplications and $k(k-1) + n$ additions in $\mathbb{Z}_q$.*

We defer the proof of this theorem to Appendix F.

While this is a significant improvement over the naive polynomial evaluation algorithm (which requires nk $\mathbb{Z}_q$ multiplications), in our application we need to evaluate f over a large set of points, and PolyEval only produces a batch of k

Algorithm PolyEval

This algorithm is parameterized by a finite field $\mathbb{Z}_q$ where q is prime, a primitive k^{th} root of unity $\omega \in \mathbb{Z}_q$, and a degree n polynomial $f \in \mathbb{Z}_q[X]$. For simplicity we assume that k divides n. The output of this algorithm is a list of points $\{(x_i, f(x_i))\}_{i\in[k]}$.

```
PolyEval$(q, k, f, n)$:
1. Parse the coefficients of $f$, with $c_i$ as the coefficient of $x^i$
2. For each $i \in [0..k-1]$, define polynomial $f_i(x) = \sum_{j\in[0..n/k]} x^j \cdot c_{jk+i}$
3. Sample $\alpha \leftarrow \mathbb{Z}_q^*$ and for each $i \in [0..k-1]$ compute $\vec{\alpha}_i = f_i(\alpha^k)$
4. Define the degree $k-1$ polynomial $h(x) = \sum_{i\in[0..k-1]} \vec{\alpha}_i x^i$
5. Let points denote the (initially empty) list of output points
6. For each $i \in [0..k-1]$, append $\left(\alpha\cdot\omega^i, h(\alpha\cdot\omega^i)\right)$ to points
7. Output points
```

Fig. 1: Improved Polynomial Evaluation

evaluations. A simple extension to produce a batch of say $m \cdot k$ evaluations is to invoke PolyEval m times independently. However it is possible that there may be some redundancy across the multiple evaluations, i.e. independent instances may evaluate f at the same point. We show via Lemma 6 and Corollary 5 in Appendix F that for the parameters relevant to our setting, the probability of there being any redundancy is negligible.

Efficiency. As per Theorem 5, PolyEval achieves the best improvement when $k \approx \sqrt{n}$. In this case, evaluating a degree n polynomial at $\sqrt{n}$ points costs roughly $2n$ multiplications, which is a factor $\sqrt{n}/2$ improvement over the naive method. This improvement is subject to the availability of appropriate k in the field in question. The setting that we consider in this paper involves the EdDSA signature scheme, which uses Curve25519 [Ber06], which in turn is of order q such that $q-1$ is divisible by 4, 3, and 11. Given that we are interested in $n < 2^{12}$ or so, we are able to find a nearly optimal k for for any value of n in our range. We plot the improvement achieved by PolyEval in Figure 2.

Comparison with ECFFT. The very recent work of Ben-Sasson et al. [BCKL21] introduces a method to enable an FFT-like recursive evaluation of a polynomial in any arbitrary $\mathbb{Z}_q$, by using isogenies of elliptic curves. Their algorithm achieves impressive asymptotic as well as concrete performance in the preprocessing model, and can be applied to our setting. In particular, their $O(n\log^2(n))$ complexity is asymptotically superior to our $O(n^{1.5})$ PolyEval algorithm. However for our parameter range, we find our PolyEval algorithm to perform better, as we show in Figure 3.

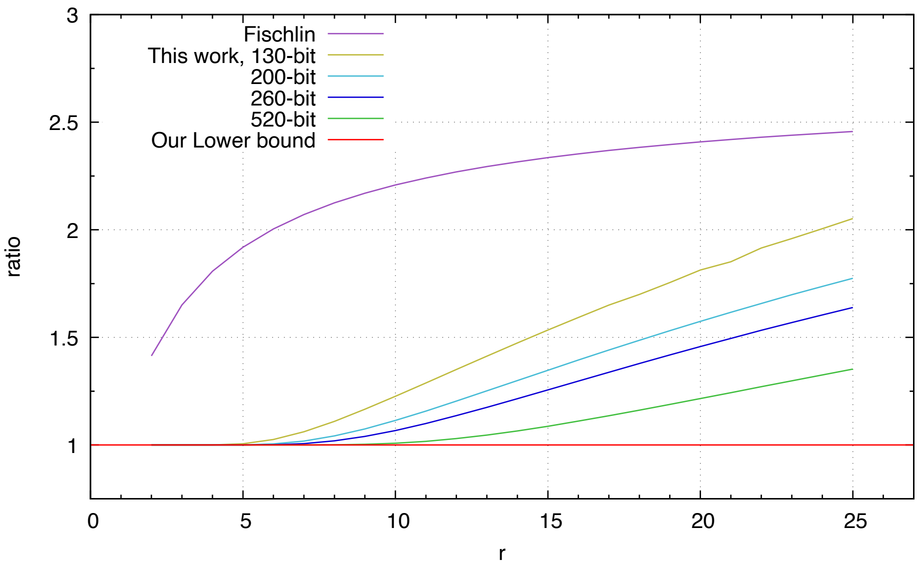

Fig. 2: This graph plots the computation cost of evaluating a polynomial of degree n up to 2^{12} at n points in $\mathbb{Z}_q$, where q is the order of the elliptic curve Curve25519 used for EdDSA. The cost is derived analytically.

Fig. 3: This graph plots the factor improvement over the naive method, in evaluating a polynomial of degree n up to 2^{14} at n points in $\mathbb{Z}_q$, where q is the order of the BN-254 elliptic curve. The improvement factor for ECFFT is taken from a public Rust implementation [wbo]. We did not re-implement PolyEval for this curve, however our Rust implementation for Ed25519 is faithful to our analytical estimate, and so we derived the improvement factor for PolyEval analytically.

4.3 Improving Prover Query Complexity $\mathsf{T}_{\mathsf{Agg}}$

First we note that tightening the parameters of [CGKN21] via a better analysis yields an improvement of 2 to 8× in the hardness setting for the proof-of-work problem. Intuitively this is because of Chalkias et al.'s direct application of Fischlin's transform by repeating a base unit sufficiently many times for the desired soundness level, whereas one can prove better parameters by directly analyzing the final construction, i.e. the event that a malicious prover finds r inversions within n queries.

Our idea. We change the underlying 'proof of work problem' solved by the prover from finding r inversions to finding an r-collision. In particular the prover now searches for $(e_j, z_j)_{j\in[r]}$ such that $H(\mathsf{prefix}, e_1, z_1) = \cdots = H(\mathsf{prefix}, e_r, z_r)$, where H is a random oracle with output bit length $\ell \geq (\lambda + r\log_2(n) - \log(r!))/(r-1)$. This yields a ≈ 1.5 to $2\times$ improvement in $\mathsf{T}_{\mathsf{Agg}}$ corresponding to the ratio of the costs of finding an r-collision to that of finding r inversions at the same security level (even with the improved analysis).

We give the full protocol and justify its parameterization below. However we defer a more precise analytical justification of why finding an r-collision is faster than finding an equivalent number of inversions at the same security level to Section 5.3, and concrete query complexity improvements to Table 3 in Appendix H.

Caveat: Memory Complexity. We note that keeping track of collisions consumes more memory—$O(\mathsf{T}_{\mathsf{Agg}})$—than the inversion construction which only needs $O(\lambda)$. In practice, however, this is quite a small amount (up to 30MB for benchmarked parameters), as shown in Table 4 in Appendix H.

Further Applications. The superior combinatorial characteristics of the collision problem over the inversion problem has interesting implications for the computation complexity of straight-line extraction even in the zero-knowledge setting. In Sections 5.1 and 5.3, we show how to improve the prover's query complexity when compiling *any* standard Sigma protocol to a NIZKPoK by $10-15\%$, and for some special Sigma protocols by up to a factor of 2. The latter is particularly significant as it matches a new lower bound that we prove.

4.4 Putting It Together – Improved EdDSA Aggregation

We combine our improvements to $\mathsf{T}_{\mathsf{Agg}}$ and $\mathsf{C}_{\mathsf{qry}}$ to obtain an EdDSA signature aggregation algorithm π_{Aggr} with substantially improved prover computation complexity, which we give below in Figure 4. We further justify its performance improvements with our benchmarks in Table 1. We postpone the security theorem to Appendix G.

5 Applying the Collision Predicate to NIZKPoK

We apply the principle of replacing hash inversions in Fischlin's transformation with hash collisions to the original NIZKPoK transform, and observe improved prover query complexity in this setting as well. We begin by considering the hash collision predicate as a *drop-in replacement* to any Sigma protocol for which Fischlin's transformation can be applied, and observe an $11-15\%$ improvement in the prover's query complexity.

To our knowledge this is the best query complexity achieved for NIZKs so far, however a natural question is to ask to what extent such techniques can be extended. To this end, we show a lower bound on the query complexity of *any* NIZK that has a straight-line non-programming extractor in Section 5.2. We find that Fischlin's construction (which is the most query efficient straight-line extractable scheme) never meets this lower bound for any non-trivial parameters.

We show in Section 5.3 that it is indeed feasible to meet this lower bound for some non-trivial parameters, by means of a new transformation based on our

Protocol π_{Aggr}

The prover P and verifier V are both given the public instance $(\mathsf{pk}_i, m_i, R_i)_{i\in[n]} \in (\mathbb{G} \times \{0,1\}^* \times \mathbb{G})^n$ while the prover also has witness $(s_i)_{i\in[n]} \in \mathbb{Z}_q^n$ for the statement $s_i \cdot G = H_{\mathsf{Sch}}(\mathsf{pk}_i, R_i, m_i) \cdot \mathsf{pk}_i + R_i \ \forall i \in [n]$. Both parties have access to an ℓ-bit Random Oracle $H : \{0,1\}^* \mapsto \{0,1\}^\ell$ where $\ell \geq (\lambda + r\log_2(n) - \log_2(r!))/(r-1)$.

```
$\mathsf{P}^H((\mathsf{pk}_i, m_i, R_i, s_i)_{i\in[n]})$:
1. Find $k$ closest to $\sqrt{n}$ such that $k \mid q-1$
2. Set $\boldsymbol{a} = (\mathsf{pk}_i, m_i, R_i)_{i\in[n]}$, and define polynomial $f(x) = \sum_{i\in[n]} x^i \cdot s_i$
3. Initialize $\mathcal{Z} = \emptyset$ and do the following until an output is produced:
   (a) Obtain $\mathsf{points} \leftarrow \mathsf{PolyEval}(q, k, f, n)$ and append each $(e, z) \in \mathsf{points}$ to $\mathcal{Z}$
   (b) If $\exists (e_1, z_1), (e_2, z_2), \cdots, (e_r, z_r) \in \mathcal{Z}$ such that
           $$H(\boldsymbol{a}, e_1, z_1) = H(\boldsymbol{a}, e_2, z_2) = \cdots = H(\boldsymbol{a}, e_r, z_r)$$
       then set $\boldsymbol{e} = (e_i)_{i\in[r]}$ and $(z_i)_{i\in[r]}$ and output $\pi = (\boldsymbol{a}, \boldsymbol{e}, \boldsymbol{z})$

$\mathsf{V}^H((\mathsf{pk}_i, m_i, R_i)_{i\in[n]}, \pi)$:
1. Parse $(\boldsymbol{a}, \boldsymbol{e}, \boldsymbol{z}) = \pi$, and $(e_i)_{i\in[r]} = \boldsymbol{e}$, and $(z_i)_{i\in[r]} = \boldsymbol{z}$.
2. Check that $H(\boldsymbol{a}, e_1, z_1) = H(\boldsymbol{a}, e_2, z_2) = \cdots = H(\boldsymbol{a}, e_r, z_r)$
3. For each $i \in [n]$, compute $S_i = H_{\mathsf{Sch}}(\mathsf{pk}, R, m) \cdot \mathsf{pk} + R$
4. For each $i \in [r]$, check that $z_i \cdot G = \sum_{i\in[n]} e^i \cdot S_i$, aborting with output 0 if not
5. Accept by outputting 1
```

Fig. 4: Collision Based Aggregation of n Signatures

n	r	Chalkias et al.		Our work		Improvement
		AggVer(ms)	AggSign	AggVer(ms)	AggSign	
512	16	137	167 ± 13.0 s	134	2.2 ± 0.07 s	76x
1024	32	485	85.5 ± 4.8 s	452 ± 6	350 ± 10 ms	244x
256	16	78	40.6 ± 2.8 s	72	901 ± 36 ms	45x
512	32	258	20.1 ± 1.4 s	255	136 ± 3 ms	147x
128	16	43	9.9 ± 0.74 s	42	363 ± 8 ms	27x
256	32	147	5.5 ± 0.31 s	143	54 ± 1 ms	101x
32	8	5.7	84.2 ± 11.6 s	5.6	7.8 ± 0.5s	11x
64	16	21	2.9 ± 0.25 s	23	78 ± 1 ms	37x
128	32	80	1.4 ± 0.08 s	84.5	20 ms	70x

Table 1: Comparing the computation cost for aggregation and aggregate-verification of n Ed25519 signatures with SHA-256 hash function used for H_1 on the same parameters from [CGKN21]. The benchmarks were run using the publically available code for [CGKN21], and a new Rust implementation of our method and the Criterion rust framework; times show a 95% confidence interval over at least 30 runs on one Intel i7-10710U core running at 3.9Ghz with 32 Gb of memory. Intervals are omitted when less than 1ms. While the aggregation methods can easily be parallelized, each of these benchmarks only use 1-core to properly compare against the implementation from [CGKN21]. The best compression ratios are achieved on the first row at roughly 53%; the last row in the table achieves the worst ratio around 75%. Both constructions have nearly the same bit size, with [CGKN21] slightly better due to smaller sized polynomial evaluation points—the difference is around 1.5% at the better compression rates.

collision predicate. Unfortunately this transformation only applies to a special class of Sigma protocols that have an r-simulatability property. We show in Appendix D how to construct such a Sigma protocol by extending Schnorr's proof of knowledge of discrete logarithm.

5.1 Unconditionally Improving Fischlin's Query Complexity

Recall that the prover in Fischlin's transformation is required to invert a fixed target of the random oracle. In particular, a proof consists of a base unit where the prover is required to find a Sigma protocol transcript (a, e, z) such that $H(\mathsf{prefix}, a, e, z) = 0^\ell$, and this unit is repeated r times to achieve $\lambda = r \cdot \ell$ bits of security. We can replace this inversion based unit by a collision based one as follows: the prover is required to find a pair of independent transcripts (a_1, e_1, z_1) and (a_2, e_2, z_2) such that $H(\mathsf{prefix}, a_1, e_1, z_1) = H(\mathsf{prefix}, a_2, e_2, z_2)$. Note that just as in the case of Fischlin, prefix includes a_1, a_2 to prevent trivial attacks. Additionally, the output length of the hash function is 2ℓ, i.e. doubled as compared to the inversion predicate.

Security. Upon fixing prefix, a prover is successful in finding an accepting pair (a_1, e_1, z_1) and (a_2, e_2, z_2) in their first attempt with probability no more than $2^{-2\ell}$. Repeating this base unit $r/2$ times achieves security $2\ell \cdot r/2 = \lambda$ bits.

Efficiency. A base unit of the collision based construction is equivalent to two base units of the inversion construction; in both cases two Sigma protocol transcripts are transmitted, and they achieve 2ℓ bits of security. With regards to computation cost, both constructions have the same cost per query made to the random oracle (i.e. computing a fresh Sigma protocol response), and therefore the difference comes down to the number of queries made per proof, i.e. the prover query complexity.

What query complexity does this induce? Consider $\mathcal{Z}_1, \mathcal{Z}_2$ to be domains from which (e_1, z_1) and (e_2, z_2) are drawn respectively, and observe that $\mathcal{Z}_1, \mathcal{Z}_2$ are entirely disjoint when $a_1 \neq a_2$. If we consider $(\mathsf{prefix}, a_1, e_1, z_1)$ and $(\mathsf{prefix}, a_2, e_2, z_2)$ to be the 'left' and 'right' halves of the collision respectively, this means that any given $(\mathsf{prefix}, a_i, e_i, z_i)$ can be a candidate pre-image for either the left or right half, but not both. This is because any given e_i, z_i can be a verifying transcript with at most one of a_1 or a_2. This task therefore becomes that of finding a *chosen prefix* collision [SLdW07]. The combinatorics of chosen prefix collisions are considerably more complex to analyze than regular collisions, making the derivation of the exact query complexity of the above construction difficult. We instead measure the query complexity induced by this predicate empirically, and report on the results in Table 2.

As our experiments show, this chosen prefix collision predicate works for the exact same Sigma protocols as Fischlin's transformation, and improves on its query complexity. A natural question for future work is if we can obtain further improvements by considering multicollisions rather than pairs of collisions.

		Fischlin	Pairwise collisions		
r	ℓ	Expected queries	ℓ	Exp queries	Improvement
8	2^{16}	64,877	2^{32}	58,190	1.11
10	2^{13}	8,233	2^{26}	7,293	1.13
12	2^{11}	2,038	2^{22}	1,824	1.12
14	2^{9}	509	2^{18}	448	1.13
16	2^{8}	267	2^{16}	232	1.15

Table 2: Comparing the computation cost of Fischlin's approach to our chosen prefix, pairwise collision approach. The reported value is the expected number of queries for finding either one preimage, or 2 collisions taken over 500-2000 experiments. Parameters for r and ℓ are set for the same 128 bit security.

5.2 Lower Bound on Prover Query Complexity

Fischlin [Fis05] proved via a meta reduction that any NIZKPoK scheme (with a non-programming extractor) for a language with a hard instance generator, must have a super-logarithmic number of queries V in λ made by the verifier to the random oracle. Fischlin's proof demonstrated asymptotic bounds due to its reliance on the hardness of the underlying language; in this work we are concerned with tight parameters for concrete security as guaranteed in the random oracle model, independently of the hardness of the underlying language. We therefore initiate a study of concrete query complexity, in particular we express this as the optimal prover query complexity P upon fixing V.

Caveat. We make a simplifying assumption, namely that the language L has a hard instance generator $\mathcal{I}$ such that the probability that any PPT algorithm is able to find a witness w for theorem $x \leftarrow \mathcal{I}(\lambda)$ is bounded by $\varepsilon_\lambda \ll 2^{-\lambda}$.

This assumption frequently does not hold as in practice one can instantiate the NIZKPoK with a concrete soundness level comparable to the hardness of instances generated by $\mathcal{I}$, however making this simplification allows us to focus on the random oracle query complexity of the NIZKPoK (which is given by parameters independent of the language) without having to account for concrete hardness of the language (which is very specific to each language and seldom leveraged by the extractor of a NIZKPoK scheme).

We begin with the following lemma, which is a tightening of [Fis05, Proposition 2]:

Lemma 3. *If* (P, V) *is a straight-line extractable NIZKPoK scheme for a* ε_λ*-hard language* L *in the random oracle model with the following characteristics for security parameter* λ*:*

- *Perfect zero-knowledge simulator* Sim
- ℓ*-bit output random oracle* H
- P *queries made by* P *to* H *in generating a proof*
- *Probability* $p_C > 0$ *of producing an accepting proof*
- V *queries made by deterministic* V *to* H *in verifying a proof, is a strict subset of the queries made by* P
- *Non-programming extractor* Ext *with error* $\leq 2^{-\lambda}$ *for an adversary that makes* $\leq V$ *queries to the random oracle*

Then it must hold that:

$$\binom{P}{V} \geq \frac{p_C}{2^{-\lambda} + \varepsilon_\lambda}$$

We can use the above lemma to derive the optimal prover query complexity for proofs that are non-trivially secure, i.e. when $V \ll \binom{P}{V}$. We define $P_{\mathsf{OPT}}[\lambda, V]$ to be the smallest prover query complexity for a given verifier query complexity V at a λ-bit security level.

Corollary 1. *If* (P, V) *is a perfectly complete straight-line extractable NIZKPoK scheme for* ε_λ*-hard language* L *in the random oracle model with all the characteristics required by Lemma 3 with the additional constraints that* $V < \lambda$ *and* $2^{-\lambda} \gg \varepsilon_\lambda$*, then the optimal prover query complexity is given by:*

$$P_{\mathsf{OPT}}[\lambda, V] \approx \left(V! \cdot 2^\lambda\right)^{\frac{1}{V}}$$

We defer both proofs to Appendix E. In subsequent text we drop the argument $[\lambda, V]$ when it is obvious. Note that P_{OPT} only characterizes the optimal prover query complexity for *perfectly complete* schemes. Since Lemma 3 accounts for schemes with arbitrary completeness errors, it is possible to amend Corollary 1 accordingly if desired. However we will see that P_{OPT} serves as a useful benchmark for our study. Interestingly Fischlin's scheme, which has the lowest prover query complexity in the literature, performs worse than P_{OPT} for all $V > 1$.

Claim 6. *Let* r *parameterize the number of repetitions of a Sigma protocol used to instantiate Fischlin's NIZK [Fis05] at a* λ*-bit security level. Then the average prover query complexity of the resulting scheme* $\mathsf{T}_{\mathsf{Fis}}$ *is a factor of* $r/(r!)^{1/r}$ *worse than the corresponding* P_{OPT}*. Therefore* $\mathsf{T}_{\mathsf{Fis}} > P_{\mathsf{OPT}}$ *for every* $r > 1$*.*

Proof. The average prover query complexity $\mathsf{T}_{\mathsf{Fis}}$ is given by the complexity of finding r inversions of the all-zero string of r independent λ/r-bit random oracles. This task requires $r \cdot 2^{\lambda/r}$ tries in expectation. Since $V = r$, the optimal prover complexity is given by $P_{\mathsf{OPT}} = (r! \cdot 2^\lambda)^{1/r}$. The ratio of the average prover complexity to the optimal is therefore:

$$\frac{\mathsf{T}_{\mathsf{Fis}}}{P_{\mathsf{OPT}}} = \frac{r \cdot 2^{\lambda/r}}{(r! \cdot 2^\lambda)^{1/r}} = \frac{r}{(r!)^{1/r}}$$

□

The ratio $\mathsf{T}_{\mathsf{Fis}}/P_{\mathsf{OPT}} = 1$ only when $r = 1$, which is of no use as the average complexity of computing a proof honestly matches the average complexity of forging a proof when $r = 1$. This ratio is $\sqrt{2} \approx 1.41$ when $r = 2$, and continues to increase as r grows, ultimately converging[7] at $e \approx 2.71$. Given this it is natural to ask, is it possible to meet P_{OPT} for any non-trivial parameters?

5.3 Special Case: $r + 1$-Special Sound Sigma Protocols

Given a Sigma protocol that is $r+1$-special sound and r simulatable (i.e. given r challenges, a simulator can produce r accepting transcripts) we are able to apply a multicollision predicate and reduce the prover's query complexity as compared with Fischlin's inversion predicate even further—to the point where we can meet P_{OPT} for a non-trivial parameter range.

Note that we present a randomized construction here—this aspect is orthogonal to query complexity. The purpose is to avoid dependence on 'quasi-unique responses', which we will discuss in detail in Section 6.

[7] $\lim_{r\to\infty} r/(r!)^{1/r} = e$

Protocol π_{NIZK}

The prover P and verifier V are both given the statement x while the prover also has a witness w for the statement $x \in L$. Both parties have access to an ℓ-bit Random Oracle $H : \{0,1\}^* \mapsto \{0,1\}^\ell$. The underlying Strongly $r+1$-special sound sigma protocol is given by $\Sigma = ((P_{\Sigma,a}, P_{\Sigma,z}), \mathsf{V}_\Sigma)$. Define $t = \ell + \lceil \log r \rceil$.

```
P^H(x, w):
1. Run $P_{\Sigma,a}(x,w)$ to obtain $\boldsymbol{a}$ and $\mathsf{state}$
2. Set $\mathcal{E} = \mathcal{Z} = \emptyset$ and do the following until an output is produced:
   (a) Uniformly sample $e \leftarrow \{0,1\}^t \setminus \mathcal{E}$
   (b) Set $z = P_{\Sigma,z}(\mathsf{state}, e)$ and append $(e,z)$ to $\mathcal{Z}$ and $e$ to $\mathcal{E}$
   (c) If $\exists (e_1,z_1),(e_2,z_2),\cdots,(e_r,z_r) \in \mathcal{Z}$ such that
           $H(\boldsymbol{a}, e_1, z_1) = H(\boldsymbol{a}, e_2, z_2) = \cdots = H(\boldsymbol{a}, e_r, z_r)$
       then set $\boldsymbol{e} = (e_i)_{i\in[r]}$ and $(z_i)_{i\in[r]}$ and output $\pi = (\boldsymbol{a}, \boldsymbol{e}, \boldsymbol{z})$

V^H(x, π):
1. Parse $(\boldsymbol{a}, \boldsymbol{e}, \boldsymbol{z}) = \pi$, and $(e_i)_{i\in[r]} = \boldsymbol{e}$, and $(z_i)_{i\in[r]} = \boldsymbol{z}$.
2. Check that $H(\boldsymbol{a}, e_1, z_1) = H(\boldsymbol{a}, e_2, z_2) = \cdots = H(\boldsymbol{a}, e_r, z_r)$
3. For each $i \in [r]$, check that $\mathsf{V}_\Sigma(x, (\boldsymbol{a}, e_i, z_i)) = 1$, aborting with output 0 if not
4. Accept by outputting 1
```

Fig. 5: Collision Based NIZK

We begin by refining the standard definition of Sigma protocols [Dam02] to incorporate a weaker notion of soundness and simulatability. This notion essentially requires (1) $r+1$-special soundness, which guarantees the success of an extractor upon being given $r+1$ accepting conversations that begin with the same first message, and (2) r-simulatability, which requires that for any statement, r accepting conversations (with the same first message) can be simulated for any r given challenges. We defer a formal definition to Appendix C, and give an instantiation based on Schnorr's PoK of discrete logarithm in Appendix D. We describe our NIZK transformation in Figure 5.

Theorem 7. *If Σ is a strongly $r+1$-special sound Sigma protocol and $\ell(r-1) = \lambda$, the protocol π_{NIZK} is a straight-line extractable NIZKPoK in the random oracle model, with an extractor that does not program the random oracle and achieves extraction error $Q/2^\lambda$ for an adversary making Q queries to the random oracle.*

Proof. (Sketch) We defer the full proof to Appendix C. Completeness follows from the pigeonhole principle, as any function that maps a domain of size $r \cdot 2^\ell$ to a range of size 2^ℓ will produce at least one r-collision. Zero-knowledge comes from the fact that the challenges $\boldsymbol{e}$ are distributed uniformly in $\{0,1\}^{t\cdot r}$, and the rest of the transcripts $\boldsymbol{a}, \boldsymbol{z}$ can be simulated by invoking $\mathsf{Sim}_\Sigma(x, r, \boldsymbol{e})$. Proof-of-knowledge follows from the fact that in order for an adversary to compute a proof by querying fewer than $r+1$ accepting Sigma protocol transcripts to H, the first r accepting transcripts it queries to H must all evaluate to the same ℓ-bit string. This happens with probability $(2^{-\ell})^{r-1} = 2^{-\lambda}$. $\square$

Query Complexity. We make use of the analysis of multicollision running times by von Mises [vM39] and revisited by Preneel [Pre93, Appendix B].

Corollary 2. *[vM39][Pre93, Theorem B.2 and pg. 283] If T balls are randomly distributed over n urns, the number T required to have at least one urn with r balls with probability $1-\exp(-\alpha_r)$ is given by the following equation:*

$$T \cdot \exp\left(-\frac{T}{r \cdot n}\right) = \left(\alpha_r \cdot n^{(r-1)} \cdot r!\right)^{1/r}$$

In order to obtain the time $\mathsf{T}_{\mathsf{Col}}$ required to find an r-collision in expectation, one must solve for T when the parameter $\alpha_r = 1$. Substituting $n = 2^{\lambda/(r-1)}$ for our context, we get that:

$$\mathsf{T}_{\mathsf{Col}} \cdot \exp\left(-\frac{\mathsf{T}_{\mathsf{Col}}}{r \cdot 2^{\lambda/(r-1)}}\right) = \left(2^{\lambda} \cdot r!\right)^{1/r} = P_{\mathsf{OPT}}$$

This equation is non-trivial to analyze relative to that of Fischlin, and so for ease of understanding we plot the ratio T/P_{OPT} for both π_{NIZK} and Fischlin's construction in Figure 6. This plot shows that for some reasonable parameterizations around $r \sim 5$, our construction achieves roughly 2x factor improvement in Prover complexity.

Fig. 6: Ratio of prover query complexities $\mathsf{T}_{\mathsf{Col}}$ and $\mathsf{T}_{\mathsf{Fis}}$ to the optimal P_{OPT} (y-axis) for different r parameters (x-axis), where $\mathsf{T}_{\mathsf{Col}}[r]$ and $\mathsf{T}_{\mathsf{Fis}}[r]$ are the number of oracle queries required to compute a proof in expectation upon fixing parameter r. Note that $\mathsf{T}_{\mathsf{Col}}/P_{\mathsf{OPT}}$ depends on the security parameter, whereas $\mathsf{T}_{\mathsf{Fis}}/P_{\mathsf{OPT}}$ is essentially invariant of it. Consequently we plot $\mathsf{T}_{\mathsf{Col}}/P_{\mathsf{OPT}}$ for a range of security parameters, where "λ-bit Col" denotes a λ-bit security level.

Finally, we note that Figure 6 only plots the ratio of Fischlin/Collision/optimal but does not convey the actual prover query complexities at those parameter choices. Table 5 in Appendix Hshows the Prover query costs below for selected parameter 130 bit security) to highlight our improvement.

6 Expanding the Applicability of Fischlin's Transform

As mentioned in Section 1, Fischlin's transformation applies to only a limited class of Sigma protocols that satisfy a *quasi-unique responses* constraint. Fischlin relied on this property to prove both zero-knowledge as well as proof of knowledge. While it is folklore that this property is not strictly necessary for the extractor, its necessity for zero-knowledge has remained thus far unclear.

We begin by showing in Section 6.1 a concrete attack on Witness Indistinguishability when Fischlin's transformation is applied to the Sigma protocol used to prove knowledge of one of two discrete logarithms [CDS94]. We then formalize a *strong special soundness* property for Sigma protocols that suffices for extraction, which includes languages that do not by default support the quasi-unique responses property, such as the logical OR Sigma protocol mentioned above. Finally we show how appropriately randomizing Fischlin's construction can achieve ZK unconditionally, for any strong special sound Sigma protocol.

6.1 Testing Witness Use in Fischlin's Transformation

Our distinguisher will not rely on the ability to query multiple accepting transcripts for the same challenge. For reference, we first recall the underlying Sigma protocol (due to Cramer et al. [CDS94]) in Figure 7.

Protocol $\Sigma_{\mathsf{DL}}^{\vee}$

The prover P and verifier V are both given the statement $(X_0, X_1) = (w_0 \cdot G, w_1 \cdot G) \in \mathbb{G}^2$ while the prover also has $w_b \in \mathbb{Z}_q$ for $b \in \{0,1\}$.

$\mathsf{P}^a_{\Sigma_{\mathsf{DL}}^{\vee}}((X_0, X_1), w_b)$:

1. Simulate a transcript for DLog proof of knowledge of X_{1-b}:
   - Sample $e_{1-b} \leftarrow \{0,1\}^\lambda$ and compute $(a_{1-b}, z_{1-b}) \leftarrow \mathsf{Sim}_{\Sigma_{\mathsf{DL}}}(X_{1-b}, e_{1-b})$
2. Sample $r_b \leftarrow \mathbb{Z}_q$ and compute $a_b = r_b \cdot G$
3. Publish commitment $a = (a_0, a_1)$ and output $\mathsf{state} = w_b, r_b, (a_{1-b}, e_{1-b}, z_{1-b})$

$\mathsf{P}^z_{\Sigma_{\mathsf{DL}}^{\vee}}(\mathsf{state}, e)$: Compute $e_b = e \oplus e_{1-b}$ and $z_b = w_b \cdot e_b + r_b$, and Output (e_0, e_1, z_0, z_1)

$\mathsf{V}(X, a, e, z)$:

1. Parse $a = (a_0, a_1)$ and $z = (e_0, e_1, z_0, z_1)$ and verify $e_0 \oplus e_1 = e$
2. Verify $z_b \cdot G = e_b \cdot X_b + a_b$ for each $b \in \{0,1\}$

Fig. 7: Proving knowledge of one of two discrete logarithms [CDS94]

An adversary attacking Witness Indistinguishability conventionally possesses two witnesses to the theorem and is given a proof π, and must determine which witness was used to produce it. We construct a more powerful type of attack, which makes use of a single witness and determines whether π was created using this witness or the opposite one. This fact will be useful when examining the protocol contexts in which our attack applies.

As we briefly discussed in Section 2.2, the attack strategy is to exploit the deterministic nature of Fischlin's prover by retrieving the Sigma protocol randomness and retracing the prover's steps. Concretely with Schnorr-style proofs, the messages z and c and the witness determine the randomness. The attacker can therefore retrieve this randomness, and simply replay the honest prover's algorithm and see if the resulting proof string is the same as the given one. The main subtle step in this attack's analysis is to argue that when this retrieve-and-retrace procedure is applied using a different witness from the one used to produce the proof string originally, there is a noticeable probability of producing a different proof string.

While the regular Witness Indistinguishability definition allows the adversary to supply both witnesses, in order to stay within the constraints of quasi-unique responses we formulate a stronger version of the WI experiment for our specific setting. In our definition the challenger samples both witnesses and gives the adversary only one of them (the other witness represents the trapdoor for the system parameter k). We define our experiment as follows:

$\mathsf{Expt}^{\mathsf{DL\text{-}WI}}_{\mathcal{A},\mathsf{P}}(1^\lambda)$:

1. The adversary $\mathcal{A}$ submits a bit $b \in \{0,1\}$ to the challenger
2. The challenger samples $w_0, w_1 \leftarrow \mathbb{Z}_q$ and sets $X_0 = g^{w_0}, X_1 = g^{w_1}$
3. The challenger tosses a coin $\beta \leftarrow \{0,1\}$, and computes $\pi \leftarrow \mathsf{P}((X_0, X_1), w_\beta)$
4. The challenger sends X_0, X_1, w_b, π to $\mathcal{A}$
5. $\mathcal{A}$ outputs a bit

The advantage $\mathsf{AdvDL\text{-}WI}[\mathcal{A},\mathsf{P}]$ of an adversary $\mathcal{A}$ is defined as:

$$|\Pr[\mathcal{A}(b, w_b, X_{1-b}, \pi) = 1 \mid \beta = 0] - \Pr[\mathcal{A}(b, w_b, X_{1-b}, \pi) = 1 \mid \beta = 1]|$$

Clearly any Witness Indistinguishable scheme will guarantee that the above advantage is negligible. We now give our concrete attack and analysis.

Lemma 4. *Let* P *be the prover's algorithm obtained by applying Fischlin's transformation [Fis05] to the Sigma protocol to prove knowledge of one of two discrete logarithms [CDS94]. Then there is an efficient adversary* $\mathcal{A}$ *such that* $\mathsf{AdvDL\text{-}WI}[\mathcal{A},\mathsf{P}]$ *is non-negligible.*

Equipped with this non-negligibly successful adversary $\mathcal{A}$, in Appendix J we will show how a natural protocol scenario that appears to enable quasi-unique responses in fact structurally resembles the $\mathsf{Expt}^{\mathsf{DL\text{-}WI}}_{\mathcal{A},\mathsf{P}}$ experiment. This allows us to deploy our $\mathsf{Expt}^{\mathsf{DL\text{-}WI}}_{\mathcal{A},\mathsf{P}}$ adversary $\mathcal{A}$ to break the security of the larger protocol.

Proof. For simplicity, we consider only a single base unit, i.e. assume that there is only one repetition in the transformed Sigma protocol.

Consider an attacker, that on input a proof $\pi = ((a_0, a_1), e, (e_0, e_1, z_0, z_1))$ obtained by applying Fischlin's transformation to $\Sigma^{\vee}_{\mathsf{DL}}$ using ℓ-bit output hash function H, and witness w_b, does the following:

1. Compute $r_b = z_b - w_b \cdot e_b$ and set $\mathsf{state}_b = w_b, r_b, (a_{1-b}, e_{1-b}, z_{1-b})$
2. Starting with $e = 0$, increment e until $H((a_0, a_1), e, (e_0, e_1, z_0, z_1)) = 0^\ell$ is found, where $(e_0, e_1, z_0, z_1) = \mathsf{P}^{z}_{\Sigma^{\vee}_{\mathsf{DL}}}(\mathsf{state}_b, e)$

3. Set $\pi_b = (a_0, a_1), e', (e'_0, e'_1, z'_0, z'_1)$
4. If $\pi_b = \pi$ output b, otherwise output $1 - b$.

Denote the witness used by the challenger to produce the proof as w_β. When $\beta = b$ the attacker outputs the correct bit with certainty since the honest prover's steps are perfectly reconstructed to produce $\pi_b = \pi$. The interesting case to analyze is when $\beta = 1 - b$. There are two possible outcomes triggered in this case, i.e., $\pi_b = \pi$ and $\pi_b \neq \pi$. The latter outcome is induced by the attacker finding an accepting transcript (a, e', z') with $e' < e$ that resulted in $H(a, e', z') = 0^\ell$ (note that $e' > e$ is impossible as we know that $H(a, e, z) = 0^\ell$, and so the prover never increments past e). The implication in this event is that π was certainly not produced using w_b; this is because had the honest prover started with witness w_b and state state_b, it would have terminated with output $\pi' = (a, e', z')$ rather than the given π.

It remains to show that this distinguishing event (call it $\mathsf{diffProof}$) occurs with non-negligible probability. Note that since the attack is always successful when $\beta = b$, the value $\Pr[\mathsf{diffProof}]$ characterizes the distinguishing advantage of this attack. This is because $\mathsf{AdvDL\text{-}WI}[\mathcal{A}, \mathsf{P}]$ can be simplified as follows, given that b is fixed:

$$|\Pr[\mathcal{A}(w_b, X_{1-b}, \pi) = b \mid \beta = b] - \Pr[\mathcal{A}(w_b, X_{1-b}, \pi) = b \mid \beta = 1 - b]|$$

$$= |1 - (1 - \Pr[\mathsf{diffProof}])| = \Pr[\mathsf{diffProof}]$$

Let $Q_{b,i}$ be the query made by the attacker that corresponds to responding to the i^{th} challenge using witness w_b; in particular

$$Q_{b,i} = (a_0, a_1), i, \mathsf{P}^z_{\Sigma^\vee_{\mathsf{DL}}}(\mathsf{state}_b, i)$$

and thus $\pi_b = Q_{b,i}$ for the smallest i such that $H(Q_{b,i}) = 0^\ell$. Define $Q_{1-b,i}$ the same way using $\mathsf{state}_{1-b} = w_{1-b}, r_{1-b}, (a_b, e_b, z_b)$, except that the query is made by the challenger rather than the attacker in this experiment (since $\beta = 1 - b$).

Claim 8. *$\forall e' \neq e$, it holds that $Q_{0,e'} \neq Q_{1,e'}$.*

Proof. Consider any $e' \neq e$. Let $e'_0 = e' \oplus e_1$ and $e'_1 = e' \oplus e_0$. Clearly $e'_0 \neq e_0$ and $e'_1 \neq e_1$ as $e' \neq e = e_0 \oplus e_1$. By the structure of $\mathsf{P}^z_{\Sigma^\vee_{\mathsf{DL}}}(\mathsf{state}_b, e')$, the queries $Q_{b,e'}$ are correspondingly constructed as follows:

$$Q_{0,e'} = (\cdots e'_0, e_1, \cdots) \text{ and } Q_{1,e'} = (\cdots e_0, e'_1, \cdots)$$

Clearly $Q_{0,e'} \neq Q_{1,e'}$ as $e_0 \neq e'_0$ and $e_1 \neq e'_1$. $\square$

Corollary 3. *$\forall e' \neq e$, the values $H(Q_{0,e'})$ and $H(Q_{1,e'})$ are independently distributed.*

Recall that the event $\mathsf{diffProof}$ is precisely the event that the attacker finds an accepting proof $\pi_b = (a, e', z')$ such that $e' < e$. Rather than characterizing $\mathsf{diffProof}$ in its entirety, we analyze a simpler special case. In particular, the event

$H(Q_{\beta,0}) \neq 0^\ell$ (implying $e > 0$ in π) and $H(Q_{1-\beta,0}) = 0^\ell$ (implying $e' = 0$ and hence $\pi_b \neq \pi$) induces $\mathsf{diffProof}$. Then applying Corollary 3 we can therefore lower bound $\Pr[\mathsf{diffProof}]$ as follows:

$$\begin{aligned}\Pr[\mathsf{diffProof}] &\geq \Pr[H(Q_{\beta,0}) \neq 0^\ell \wedge H(Q_{1-\beta,0}) = 0^\ell] \\ &= \Pr[H(Q_{\beta,0}) \neq 0^\ell] \cdot \Pr[H(Q_{1-\beta,0}) = 0^\ell] \\ &= \frac{2^\ell - 1}{2^\ell} \cdot \frac{1}{2^\ell} = \frac{2^\ell - 1}{2^{2\ell}}\end{aligned}$$

As we know that $\ell \in O(\log \lambda)$ is necessary for completeness, the denominator of the above value $2^{2\ell} \in \mathsf{poly}(\lambda)$. We therefore conclude that $\Pr[\mathsf{diffProof}]$ is non-negligible in λ, and this completes the analysis.

□

6.2 Strong Special Soundness

Before describing how to patch the above attack, we present an easily verifiable property of Sigma protocols for which our transformation applies. Rather than attempting to quantify the ability of an adversary to induce a bad event, we take a constructive approach in our definition; i.e., it is easier to evaluate precise deterministic conditions (such as special soundness) rather than reason about probabilistic/computational system parameters (as in quasi-unique responses).

Our definition is a mild strengthening of the two-special soundness notion for Sigma protocols [Dam02], and so we call it *strong* two-special soundness—also in homage to the similar concept of *strong* unforgeability for signature schemes. Informally stated, a strongly two-special sound sigma protocol has an extractor which when given two distinct accepting transcripts (a, e, z) and (a, e', z') that share the same first message, outputs a witness for the statement with certainty (note that $e = e'$ is allowed). The standard two-special soundness notion enforces that $e \neq e'$ for the extractor's success. We give the formal definition in Definition 3 in Appendix A.

Many natural sigma protocols (including logical compositions [CDS94], Okamoto's identification protocol [Oka93], etc.) satisfy this definition (but may not satisfy quasi-unique responses). There are two notable natural examples that may not meet this definition: (1) Blum's protocol to prove knowledge of a Hamiltonian cycle [Blu86] allows the prover to open any cycle in the graph and it is unclear as to how an extractor for strong special soundness can deal with such a situation, and (2) the Sigma protocol that underlies EdDSA [BDL+12], which is Schnorr's scheme implemented over an elliptic curve group of composite order. The lax verification equation in the original specification means that the verifier accepts multiple discrete logarithms for the same curve point. However we stress that this is due to lax realization of the abstraction required for Schnorr's sigma protocol, and is easily fixed in works that succeeded the original spec [CGN20, BCJZ21]. Note that both cases will not support quasi-unique responses either, if they are not strong special sound.

Note that any standard Sigma protocol that is not strongly two-special sound can not have quasi-unique responses. In particular by definition the only way to retain standard special soundness while violating strong two-special soundness is by presenting accepting transcripts $(a, e, z_1), (a, e, z_2)$ that do not yield a witness for the theorem when given to the extractor. Any notion of *efficient* adversaries being unable to find such transcripts in the case of quasi-unique responses is captured by amending the theorem for the strong two-special sound Sigma protocol to include a disjunctive clause for knowledge of the system parameter trapdoor.

With our definition in place, we study how to compile such Sigma protocols to NIZKPoKs using Fischlin's technique.

6.3 Randomization Extends Fischlin's Technique

The issue in Fischlin's transformation is that the prover's algorithm is deterministic and consequently re-traceable. Indeed, if one were to instantiate the transformation of Pass [Pas03] by simply constructing a hash tree of accepting protocol transcripts instead of a Merkle tree of *commitments* to such transcripts, the same issue as described above would present itself more directly: given a proof and candidate witness for the statement, one could simply extract the prover's randomness and test if recomputing the proof once again yields the given one. This issue is implicitly avoided by Pass (at constant factor overhead) by constructing the Merkle tree with commitments to protocol transcripts. However it is unclear how to make such an approach work with Fischlin's transform; using randomized commitments appears to be at odds with obtaining soundness.

We show that an alternate method of randomization can be used to extend Fischlin's technique to any strong special sound Sigma protocol. The idea is to randomize the NIZK prover's algorithm so that the prover randomly steps through the challenge space until an accepting transcript that hashes to the all-zero string is found. Intuitively, proofs produced with this modified transformation do not leak any information about how many queries the prover had to make in order to find an accepting transcript. This makes it impossible for a distinguisher to retrace the steps of a prover even given all witnesses as it does not have access to the random sequence in which the prover queried the random oracle. We give a formal description of the modified transformation in Figure 9 in Appendix B, along with a proof of security.

Bibliography

[ABGR13] Prabhanjan Ananth, Raghav Bhaskar, Vipul Goyal, and Vanishree Rao. On the (in)security of Fischlin's paradigm. In *TCC 2013*, 2013.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *ACM CCS 2017*, 2017.

[BCJZ21] Jacqueline Brendel, Cas Cremers, Dennis Jackson, and Mang Zhao. The provable security of ed25519: Theory and practice. In *IEEE S&P 2021*, 2021.

[BCKL21] Eli Ben-Sasson, Dan Carmon, Swastik Kopparty, and David Levit. Elliptic curve fast fourier transform (ECFFT) part I: fast polynomial algorithms over all finite fields. *ECCC*, page 103, 2021.

[BCR+19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In *EUROCRYPT 2019, Part I*, 2019.

[BDL+12] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *Journal of Cryptographic Engineering*, 2012.

[Ber06] Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. In *PKC 2006*, 2006.

[Blo] Average transactions per block — blockchain.com. https://www.blockchain.com/charts/n-transactions-per-block. Accessed: 2022-Feb-11.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In *ASIACRYPT 2001*, 2001.

[Blu86] Manuel Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, volume 1, page 2. Citeseer, 1986.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO'94*, 1994.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In *CRYPTO 2001*, 2001.

[CGKN21] Konstantinos Chalkias, François Garillot, Yashvanth Kondi, and Valeria Nikolaenko. Non-interactive half-aggregation of eddsa and variants of schnorr signatures. In *CT-RSA 2021*, 2021.

[CGN20] Konstantinos Chalkias, François Garillot, and Valeria Nikolaenko. Taming the many eddsas. In *SSR*, 2020.

[CJS14] Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical UC security with a global random oracle. In *ACM CCS 2014*, 2014.

[Dam02] Ivan Damgård. On Σ-protocols. In *Lecture Notes, University of Aarhus, Department for Computer Science*, 2002.

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *CRYPTO 2005*, 2005.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*, 1987.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *22nd ACM STOC*, 1990.

[GLSY04] Rosario Gennaro, Darren Leigh, R. Sundaram, and William S. Yerazunis. Batching Schnorr identification scheme with applications to privacy-preserving authorization and low-bandwidth communication devices. In *ASIACRYPT 2004*, 2004.

[HMPs14] Susan Hohenberger, Steven Myers, Rafael Pass, and abhi shelat. ANONIZE: A large-scale anonymous survey system. In *IEEE S&P*, 2014.

[Lin22] Yehuda Lindell. Simple three-round multiparty schnorr signing with full simulatability. *IACR Cryptol. ePrint Arch.*, page 374, 2022.

[Oka93] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO'92*, 1993.

[Pas03] Rafael Pass. On deniability in the common reference string and random oracle model. In *CRYPTO 2003*, 2003.

[Pre93] Bart Preneel. *Analysis and design of cryptographic hash functions*. PhD thesis, Katholieke Universiteit te Leuven, 1993.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *EUROCRYPT'96*, 1996.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 1991.

[SG02] Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, 15(2):75–96, March 2002.

[SLdW07] Marc Stevens, Arjen K. Lenstra, and Benne de Weger. Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities. In *EUROCRYPT 2007*, 2007.

[Unr15] Dominique Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In *EUROCRYPT 2015*, 2015.

[vM39] Richard von Mises. *Über Aufteilungs-und Besetzungswahrscheinlichkeiten*. na, 1939.

[vzGG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3 edition, 2013.

[wbo] Ecfft algorithms on the bn254 base field. `https://github.com/wborgeaud/ecfft-bn254`. Accessed: 2022-Feb-12.

Supplementary Material

A Definitions

The standard definition of a Sigma protocol is given below.

Definition 2. *[Dam02] A Sigma protocol for relation R is a three move public coin protocol between a prover P_Σ and verifier V_Σ that has the following properties:*

– ***Completeness**: If P_Σ (with private input w) and V_Σ with public input x such that $(x, w) \in R$ execute the protocol honestly, then the protocol always terminates with V accepting.*

– ***Two-special soundness**: There exists an efficient extractor* Ext *which given as input the accepting conversations $T = (a, e, z)$ and $T' = (a, e', z')$ for statement x such that $e \neq e'$, outputs w such that $(x, w) \in R$.*

– ***Honest verifier zero-knowledge**: There exists an efficient simulator* Sim *which upon input a statement x and challenge e outputs a, z such that (a, e, z) is an accepting conversation. Moreover when e is uniformly chosen, (a, e, z) is distributed identically to an execution of the honest protocol.*

A strong-special sound Sigma protocol—which is a notion that we introduce in this paper—additionally has the following property:

Definition 3. *A strongly two-special sound Sigma protocol for relation R is a three move protocol between a prover P and verifier V that is complete and honest verifier zero-knowledge as per Definition 2, and additionally has the following property:*

– ***Strong two-special soundness**: There exists an extractor* Ext *which given as input the accepting conversations $T = (a, e, z)$ and $T' = (a, e', z')$ for statement x such that $T \neq T'$, outputs w such that $(x, w) \in R$.*

Next we present the definition of straightline extraction as given by Pass.

Definition 4 ([Pas03]). *We say that an interactive proof with negligible soundness (P, V) for the language $L \in NP$, with the witness relation R_L, is straight-line witness extractable in the RO model if for every PPT machine P^* there exists a PPT witness extractor machine E such that for all $x \in L$, all $y, r \in \{0, 1\}^*$, if $P^*_{x,y,r}$ convinces the honest verifier with non-negligible probability, on common input x, then $E(view_V[(P^*x, y, r, V(x))], \ell) \in RL(x)$ with overwhelming probability, where $P^*_{x,y,r}$ denotes the machine P^* with common input fixed to x, auxiliary input fixed to y and random tape fixed to r, $view_V[(P^*_{x,y,r}, V(x))]$ is V's view including its random tape, when interacting with $P^*_{x,y,r}$, and ℓ is a list of all oracle queries and answers posed by $P^*_{x,y,r}$ and V.*

We recall Fischlin's transformation in Figure 8.

Protocol $\pi_{\mathsf{NIZK}}^{\mathsf{Fis05}}$

The prover P and verifier V are both given the statement x while the prover also has a witness w for the statement $x \in L$. The security parameter λ defines the integers r, ℓ, t. These integers are related as $r \cdot \ell = 2^\lambda$, and $t = \lceil \log \lambda \rceil \cdot \ell$. Both parties have access to a Random Oracle $H : \{0,1\}^* \mapsto \{0,1\}^\ell$. The underlying sigma protocol is given by $\Sigma = ((\mathsf{P}_\Sigma^a, \mathsf{P}_\Sigma^z), \mathsf{V}_\Sigma)$.

```
P^H(x, w):
1. For each $i \in [r]$, compute $(a_i, \mathsf{state}_i) \leftarrow \mathsf{P}_\Sigma^a(x, w)$
2. Set $\boldsymbol{a} = (a_i)_{i\in[r]}$, and initialize $e_i = -1$ for each $i \in [r]$
3. For each $i \in [r]$, do the following:
   (a) If $e_i > t$, abort. Otherwise increment $e_i$ and compute $z_i = \mathsf{P}_\Sigma^z(\mathsf{state}_i, e_i)$
   (b) If $H(\boldsymbol{a}, i, e_i, z_i) \neq 0^\ell$, repeat Step 3a
4. Output $\pi = (a_i, e_i, z_i)_{i\in[r]}$

V^H(x, π):
1. Parse $(a_i, e_i, z_i)_{i\in[r]} = \pi$, and set $\boldsymbol{a} = (a_i)_{i\in[r]}$
2. For each $i \in [r]$, verify that $H(\boldsymbol{a}, i, e_i, z_1) = 0^\ell$ and $\mathsf{V}_\Sigma(x, (a_i, e_i, z_i)) = 1$, aborting with output 0 if not
3. Accept by outputting 1
```

Fig. 8: Fischlin's Transformation [Fis05]

B Fischlin's Transformation for Any Strong 2-Special Sound Sigma Protocol

In this section we present our randomization of Fischlin's transform that compiles any strong two-special sound Sigma protocol to a NIZKPoK. We give the protocol $\pi_{\mathsf{NIZK}}^{\mathsf{F\text{-}rand}}$ in Figure 9 and prove it secure in Theorem 9.

Theorem 9. *If Σ is a strongly two-special sound sigma protocol for the language L, then protocol $\pi_{\mathsf{NIZK}}^{\mathsf{F\text{-}rand}}$ is a straight-line extractable non-interactive zero-knowledge proof of knowledge for the language L in the random oracle model.*

Proof. **Completeness:** follows from the same analysis as Fischlin [Fis05]. Denote by Q_{i,e_i} the query made by P in Step 3c of its algorithm. The only event in which the prover does not find an accepting proof is when $\exists i \in [r]$ such that $\forall e_i \in \{0,1\}^t$, $H(Q_{i,e_i}) \neq 0^\ell$. Call this event fail. As each $H(Q_{i,e_i})$ is independent,

Protocol $\pi_{\mathsf{NIZK}}^{\mathsf{F\text{-}rand}}$

The prover P and verifier V are both given the statement x while the prover also has a witness w for the statement $x \in L$. The security parameter λ defines the integers r, ℓ, t. These integers are related as $r \cdot \ell = 2^\lambda$, and $t = \lceil \log \lambda \rceil \cdot \ell$. Both parties have access to a Random Oracle $H : \{0,1\}^* \mapsto \{0,1\}^\ell$. The underlying sigma protocol is given by $\Sigma = ((\mathsf{P}^a_\Sigma, \mathsf{P}^z_\Sigma), \mathsf{V}_\Sigma)$.

```
$\mathsf{P}^H(x, w)$:
1. For each $i \in [r]$, compute $(a_i, \mathsf{state}_i) \leftarrow \mathsf{P}^a_\Sigma(x, w)$
2. Set $\boldsymbol{a} = (a_i)_{i\in[r]}$
3. For each $i \in [r]$, do the following:
   (a) Set $\mathcal{E}_i = \emptyset$
   (b) Sample $e_i \leftarrow \{0,1\}^t \setminus \mathcal{E}_i$ and compute $z_i = \mathsf{P}^z_\Sigma(\mathsf{state}_i, e_i)$
   (c) If $H(\boldsymbol{a}, i, e_i, z_i) \neq 0^\ell$, update $\mathcal{E}_i = \mathcal{E}_i \cup \{e_i\}$ and repeat Step 3b
4. Output $\pi = (a_i, e_i, z_i)_{i\in[r]}$
$\mathsf{V}^H(x, \pi)$:
1. Parse $(a_i, e_i, z_i)_{i\in[r]} = \pi$, and set $\boldsymbol{a} = (a_i)_{i\in[r]}$
2. For each $i \in [r]$, verify that $H(\boldsymbol{a}, i, e_i, z_1) = 0^\ell$ and $\mathsf{V}_\Sigma(x, (a_i, e_i, z_i)) = 1$, aborting with output 0 if not
3. Accept by outputting 1
```

Fig. 9: Randomized Fischlin's Transformation

we can bound the probability of fail as follows:

$$\begin{aligned}
\Pr[\mathsf{fail}] &= \Pr[\exists i \in [r] : \forall e_i \in \{0,1\}^t,\ H(Q_{i,e_i}) \neq 0^\ell] \\
&\leq \sum_{i\in[r]} \Pr[\forall e_i \in \{0,1\}^t,\ H(Q_{i,e_i}) \neq 0^\ell] \\
&= \sum_{i\in[r]} \prod_{e_i\in\{0,1\}^t} \Pr[H(Q_{i,e_i}) \neq 0^\ell] \\
&= \sum_{i\in[r]} \prod_{e_i\in\{0,1\}^t} \left(1 - \frac{1}{2^\ell}\right) = r \cdot \left(1 - \frac{1}{2^\ell}\right)^{2^t} \\
&= r \cdot \left(1 - \frac{1}{2^\ell}\right)^{\lambda \cdot 2^\ell} \approx r \cdot \frac{1}{e^\lambda} \\
&\leq 2^{-\lambda}
\end{aligned}$$

Proof of knowledge: This follows from the same analysis as Fischlin [Fis05] as well.

The event in which this extractor fails is the event in which an adversarial prover P^* is able to produce a proof π by querying no more than a single accepting Sigma protocol transcript for each $i \in [r]$ to the random oracle. We first ignore all queries made to H that are not accepting transcripts, and then separate queries

Extractor $\mathsf{Ext}_{\mathsf{NIZK}}$

The extractor is given the statement x, a proof π, and the list of queries to the random oracle $\boldsymbol{Q}$ that were made by the adversary in the production of this proof. In addition to this, this extractor has access to the extractor Ext_{Σ} of the strongly special sound sigma protocol, which requires 2 accepting transcripts (with the same a value) in order to produce a witness w for the statement.

```
$\mathsf{Ext}_{\mathsf{NIZK}}(x, \pi, \boldsymbol{Q})$:
1. Parse $(a_i, e_i, z_i)_{i\in[r]} = \pi$, and set $\boldsymbol{a} = (a_i)_{i\in[r]}$
2. Search $\boldsymbol{Q}$ until a query of the form $(\boldsymbol{a}, i, e, z)$ is found such that $(e, z) \neq (e_i, z_i)$,
   and $\mathsf{V}_{\Sigma}(x, a_i, e, z)) = 1$
3. Output $\mathsf{Ext}_{\Sigma}(a_i, e_i, e, z_i, z)$
```

Fig. 10: Extracting a witness

prefixed by different $\boldsymbol{a}$ as they essentially instantiate independent random oracles (and can not be combined with one another to produce a proof). For a given $\boldsymbol{a}$, the event in which the adversary is able to output an accepting proof with fewer than 2 accepting transcripts (prefixed by $\boldsymbol{a}$) queried to H for each $i \in [r]$ is exactly the event that the first such accepting transcript queried to H for every $i \in [r]$ evaluates to 0. This is equivalent to r independent uniformly chosen ℓ-bit strings being equal to 0, which happens with probability $(2^{-\ell})^r = 2^{-\lambda}$. For an adversary that makes $|\boldsymbol{Q}|$ queries to the random oracle, the extraction error is therefore bounded by $|\boldsymbol{Q}|/2^{\lambda}$.

Zero-knowledge: We describe how to simulate a proof in Figure 11, and then show its indistinguishability from a real proof.

Simulator $\mathcal{S}^{\mathsf{F}}_{\mathsf{NIZK}}$

Simulator $\mathcal{S}^{\mathsf{F}}_{\mathsf{NIZK}}$ is given the statement x, and has the ability to program the Random Oracle H. In addition to this $\mathcal{S}^{\mathsf{F}}_{\mathsf{NIZK}}$ is given the simulator for the Sigma protoocol Sim_{Σ}.

```
$\mathcal{S}^{\mathsf{F}}_{\mathsf{NIZK}}(x)$:
1. Uniformly sample $e_i \leftarrow \{0,1\}^t$ for each $i \in [r]$ and set $\boldsymbol{e} = (e_i)_{i\in[r]}$
2. Run the simulator for the sigma protocol to obtain $(a_i, z_i) \leftarrow \mathsf{Sim}_{\Sigma}(x, e_i)$ for
   each $i \in [r]$
3. Program the random oracle $H$ so that $H(\boldsymbol{a}, e_i, z_i) = 0$ for each $i \in [r]$
4. Emulate $H$ as a random oracle 'honestly' for every other query
5. Output $\pi = (\boldsymbol{a}, \boldsymbol{e}, \boldsymbol{z})$
```

Fig. 11: Simulator for Zero-Knowledge

We argue that the simulation is indistinguishable from a real proof through a sequence of hybrid experiments, which are defined as follows.

Hybrid $\mathcal{H}_1$. The real prover's algorithm (P from $\pi_{\mathsf{NIZK}}^{\mathsf{F\text{-}rand}}$) is used to find $(\boldsymbol{a}, \boldsymbol{e}, \boldsymbol{z})$ such that $H(\boldsymbol{a}, e_1, z_1) = \cdots = H(\boldsymbol{a}, e_r, z_r) = 0$ where H is emulated as a random oracle by the standard technique of maintaining a (query, response) table. The difference from the real prover's algorithm is merely syntactic.

Hybrid $\mathcal{H}_2$. Implement Step 3 of $\mathcal{S}_{\mathsf{NIZK}}^{\mathsf{F}}$. In particular in this experiment, the random oracle H is implemented as follows:

1. The first r queries by the honest prover $Q_1, Q_2, \cdots Q_r$ (where each $Q_i = (\boldsymbol{a}, e_i, z_i)$ as generated by P) will receive 0 as a response, i.e. $H(Q_1) = H(Q_2) = \cdots = H(Q_k) = 0$
2. Emulate H as a random oracle 'honestly' for every other query

This hybrid differs from the last in that here the prover P will terminate after the first r queries it makes to H, whereas in $\mathcal{H}_1$ since H is not programmed to shortcut to 0, P will have to 'work' to find accepting transcripts that evaluate to 0. Since the difference in running time of $\mathcal{H}_2$ and $\mathcal{H}_1$ is invisible to a distinguisher and $\boldsymbol{a}$ are generated identically in both hybrids, the only component that remains to be analyzed is $\boldsymbol{e}$ (since $\boldsymbol{z}$ is implicitly fixed by $w, \boldsymbol{a}, \boldsymbol{e}$). In $\mathcal{H}_1$, each e_i represents the index at which the first pre-image of 0 was found by P relative to $H(\boldsymbol{a}, i, \cdot)$. Since P steps through pre-images uniformly at random and H is a random oracle (i.e. H has independent uniformly random outputs for every pair of distinct inputs) each e_i is distributed uniformly in $\{0,1\}^t$ in $\mathcal{H}_1$. In $\mathcal{H}_2$, each e_i is clearly uniformly distributed in $\{0,1\}^t$ as it corresponds to the first r challenges tried by P, which are sampled uniformly and independently.

As $\boldsymbol{a}, \boldsymbol{e}, \boldsymbol{z}$ are distributed identically in $\mathcal{H}_2$ and $\mathcal{H}_1$, the only distinguishing event corresponds to the programming of H, i.e. if the adversary is able to query H on some index that $\mathcal{H}_2$ subsequently programs to a different value. Since $\boldsymbol{a}$ has at least λ bits of entropy and is a prefix for all queries programmed in $\mathcal{H}_2$, this distinguishing event happens with probability no greater than $|\boldsymbol{Q}|/2^\lambda$, where $|\boldsymbol{Q}|$ is the number of queries made by the adversary to the random oracle.

Hybrid $\mathcal{H}_3$. We define hybrid experiment ${\mathcal{H}_3}^0$ to be the same as the last, with the only change being that the vector of challenges $\boldsymbol{e}$ is sampled before invoking $P_{\Sigma,a}$. This change is merely syntactic, and ${\mathcal{H}_3}^0$ is distributed identically to $\mathcal{H}_2$. We now define a sequence of sub-hybrids $\{{\mathcal{H}_3}^i\}_{i\in[r]}$ as follows: hybrid experiments ${\mathcal{H}_3}^{i-1}$ and ${\mathcal{H}_3}^i$ are identical except that they differ in their computation of (a_i, z_i). In particular, ${\mathcal{H}_3}^{i-1}$ computes $(a_i, \mathsf{state}_i) \leftarrow P_{\Sigma,a}$ and $z_i \leftarrow P_{\Sigma,z}(e_i, \mathsf{state}_i)$ whereas ${\mathcal{H}_3}^i$ computes $(a_i, z_i) \leftarrow \mathsf{Sim}_\Sigma(x, e_i)$. Clearly distinguishing ${\mathcal{H}_3}^{i-1}$ from ${\mathcal{H}_3}^i$ is equivalent to distinguishing a simulated Sigma protocol transcript from a real one. By perfect simulation of the sigma protocol, we have that ${\mathcal{H}_3}^{i-1} \equiv {\mathcal{H}_3}^i$ for each $i \in [r]$.

The final experiment in this sequence $\mathcal{H}_3{}^r$ implements Steps 1 and 2 of $\mathcal{S}^{\mathsf{F}}_{\mathsf{NIZK}}$ and is entirely independent of the witness, which completes the process of replacing the real $\mathsf{P}(x, w)$ with the simulation $\mathcal{S}^{\mathsf{F}}_{\mathsf{NIZK}}(x)$.

□

C Full Proof of Theorem 7

We first define r-special sound sigma protocols.

Definition 5. *Let λ be the security parameter, which is polynomially related to the instance size. A strongly r-special sound Sigma protocol for relation R is a three move public coin protocol between a prover P_Σ and verifier V_Σ that has the following properties:*

– ***Completeness:*** *If P_Σ (with private input w) and V_Σ with public input x such that $(x, w) \in R$ execute the protocol honestly, then the protocol always terminates in $\mathsf{poly}(\lambda)$ time with V accepting.*

– ***Strong r-special soundness:*** *There exists a PPT extractor Ext which given as input the accepting conversations $\{T_i = (a, e_i, z_i)\}_{i\in[r]}$ for statement x such that $T_i \neq T_j$ for every distinct pair $i, j \in [r]$, outputs w such that $(x, w) \in R$.*

– ***Honest verifier zero-knowledge/$r-1$ Simulatability:*** *There exists a PPT simulator Sim which upon input a statement x and challenges $\{e_i\}_{i\in[r-1]}$ outputs $a, \{z_i\}_{i\in[r-1]}$ such that each (a, e_i, z_i) is an accepting conversation.*

We restate the theorem below, and give the full proof.

Theorem 10. *If Σ is a strongly $r+1$-special sound Sigma protocol and $\ell(r-1) = \lambda$, the protocol π_{NIZK} is a straight-line extractable NIZKPoK in the random oracle model, with an extractor that does not program the random oracle and achieves extraction error $Q/2^\lambda$ for an adversary making Q queries to the random oracle.*

Proof. We first argue completeness, then extraction and zero-knowledge.

Completeness: The prover P terminates successfully with a proof when it finds a multicollision of size r for a function that maps a domain of size $r \cdot 2^\ell$ to a range of size 2^ℓ. By the pigeonhole principle, there exists at least one such multicollision, and since the prover checks the domain exhaustively, such a multicollision is always found.

Extraction: We give the straight-line extractor $\mathsf{Ext}_{\mathsf{NIZK}}$ in Figure 12 and then argue that it fails with probability exponentially small in λ.

The event in which this extractor fails is the event in which an adversarial prover P^* is able to produce a proof π by querying no more than r accepting Sigma protocol transcripts to the random oracle. We first ignore all queries made to H that are not accepting transcripts, and then separate queries prefixed by different $\boldsymbol{a}$ as they essentially instantiate independent random oracles (and are not compatible with one another). For a given $\boldsymbol{a}$, the event in which the adversary is able to output an accepting proof with fewer than $r + 1$ accepting transcripts (prefixed by $\boldsymbol{a}$) queried to H is exactly the event that all of the first r such

Extractor $\mathsf{Ext}_{\mathsf{NIZK}}$

The extractor is given the statement x, a proof π, and the list of queries to the random oracle $\boldsymbol{Q}$ that were made by the adversary in the production of this proof. In addition to this, this extractor has access to the extractor Ext_Σ of the strongly $r+1$ special sound sigma protocol, which requires $r+1$ accepting transcripts (with the same a value) in order to produce a witness w for the statement.

```
Ext_NIZK(x, π, Q):
1. Parse $(\boldsymbol{a}, \boldsymbol{e}, \boldsymbol{z}) = \pi$, and $(e_i)_{i\in[r]} = \boldsymbol{e}$, and $(z_i)_{i\in[r]} = \boldsymbol{z}$
2. Initialize $\tau = (e_i, z_i)_{i\in[r]}$
3. Search $\boldsymbol{Q}$ until a query of the form $(\boldsymbol{a}, e, z)$ is found such that $(e, z) \notin \tau$, and
   $\mathsf{V}_\Sigma(x, (\boldsymbol{a}, e, z)) = 1$
4. Output $\mathsf{Ext}_\Sigma(a_i, \tau)$
```

Fig. 12: Extracting a witness

accepting transcripts queried to H evaluate to the same value. This is equivalent to r independent uniformly chosen ℓ-bit strings being equal, which happens with probability $(2^{-\ell})^{(r-1)} = 2^{-\lambda}$. For an adversary that makes Q queries to the random oracle, the extraction error is therefore bounded by $Q/2^\lambda$.

Zero-knowledge: We describe how to simulate a proof in Figure 13, and then show its indistinguishability from a real proof.

Simulator $\mathcal{S}_{\mathsf{NIZK}}$

Simulator $\mathcal{S}_{\mathsf{NIZK}}$ is given the statement x, and has the ability to program the Random Oracle H. In addition to this $\mathcal{S}_{\mathsf{NIZK}}$ is given the simulator for the Sigma protoocol Sim_Σ. Let $t = \ell + \log r$

```
S_NIZK(x):
1. Uniformly sample $e_i \leftarrow \{0,1\}^t$ for each $i \in [r]$ and set $\boldsymbol{e} = (e_i)_{i\in[r]}$
2. Run the simulator for the sigma protocol to obtain $(\boldsymbol{a}, \boldsymbol{z}) \leftarrow \mathsf{Sim}_\Sigma(x, r, \boldsymbol{e})$
3. Sample $v \leftarrow \{0,1\}^\ell$
4. Program the random oracle $H$ so that $H(\boldsymbol{a}, e_i, z_i) = v$ for each $i \in [r]$
5. Emulate $H$ as a random oracle 'honestly' for every other query
6. Output $\pi = (\boldsymbol{a}, \boldsymbol{e}, \boldsymbol{z})$
```

Fig. 13: Simulator for Zero-Knowledge

We argue that the simulation is indistinguishable from a real proof through a sequence of hybrid experiments, which are defined as follows.

Hybrid $\mathcal{H}_1$. The real prover's algorithm (P from π_{NIZK}) is used to find $(\vec{a}, \vec{e}, \vec{z})$ such that $H(\boldsymbol{a}, e_1, z_1) = \cdots = H(\boldsymbol{a}, e_r, z_r)$ where H is emulated as a random oracle by the standard technique of maintaining a (query, response) table. The difference from the real prover's algorithm is merely syntactic.

Hybrid $\mathcal{H}_2$. Implement Steps 3 and 4 of $\mathcal{S}_{\mathsf{NIZK}}$. In particular in this experiment, the random oracle H is implemented as follows:

1. Sample $v \leftarrow \{0,1\}^{\ell}$
2. The first r queries by the honest prover $Q_1, Q_2, \cdots Q_r$ (where each $Q_i = (\vec{a}, e_i, z_i)$ as generatred by P) will receive v as a response, i.e. $H(Q_1) = H(Q_2) = \cdots = H(Q_k) = v$
3. Emulate H as a random oracle 'honestly' for every other query

This hybrid differs from the last in that here the prover P will terminate after the first r queries it makes to H, whereas in $\mathcal{H}_1$ since H is not programmed to shortcut to a multicollision, P will have to 'work' to find a multicollision. Since the difference in running time of $\mathcal{H}_2$ and $\mathcal{H}_1$ is invisible to a distinguisher and $\boldsymbol{a}$ are generated identically in both hybrids, the only component that remains to be analyzed is $\boldsymbol{e}$ (since $\boldsymbol{z}$ is implicitly fixed by $w, \boldsymbol{a}, \boldsymbol{e}$). In $\mathcal{H}_1$, $\boldsymbol{e}$ represents the indices of the first multicollision found by P relative to H. Since P steps through pre-images uniformly at random and H is a random oracle (i.e. H has independent uniformly random outputs for every pair of distinct inputs) $\boldsymbol{e}$ is distributed uniformly in $\{0,1\}^{t\times r}$ in $\mathcal{H}_1$. In $\mathcal{H}_2$, $\boldsymbol{e}$ is clearly uniformly distributed in $\{0,1\}^{t\times r}$ as it corresponds to the first r challenges tried by P, which are sampled uniformly and independently.

As $\boldsymbol{a}, \boldsymbol{e}, \boldsymbol{z}$ are distributed identically in $\mathcal{H}_2$ and $\mathcal{H}_1$, the only distinguishing event corresponds to the programming of H, i.e. if the adversary is able to query H on some index that $\mathcal{H}_2$ subsequently programs to a different value. Since $\boldsymbol{a}$ has at least λ bits of entropy and is a prefix for all queries programmed in $\mathcal{H}_2$, this distinguishing event happens with probability no greater than $Q/2^{\lambda}$, where Q is the number of queries made by the adversary to the random oracle.

Hybrid $\mathcal{H}_3$. Replace the role of P in generating $(\boldsymbol{a}, \boldsymbol{e}, \boldsymbol{z})$ by Steps 1 and 2 of $\mathcal{S}_{\mathsf{NIZK}}$. In particular while $\mathcal{H}_2$ computes $\boldsymbol{a}, \mathsf{state} \leftarrow P_{\Sigma,a}(w)$, samples each challenge $e_i \leftarrow \{0,1\}^{t\times r}$, and produces each $z_i \leftarrow P_{\Sigma,z}(\mathsf{state}, e_i)$, this hybrid simply computes $(\boldsymbol{a}, \boldsymbol{e}, \boldsymbol{z}) \leftarrow \mathcal{S}_{\mathsf{NIZK}}(x)$. This modification still retains perfect correctness, as $\mathcal{H}_2$ already programs H to 'shortcut' to a multicollision upon being queried on each $(\boldsymbol{a}, e_i, z_i)$ produced. Indistinguishability of $(\boldsymbol{a}, \boldsymbol{e}, \boldsymbol{z})$ produced in $\mathcal{H}_3$ and $\mathcal{H}_2$ directly follows from r-simulatability of the Sigma protocol; there is a trivial lossless reduction to translate a distinguisher for $\mathcal{H}_3$ and $\mathcal{H}_2$ to a distinguisher for r-simulatibility of the Sigma protocol.

The final hybrid experiment $\mathcal{H}_2$ implements the simulator $\mathcal{S}_{\mathsf{NIZK}}$ in its entirety, and does not take the witness w as an input. As we show that for any $(x, w) \in R$, it holds that $\mathsf{P}^H(w, x) \equiv \mathcal{H}_1(w, x) \approx \mathcal{H}_3(x) \equiv \mathcal{S}_{\mathsf{NIZK}}(x)$, zero-knowledge of π_{NIZK} is hence proven.

□

D Strongly r-special Sound Schnorr

It is easy to modify Schnorr's proof of knowledge of discrete logarithm protocol [Sch91] to an r-special sound Sigma protocol with $r-1$-simulatability. This is achieved (in spirit) by instantiating the batched Schnorr protocol of Gennaro et al. [GLSY04] where one 'batches' $r-1$ random instances with the given instance. Intuitively in order to prove knowledge of the discrete log $x \in \mathbb{Z}_q$ of a public $X \in \mathbb{G}$ (where $\mathbb{G}$ is say an elliptic curve group), the prover samples a random degree $r-1$ polynomial $f \in \mathbb{Z}_q[X]$ such that $f(0) = x$, and publishes $\boldsymbol{a} = (f(i) \cdot G)_{i \in [r-1]}$. Given a challenge $e \in \mathbb{Z}_q^*$, the prover reveals $f(e)$, which the verifier can check is indeed the discrete logarithm of $F(e)$ by interpolation in the exponent, where $F \in \mathbb{G}[X]$ is the degree $r-1$ polynomial such that $F(0) = X$ and $\{F(i) = \boldsymbol{a}_i\}_{i \in [r-1]}$.

We give the protocol $\Sigma_{r,\mathsf{DL}}$ in Figure 14.

Protocol $\Sigma_{r,\mathsf{DL}}$

The prover $\mathsf{P} = (P_{\Sigma,a}, P_{\Sigma,z})$ and verifier V are both given public parameters $(\mathbb{G}, G, q)$, $r \in \mathbb{Z}$, and the statement $X = x \cdot G$. The prover additionally has witness x as private input.

$P_{\Sigma,a}(X, x)$:
1. Sample $r-1$-degree polynomial $f \in \mathbb{Z}_q[X]$ such that $f(0) = x$
2. Compute commitment $\boldsymbol{a} = (f(i) \cdot G)_{i \in [r-1]}$, and set $\mathsf{state} = f$
3. Output $(\mathsf{state}, \boldsymbol{a})$

$P_{\Sigma,z}(\mathsf{state}, e)$:
1. Parse $e \in \mathbb{Z}_q^*$ and output $f(e)$

$\mathsf{V}(X, \boldsymbol{a}, e, z)$:
1. Parse $a_1, a_2, \cdots, a_r = \boldsymbol{a}$
2. Define degree $r-1$ polynomial $F \in \mathbb{G}[X]$ such that $F(0) = X$ and $F(i) = a_i$
3. Output $F(e) \overset{?}{=} z \cdot G$

Fig. 14: r-special sound proof of Discrete Log

Theorem 11. *The protocol $\Sigma_{r,\mathsf{DL}}$ is a strongly r-special sound Sigma protocol for the language DLog.*

Proof. Completeness is easy to verify. $r-1$-simulatability and r-special soundness are discussed below.

$r-1$-*simulatability.* Transcript $\boldsymbol{a}, (z_i)_{i \in [r-1]}$ can be simulated given $X = g^x$ and $e_1, e_2, \cdots, e_r - 1$ as follows:
1. Sample $z_i \leftarrow \mathbb{Z}_q$ and compute $Z_i = z_i \cdot G$ for each $i \in [r-1]$

2. Define degree $r-1$ polynomial $F \in \mathbb{G}[X]$ such that $F(0) = X$ and $F(e_i) = Z_i$
3. Compute $\boldsymbol{a} = \{F(i)\}_{i\in[r-1]}$
4. Output $\boldsymbol{a}, (z_i)_{i\in[r-1]}$

The real prover samples f by choosing $\{f(i)\}_{i\in[r-1]}$ uniformly, and publishes $\boldsymbol{z} = \{f(e_i)\}_{i\in[r-1]}$ which is effectively uniform in $\mathbb{Z}_q^r$. The simulator chooses uniform $\boldsymbol{z} = \{f(e_i)\}_{i\in[r-1]}$ directly, and so $\boldsymbol{z}$ is distributed identically in both executions. As $\mathbb{Z}_q$ is isomorphic to $\mathbb{G}$, the $\boldsymbol{a}$ values are fixed given $X, \boldsymbol{z}$, which accounts for all components in the view and proves that the simulated and real values are identically distributed.

Strong r-special soundness. Given r accepting transcripts (ie. correct polynomial evaluations), by the facts that there can exist at most one $r-1$-degree polynomial passing through r points that $\mathbb{Z}_q$ and $\mathbb{G}$ are isomorphic, the points $(e_1, z_1), (e_2, z_2), \cdots, (e_r, z_r)$ fully specify $f \in \mathbb{Z}_q[X]$ such that $\{f(i)\cdot G = a_i\}_{i\in[r-1]}$ and $f(0)\cdot G = X$. Therefore x is given by $f(0)$. Note that 'strong' special soundness is achieved trivially as there is a unique z that satisfies any challenge e.

$\square$

Efficiency. A single instance of the strongly r-special sound Schnorr is equivalent in bandwidth, proving, and verification cost to r copies of the regular 2-special sound Schnorr Sigma protocol. However each new prover response requires a factor of r more $\mathbb{Z}_q$ (scalar) multiplications to compute than a single copy of the regular 2-special sound Schnorr Sigma protocol. Our analysis, however, focuses on minimizing the number of hash queries to the random oracle.

E Query Complexity Bounds

We restate and prove Lemma 3 and Corollary 1.

Lemma 5. *If (P, V) is a straight-line extractable NIZKPoK scheme for language L in the random oracle model with the following characteristics for security parameter λ:*

– *Perfect zero-knowledge simulator* Sim
– *ℓ-bit output random oracle H*
– *P queries made by* P *to H in generating a proof*
– *Probability $p_C > 0$ of producing an accepting proof*
– *V queries made by deterministic* V *to H in verifying a proof, is a strict subset of the queries made by* P
– *Non-programming extractor* Ext *with error $\le 2^{-\lambda}$ for an adversary that makes $\le V$ queries to the random oracle*

Then it must hold that:

$$\binom{P}{V} \ge \frac{p_C}{2^{-\lambda} + \varepsilon_\lambda}$$

Proof. The idea is to show that if $\binom{P}{V}$ is too small, then a malicious prover can succeed in producing a verifying proof by just guess the queries that V would make in verifying a proof, and simulating the remaining ones. This means an extractor should be able to produce a witness using just the queries that V makes (since those are the only queries that this malicious prover P makes) and this contradicts the hardness of the language.

We begin by constructing a new Prover algorithm P' which internally runs P, but simulates most of the random oracle calls for P and only makes a total of V external calls to the real oracle H: $\mathsf{P}'^H(x, w, \mathsf{P})$:

1. Sample a set of *indices* $Q \subset [1, \ldots, P]$ such that $|Q| = V$
2. Define oracle $H'(v)$ as follows:
   – If this is the i^{th} invocation of the oracle and if $i \in Q$ then return $H(v)$
   – Otherwise return a uniform $\{0,1\}^\ell$
3. Obtain $\pi \leftarrow \mathsf{P}^{H'}(x, w)$ and output π

Let $\mathcal{Q}_P$ represent the queries to H' made by P. Assuming no redundant queries in $\mathcal{Q}_P$, we note that H' agrees with H on V randomly chosen queries, and the two are completely independent on all other inputs.

By completeness of (P, V), it holds with probability p_C that $\mathsf{V}^{H'}(x, \pi) = 1$. Our goal is to instead analyze the probability that $\mathsf{V}^H(x, \pi)$ accepts, i.e., the verifier who makes queries to the real external oracle H accepts π. Denote the queries made by V to H' as $\mathcal{Q}_V$. Given that $\mathcal{Q}_V \subset \mathcal{Q}_P$, and that H' agrees with H on V values,

$$\begin{aligned}\Pr\left[\mathsf{V}^H(x,\pi) = 1 : \pi \leftarrow \mathsf{P}'^H(x, w, \mathsf{P})\right] &= \Pr\left[\mathsf{V}^H(x,\pi) = 1 : \pi \leftarrow \mathsf{P}^H(x, w)\right] \\ &\quad \cdot \Pr[H'(x) = H(x), \forall x \in \mathcal{Q}_V] \\ &\geq p_C \cdot \binom{P}{V}^{-1}\end{aligned}$$

Recall that the extractor's error (in this case $2^{-\lambda}$) represents the difference between the probability that a malicious prover P^* is able to produce a proof π, and the probability that the extractor Ext is able to produce a witness w for x when given the proof π and list of queries made by P^* in its production. Note that P' only queries $\mathcal{Q}_V$ to H, and so the set $\mathcal{Q}_V$ fully characterizes the list of queries made by the malicious prover. We therefore determine that:

$$\Pr[w \leftarrow \mathsf{Ext}(x, \pi, \mathcal{Q}_V) : \pi \leftarrow \mathsf{P}'^H(x, w, \mathsf{P})]$$

$$\geq \Pr\left[\mathsf{V}^{H'}(x,\pi) = 1 : \pi \leftarrow \mathsf{P}'^H(x, w, \mathsf{P})\right] - 2^{-\lambda}$$

$$\geq p_C \cdot \binom{P}{V}^{-1} - 2^{-\lambda}$$

As a final step, we replace $\pi \leftarrow \mathsf{P'}^H(x, w, \mathsf{P})$ by $(\pi, H) \leftarrow \mathsf{Sim}(x)$ to remove reliance on the witness w. Note that these two distributions of (π, H) are identical due to the fact that when $\mathsf{P'}$ outputs a proof, it is identically distributed to the output of honest P, and that the perfect simulation is distributed identically to the output of honest P. The set $\mathcal{Q}_V$ is fully specified by x, π, H as we show below.
$\mathcal{A}(x)$:

1. Compute $(\pi, H) \leftarrow \mathsf{Sim}(x)$
2. Construct $\mathcal{Q}_V$ by collecting the queries to H made by $\mathsf{V}^H(x, \pi)$
3. Output $\pi, \mathcal{Q}_V$

Firstly due to the perfect simulation we note that

$$\begin{aligned}\Pr[w \leftarrow \mathsf{Ext}(x, \pi, \mathcal{Q}_V) : (\pi, \mathcal{Q}_V) \leftarrow \mathcal{A}(x)] &= \Pr[w \leftarrow \mathsf{Ext}(x, \pi, \mathcal{Q}_V) : \pi \leftarrow \mathsf{P'}^H(x, w, \mathsf{P})] \\ &\geq p_C \cdot \binom{P}{V}^{-1} - 2^{-\lambda}\end{aligned}$$

Second we note that $w \leftarrow \mathsf{Ext}(x, \pi, \mathcal{Q}_V) : (\pi, \mathcal{Q}_V) \leftarrow \mathcal{A}(x)$ constitutes a PPT adversary that finds a witness for any $x \in L$. Since L has a hard instance generator $\mathcal{I}$ that admits a maximum advantage of ε_λ, for $x \leftarrow \mathcal{I}(\lambda)$ it holds that

$$\varepsilon_\lambda \geq \Pr[w \leftarrow \mathsf{Ext}(x, \pi, \mathcal{Q}_V) : (\pi, \mathcal{Q}_V) \leftarrow \mathcal{A}(x)] \geq p_C \cdot \binom{P}{V}^{-1} - 2^{-\lambda}$$

Rearranging, we have that

$$\binom{P}{V} \geq \frac{p_C}{2^{-\lambda} + \varepsilon_\lambda}$$

and this proves the lemma. □

We can use the above lemma to derive the optimal prover query complexity for proofs that are non-trivially secure, i.e. when $V \ll \binom{P}{V}$. We define $P_{\mathsf{OPT}}(V)$ to be the smallest prover query complexity for a given verifier query complexity V.

Corollary 4. *If* (P, V) *is a perfectly complete straight-line extractable NIZKPoK scheme for a* ε_λ*-hard language* L *in the random oracle model with all the characteristics required by Lemma 3 with the additional constraint that* $V < \lambda$ *and* $2^{-\lambda} \gg \varepsilon_\lambda$*, then the optimal prover query complexity is given by:*

$$P_{\mathsf{OPT}}(V) \approx \left(V! \cdot 2^\lambda\right)^{\frac{1}{V}}$$

Proof. As $2^{-\lambda} \gg \varepsilon_\lambda$, we make the approximation $2^{-\lambda} + \varepsilon_\lambda \approx 2^{-\lambda}$. From Lemma 3 we have that P_{OPT} is the smallest P such that $\binom{P}{V} \geq 2^\lambda$ since $p_C = 1$. Simplifying, we have that:

$$\begin{aligned} 2^\lambda &\leq \binom{P}{V} \\ 2^\lambda \cdot V! &= \prod_{i \in [0,V)} (P_{\mathsf{OPT}} - i) \\ &\approx (P_{\mathsf{OPT}})^V \end{aligned}$$

Upon rearranging the terms, we get the statement of the corollary. □

F Proof of PolyEval

Theorem 12. *Given a prime q, degree n polynomial $f \in \mathbb{Z}_q[X]$, and primitive k^{th} root of unity $\omega \in \mathbb{Z}_q$, Algorithm* PolyEval *outputs a list of k distinct points that lie on f at a cost of $k^2+n+2\log k$ multiplications and $k(k-1)+n$ additions in $\mathbb{Z}_q$.*

Proof. We begin by showing correctness. It suffices to show that for any $\alpha \in \mathbb{Z}_q^*$, the corresponding polynomial h agrees with f on the points $\{\alpha \cdot \omega^j\}_{j\in[0..k-1]}$. First we establish that $f(x) = \sum_{i\in[0..k-1]} x^i f_i(x^k)$ for every $x \in \mathbb{Z}_q$—this follows from the definition of f_i. Next we use the fact that ω is a k^{th} root of unity to simplify the expansion of $f(\alpha \cdot \omega^j)$ as follows:

$$\begin{aligned} f(\alpha \cdot \omega^j) &= \sum_{i\in[0..k-1]} (\alpha \cdot \omega^j)^i f_i((\alpha \cdot \omega^j)^k) = \sum_{i\in[0..k-1]} (\alpha \cdot \omega^j)^i f_i(\alpha^k) \\ &= \sum_{i\in[0..k-1]} (\alpha \cdot \omega^j)^i \vec{\alpha}_i = h(\alpha \cdot \omega^j) \end{aligned}$$

Now we count the number of multiplications in $\mathbb{Z}_q$ used by PolyEval. Step 3 requires computing α^k ($2\log k$ multiplications by repeated squaring) and evaluating k degree n/k polynomials. Assuming we naively make use of Horner's rule (n/k multiplications and as many additions per polynomial), it costs n multiplications and n additions in $\mathbb{Z}_q$ to evaluate these polynomials, for a total of $n + 2\log k$ $\mathbb{Z}_q$ multiplications and n additions induced by Step 3. Finally, in Step 6 we require k multiplications to generate each $\alpha \cdot \omega^i$, and we can evaluate the degree $k-1$ polynomial h at k points using Horner's rule, bringing the cost for this step to k^2 multiplications and $k(k-1)$ additions in $\mathbb{Z}_q$. Across all steps, the total number of operations required are $k^2 + n + 2\log k$ multiplications, and $k(k-1) + n$ additions in $\mathbb{Z}_q$. This proves the theorem. □

Lemma 6. *The probability that m independent invocations of* PolyEval *with the same polynomial $f \in \mathbb{Z}_q[X]$ and parameter k will output fewer than $m \cdot k$ distinct points (i.e. repeat at least one point) is at most $m^2k/2q$*

Proof. In the event of a repetition, two independent invocations sample α and α' that induce at least one common point, i.e. $\alpha \cdot \omega^i = \alpha' \cdot \omega^j$ for some $i, j \in [k]$. Rearranging the terms, we see that it must be the case that the ratio α/α' is an integer power of ω. Note that there are exactly k integer powers of ω in $\mathbb{Z}_q$, i.e. the multiplicative subgroup that it generates. For any fixed $x \in \mathbb{Z}_q^*$, the probability that a uniformly chosen $y \in \mathbb{Z}_q$ is such that the ratio y/x lands in this subgroup is k/q.

If we denote α_i as the α value sampled by the i^{th} invocation of PolyEval and correspondingly $\vec{A}_i = \{\alpha_i \cdot \omega^j\}_{j \in [0..k-1]}$, we can therefore bound the event of a repetition as follows:

$$\begin{aligned}
\Pr[\exists i, j \in [m] : i \neq j, \vec{A}_i \cap \vec{A}_j \neq \varnothing] &= \Pr\left[\bigvee_{i,j \in [m]} \vec{A}_i \cap \vec{A}_j \neq \varnothing\right] \\
&\leq \sum_{i \in [m-1]} \sum_{j \in [i+1..m]} \Pr[\vec{A}_i \cap \vec{A}_j \neq \varnothing] \\
&\leq \sum_{i \in [m-1]} \sum_{j \in [i+1..m]} \frac{k}{q} \leq \frac{m^2 k}{2q}
\end{aligned}$$

This proves the lemma. □

Corollary 5. *Given a parameter λ, if $q \in \Omega(2^\lambda)$ and $m, k \in \mathsf{poly}(\lambda)$, the probability that m independent invocations of PolyEval with the same polynomial will result in a redundant evaluation is negligible in λ.*

G Signature Aggregation Parameters

Define the relation R_{Agg} as:

$$R_{\mathsf{Agg}} = \{(x, w) \mid x = (\mathsf{pk}_1, m_1, \dots, \mathsf{pk}_n, m_n), w = (s_1, \dots, s_n), \\ \mathsf{Verify}(m_i, \mathsf{pk}_i, s_i) = \mathsf{true} \text{ for } \forall i \in [n]\}$$

i.e. each $s_i \in \mathbb{Z}_q$ is a signature on message $m_i \in \{0,1\}^*$ under Schnorr public key $\mathsf{pk}_i \in \mathbb{G}$, as per the Schnorr Verify algorithm.

Theorem 13. *Protocol π_{Aggr} is a proof of knowledge for the relation R_{Agg} with straight-line extraction in the random oracle model.*

Proof. (Sketch) We know from [CGKN21, Theorem 1] that the underlying Sigma protocol is n-special sound, which implies that once a malicious prover has queried n accepting transcripts to the random oracle, the entire witness can be extracted. It therefore suffices to analyze the smallest ℓ that guarantees that a cheating prover is unable find an r-collision within $\leq n$ queries except with probability $2^{-\lambda}$. The number of events (i.e. assignments of random oracle outputs) in which the first n queries to an ℓ-bit random oracle contain an r-collision is at most:

$$\binom{n}{r} \cdot 2^\ell \cdot (2^\ell)^{(n-r)}$$

Here $\binom{n}{r}$ counts the number of combinations of indices to 'plant' an r-collision, there are 2^ℓ values that the collision can take, and there are $(2^\ell)^{(n-r)}$ assignments of the remaining $n-r$ indices. This term is not tight since we double-count $r+1$ collisions, triple count $r+2$ collisions, etc. but their impact is minimal. Since there are a total of $2^{n\ell}$ equally likely possible output assignments to n random oracle queries, we have that:

$$\Pr[r\text{-collision within the first } n \text{ steps}] \leq \frac{\binom{n}{r} \cdot 2^\ell \cdot (2^\ell)^{(n-r)}}{2^{n\ell}}$$

It remains to examine the constraint on ℓ that will induce the above probability to be upper bounded by $2^{-\lambda}$:

$$\begin{aligned}
\frac{\binom{n}{r} \cdot 2^\ell \cdot (2^\ell)^{(n-r)}}{2^{n\ell}} &\leq 2^{-\lambda} \\
\binom{n}{r} 2^{\ell(1+n-r-n)} &\leq 2^{-\lambda} \\
\frac{n^r}{r!} 2^{\ell(1-r)} &\leq 2^{-\lambda} \\
2^{\ell(1-r)} &\leq r! \cdot 2^{-(\lambda + r\log_2(n))} \\
&\leq 2^{-(\lambda + r\log_2(n) - \log_2(r!))} \\
2^{\ell(r-1)} &\geq 2^{\lambda + r\log_2(n) - \log_2(r!)} \\
\ell &\geq (\lambda + r\log_2(n) - \log_2(r!))/(r-1)
\end{aligned}$$

which is precisely the constraint adhered to by ℓ in π_{Aggr}. □

H Collisions Versus Inversions – Concrete Value Tables

I Other Applications of PolyEval

The algorithm PolyEval is generally useful in settings where one has to evaluate a degree n polynomial in $\mathbb{Z}_q$, where n ranges from say 2^5 to 2^{14}, and $q-1$ is 'slightly smooth', i.e. there are enough $k \approx \sqrt{n}$ values that divide $q-1$. Such settings include the base fields of common elliptic curves such as Curve25519 (discussed in this paper in the context of EdDSA), and secp256k1 (used by Bitcoin and others for ECDSA). We describe some of these settings where PolyEval can be relevant in this section.

Threshold Cryptography. A common method to protect signing/encryption keys is to distribute them across a number m of devices, so that reconstructing or operating with the key requires a threshold t of the devices to cooperate. This is typically done by using Shamir's secret sharing in the base field of the elliptic curve, i.e. defining a degree $t-1$ polynomial f such that $f(0) = \mathsf{sk}$ encodes the secret key, and each party P_i receives $f(i)$. When t is in the range of 2^5 to 2^{14}, PolyEval can speed up the generation of these shares for threshold versions of EdDSA and ECDSA keys.

n	r	Collision (This work)	Inversion	Improvement
1024	8	9.33×10^7	2.68×10^8	2.8
512	8	5.11×10^7	1.34×10^8	2.6
512	16	2.95×10^5	5.24×10^5	1.7
1024	32	3.55×10^4	6.55×10^4	1.8
256	16	1.57×10^5	2.62×10^5	1.6
512	32	1.86×10^4	3.28×10^4	1.7
128	16	8.36×10^4	1.31×10^5	1.5
256	32	9.80×10^3	1.64×10^4	1.6
32	8	2.53×10^6	8.39×10^6	3.3
64	16	2.38×10^4	6.55×10^4	2.7
128	32	5.19×10^3	8.19×10^3	1.5

Table 3: Prover/aggregator query complexity $\mathsf{T}_{\mathsf{Agg}}$ when using a collision based predicate to aggregate n signatures, as opposed to inversions (with a tighter parameterization than [CGKN21]), for a range of r parameters. Expected running times are derived analytically [vM39, Pre93]

Verifiable Secret Sharing and Beyond. There are numerous constructions to upgrade the security of secret sharing schemes to tolerate a malicious dealer and participants, i.e. *verifiable* secret sharing (VSS). Simple VSS schemes such as Feldman'sfor groups where the discrete logarithm assumption is assumed to hold form the basis for distributed key generation protocolsfor ECDSA/EdDSA. VSS can also form the basis for *verifiable encryption*, where a ciphertext can be verified to encrypt the discrete logarithm of a public point (say encrypt the secret component of an EdDSA/ECDSA public key), when it is combined with MPC-in-the-head techniques. In this case, the degree of the polynomial corresponds to the number of 'transcripts' that must be checked, which for a 128 or 256 bit security level falls within the previously mentioned range for which PolyEval provides significant savings.

J Where to Apply the Attack

Given the attack in Section 6.1, when is it safe to apply Fischlin's transformation? Recall that the security of Fischlin's transformation hinges on "quasi-unique responses" as in Definition 1.

We argue that ensuring this property in a larger system is not always straightforward for languages where the same statement can have multiple witnesses, even when no individual party has more than one witness. In particular, a larger cryptographic application that makes use of such proofs as subprotocols may rely on the ability of the same proof to be produced indistinguishably by different methods, for example by an honest party using a witness in the real protocol, and

n	r	Expected Memory Usage
1024	8	8.9GB
512	8	4.9GB
512	16	28.3MB
1024	32	3.4MB
256	16	15MB
512	32	1.7MB
128	16	8MB
256	32	0.9MB
32	8	242MB
64	16	2.2MB
128	32	0.5MB

Table 4: Prover/aggregator memory complexity when using a collision based predicate to aggregate n signatures for a range of r parameters, for a naive implementation. Derived analytically [vM39, Pre93]

by a simulator using a trapdoor in the ideal protocol. This subtlety is brought out in the following example protocol between Alice (who only has public input $B \in \mathbb{G}$) and Bob (who has private input $b \in \mathbb{Z}_q$):

– Alice samples $a \leftarrow \mathbb{Z}_q$, sets $A = g^a$, and computes π_A as the Witness Hiding PoK of $\mathsf{DLog}_g(A)$. Alice sends A, π_A to Bob.
– Bob and computes π_B as the WIPoK of $\mathsf{DLog}_g(A) \vee \mathsf{DLog}_g(B)$ using b as a witness. Bob sends B, π_B to Alice.

Fischlin's proof does not directly cover this use case, but it is suggested informally [Fis05, pg. 13] that their construction extends to logical compositions, etc. in the presence of a system parameter enforcing quasi unique responses.

When Alice is corrupt in the above protocol, her view can be simulated without knowledge of b. In particular the simulator simply extracts a from π_A and uses a as a witness to compute π_B as the WIPoK of $\mathsf{DLog}_g(A) \vee \mathsf{DLog}_g(B)$. This simulation is efficient due to extractability of the WHPoK, and will be indistinguishable from the real protocol by witness indistinguishability of the WIPoK. Simultaneously the discrete log b of B is efficiently extractable due to the witness hiding property of WHPoK (in conjunction with the hardness of the discrete logarithm problem) and the extractablity of the WIPoK. This template due to Feige and Shamir [FS90] was used by Pass [Pas03] to construct a deniable two round zero-knowledge argument in the random oracle model, where the simulator does not rely on programming the random oracle. As shown by Canetti *et al.* [CJS14] this allows for secure composition in the Global Random Oracle model.

Can we safely use Fischlin's transformation here? At first glance, the above protocol appears to be conducive to quasi-unique responses for the sigma protocols

r	Lower bound	This Work	Fischlin
4	1.34×10^{10}	1.34×10^{10}	2.43×10^{10}
5	1.75×10^{8}	1.76×10^{8}	3.36×10^{8}
6	9.97×10^{6}	1.02×10^{7}	2.00×10^{7}
7	1.32×10^{6}	1.40×10^{6}	2.73×10^{6}
8	2.93×10^{5}	3.26×10^{5}	6.23×10^{5}
9	9.25×10^{4}	1.08×10^{5}	2.01×10^{5}
10	3.71×10^{4}	4.55×10^{4}	8.19×10^{4}

Table 5: Prover work in our π_{NIZK} construction for NIZK given an $r+1$-special sound Sigma protocol, as a function of r for 130-bit security. Fixing the soundness error and the proof size (which is governed by r), this table of analytical estimates shows that our construction almost meets our lower-bound while using a factor of between $2\sqrt{2/\pi}$ and 2 fewer queries than Fischlin's transform.

that would underlie π_A as well as π_B. Indeed Alice only knows a[8] allowing B to be treated as a system parameter if Alice is corrupt, and Bob only knows b which allows A to be considered a system parameter when Bob is corrupt, therefore neither party has the ability to efficiently compute multiple accepting responses for the same challenge in $\Sigma^{\vee}_{\mathsf{DL}}$.

However this scenario structurally resembles $\mathsf{Expt}^{\mathsf{DL\text{-}WI}}_{\mathcal{A},\mathsf{P}}$, i.e. since our attack on WI for Fischlin's transformation does not require knowledge of both witnesses, it can be applied here. In particular Alice knows a, and so can test whether the proof π_B was computed using a or b as the witness. This allows her to distinguish between the real protocol (where Bob uses b as the witness to compute π_B) and the simulation (where π_B is generated by the simulator using a as the witness).

We therefore make the case that a cleaner definition is required, ideally one that does not require reasoning about the context in which a sigma protocol is used.

K A Note on Exact Versus 'Near' Inversions

The version of the transformation described here is referred to as the 'basic' one by Fischlin. They proceed to tweak the Verifier to accept 'near' inversions, where it is sufficient for the Prover to output transcripts $\tau_1, \cdots, \tau_r$ such that $H(\tau_i)$ is interpreted as a positive integer and $\sum_i H(\tau_i) < S$ for some parameter $S \approx r$. The purpose of this change is to reduce the completeness error for the Prover (by increasing the soundness error). Our discussion on quasi-unique responses is unaffected by this change as the Prover is still deterministic and the same

[8]We ignore the prospect of obtaining auxiliary information about b, for eg. b could be sampled uniformly as part of a larger protocol.

vulnerability persists. Regarding Prover query complexity, it is already pointed out in [Fis05] that relaxing this requirement for an accepting proof increases the soundness error, and adjusting the hash function parameter ℓ to retain the same r, λ values results in an increase in the expected Prover query complexity. Consequently we do not discuss the near-inversion variant further in this paper, and every reference to Fischlin's construction will pertain to the basic exact inversion predicate.