# Time- and Space-Efficient Arguments from Groups of Unknown Order

Alexander R. Block[1](✉), Justin Holmgren[2], Alon Rosen[3], Ron D. Rothblum[4], and Pratik Soni[5]

[1] Purdue University, West Lafayette, USA
block9@purdue.edu
[2] NTT Research, Sunnyvale, CA, USA
justin.holmgren@ntt-research.com
[3] IDC Herzliya, Herzliya, Israel
alon.rosen@idc.ac.il
[4] Technion, Haifa, Israel
rothblum@cs.technion.ac.il
[5] Carnegie Mellon University, Pittsburgh, USA
psoni@andrew.cmu.edu

**Abstract.** We construct public-coin time- and space-efficient zero-knowledge arguments for **NP**. For every time $T$ and space $S$ non-deterministic RAM computation, the prover runs in time $T \cdot \mathrm{polylog}(T)$ and space $S \cdot \mathrm{polylog}(T)$, and the verifier runs in time $n \cdot \mathrm{polylog}(T)$, where $n$ is the input length. Our protocol relies on hidden order groups, which can be instantiated with a trusted setup from the hardness of factoring (products of safe primes), or without a trusted setup using class groups. The argument-system can heuristically be made non-interactive using the Fiat-Shamir transform.

Our proof builds on DARK (Bünz et al., Eurocrypt 2020), a recent succinct and efficiently verifiable *polynomial commitment scheme*. We show how to implement a variant of DARK in a time- and space-efficient way. Along the way we:

1. Identify a significant gap in the proof of security of DARK.
2. Give a non-trivial modification of the DARK scheme that overcomes the aforementioned gap. The modified version also relies on significantly weaker cryptographic assumptions than those in the original DARK scheme. Our proof utilizes ideas from the theory of integer lattices in a novel way.
3. Generalize Pietrzak's (ITCS 2019) proof of exponentiation (PoE) protocol to work with general groups of unknown order (without relying on any cryptographic assumption).

In proving these results, we develop general-purpose techniques for working with (hidden order) groups, which may be of independent interest.

## 1 Introduction

Significant overhead in prover efficiency is the main roadblock standing between zero-knowledge proofs and widespread deployment. While there has been exten-

T. Malkin and C. Peikert (Eds.): CRYPTO 2021, LNCS 12828, pp. 123–152, 2021.
https://doi.org/10.1007/978-3-030-84259-8_5

sive work on optimizing the *running time* of the prover, much less attention has been drawn to the *space (or memory) usage*. In particular, most protocols in the literature suffer from the drawback that memory consumption by the prover is exceedingly large: computations that take time $T$ and space $S$ to compute directly, require the prover to invest $\Omega(T)$ *space* in order to prove correctness (with some notable exceptions [9–11,13,31,44] to be discussed shortly). Moreover, due to the way that modern memory architectures work, large memory usage also inevitably leads to more cache misses and slower runtime. Thus, space efficiency of the prover is a severe bottleneck to enabling zero-knowledge proofs for large-scale complex computations.

The recent work of Block et al. [11] constructed the first *publicly verifiable*[1] zero-knowledge proofs (under standard cryptographic assumptions) in which the prover is efficient both in terms of time and space. In more detail, for every **NP** relation $R$, for which membership can be computed in time $T$ and space $S$, the prover (given as input the instance and corresponding witness) can be implemented in time $T \cdot \text{poly}(\lambda, \log T)$ and space $S \cdot \text{poly}(\lambda, \log T)$ and the verifier can be implemented in time roughly $T \cdot \text{poly}(\lambda, \log T)$, where here and throughout this work $\lambda$ denotes the security parameter. The fact that verification takes $\Omega(T)$ time is a significant drawback of this protocol and precludes applications like delegation of computation.

### 1.1 Our Results

In this work we overcome the main disadvantage of the work of Block et al. by constructing zero-knowledge proofs with a time- and space-efficient prover *and* poly-logarithmic verification. For this result we rely on groups of unknown order, which are discussed immediately after the statement of Theorem 1.1.

**Theorem 1.1 (Informally Stated, see Theorem 4.1).** *Assume that there exists a group for which the hidden order assumption holds. Then, every* **NP** *relation that can be verified by a time $T$ and space $S$ RAM machine has a public-coin zero-knowledge argument-system in which the prover, given as input the instance $x$ and witness $w$, runs in time $T \cdot \text{poly}(\lambda, \log T)$ and uses space $S \cdot \text{poly}(\lambda, \log T)$. The verifier runs in time $|x| \cdot \text{poly}(\lambda, \log T)$, the communication complexity is $\text{poly}(\lambda, \log T)$ and the number of rounds is $O(\log T)$.*

*The argument-system uses a* common reference string*, which is simply a description of the hidden-order group $\mathbb{G}$ and a random element $g \in \mathbb{G}$.*

As usual, the protocol can heuristically be made *non-interactive* by applying the Fiat-Shamir [29] transform. It is also worth noting that a result similar to Theorem 1.1 was not known even *without* the zero-knowledge requirement.

[1] Public verifiability has emerged as a central requirement for proof-systems. In a nutshell it means that anyone who possesses the proof-string can verify its correctness (while possibly also requiring access to a common reference string). We mention that time- and space-efficient protocols that are either *privately-verifiable* or based on non-standard computational assumptions were previously known. See Sect. 1.2.

As for the assumption that we use, the *hidden order assumption* for a group $\mathbb{G}$ states that given a random group element $g \in \mathbb{G}$ it is computationally infeasible to find (any multiple of) the order of $g$. For example, assuming the hardness of factoring $N$ which is a product of two safe primes, the group $\mathbb{Z}_N^*$, is a hidden order group. Therefore, our scheme can be instantiated assuming the hardness of factoring (products of safe primes).

Lately there has been much interest in *public-coin* hidden order groups which means that the description of the group can be generated without a trusted party (aka a *transparent* setup). This is not known for the factoring based group (since one needs to be able to generate a hard instance for factoring without using private coins). However, as pointed out in [21,26], class groups of an imaginary quadratic field are a candidate public-coin hidden order group. Since our common reference string only includes a description of the group and a random element, using class groups we obtain a protocol that does not require a trusted setup.

*Time- and Space-efficient Polynomial Commitments.* Theorem 1.1 is derived from a new polynomial-commitment scheme that we construct, based on a prior scheme due to Bünz et al. [21]. Roughly speaking, a *polynomial-commitment scheme* allows Alice to commit to a low degree polynomial $P$ so that later Bob can ask her for evaluations $P(x)$ along with proofs that the supplied values are indeed consistent with her commitment (see Sect. 3.4 for the formal definition). Polynomial commitments have drawn significant attention recently (see Sect. 1.2), especially due to their use in compiling ideal model information-theoretic proof-systems into real-world protocols. Most works use polynomial commitments in order to obtain shorter proof sizes. In contrast, following [11], we use polynomial commitments to enable a small space (and time) implementation of the prover. We believe that this aspect of polynomial commitments will be a key enabler of large-scale zero-knowledge proofs.

For simplicity, and since it suffices for proving Theorem 1.1, we focus on polynomial commitments for *multilinear*[2] polynomials $P : \mathbb{F}^n \to \mathbb{F}$, where $\mathbb{F}$ is a prime order field. Following [11], we consider polynomial commitments in a streaming model, in which the committer are given *(multi-pass) streaming access* to the representation of the polynomial; in our case, the restriction of the multilinear polynomial to the Boolean hypercube. This streaming model is motivated by the fact that when using the commitment scheme to construct an efficient argument-system, the prover commits to a transcript of the computation - which can indeed be generated in a streaming manner in small space.

In order to construct their time- and space-efficient arguments, [11] first construct a polynomial commitment scheme for multilinear polynomials in which the prover runs in quasi-linear time (in the description of the polynomial, which is of size $2^n \cdot \log(|\mathbb{F}|)$) and logarithmic space. However, the *verifier* for their evaluation proof also runs in time that is linear in the size of the polynomial. This is the core reason that the argument-system constructed in [11] does not achieve *sub-linear* verification. In contrast, we give a polynomial commitment scheme in which the prover is time- and space-efficient and verification *is* poly-logarithmic.

[2] Recall that a multi-variate polynomial is multilinear if its degree in each variable is at most 1.

**Theorem 1.2 (Informally Stated, see Theorem 4.2).** *Assume that there exists a group for which the hidden order assumption holds. Then, there exists a polynomial commitment scheme for multilinear polynomials* $P : \mathbb{F}^n \to \mathbb{F}$ *over a prime order field* $\mathbb{F}$ *(of size* $|\mathbb{F}| \leqslant 2^{\mathrm{poly}(n)}$*) with the following efficiency properties:*

1. *Commitment and evaluation proofs can be computed in time* $2^n \cdot \mathrm{poly}(n, \lambda)$ *and space* $n \cdot \mathrm{poly}(\lambda)$*, given multi-pass streaming access to the evaluations of* $P$ *on the Boolean hypercube.*
2. *The communication complexity and verification time are both* $\mathrm{poly}(n, \lambda)$.

*Similarly to Theorem 1.1, the commitment scheme is defined relative to a reference string containing the description of the hidden order group and a random group element.*

Theorem 1.1 follows from Theorem 1.2 using techniques from the work of Block et al. [11]. Namely, we use a time- and space-efficient *polynomial interactive oracle proof*[3] (polynomial IOP), constructed in [11] (based on the 2-prover MIP of [13]). We then compile this polynomial IOP into an argument-system using the polynomial commitment of Theorem 1.2 in the natural way: namely, rather than sending polynomials in the clear, the prover simply commits to them and later proves correctness of evaluations queries. This compilation results in a *succinct* argument, which can be made zero-knowledge (while preserving time- and space-efficiency) using standard techniques [4] (see [12] for details).

Our proof of Theorem 1.2 builds on a recent remarkable polynomial-commitment scheme called DARK (for Diophantine Argument of Knowledge), due to Bünz et al. [21]. This polynomial commitment scheme was the first such scheme to achieve logarithmic size proofs and verification time.

We make several significant improvements to the DARK scheme.

1. **Identifying and Bypassing a Gap in DARK:** We identify a gap in the security proof of [21]. We elaborate on this gap in Sect. 2.2. We emphasize that we do not know whether this gap can lead to an attack on the DARK scheme. Nevertheless, we find this gap to be significant and in particular we do not know how to fix their security proof. We mention that we have been informed [23] that the same gap was discovered independently by the authors of [21].

   To obtain our polynomial commitment scheme, we therefore make a non-trivial modification of the DARK scheme and show that this modification suffices to prove security. Our security proof relies on a new lemma on the existence of integral inverses for uniformly random rectangular binary matrices, which we prove. Our proof is based on ideas from the mathematical theory

[3] A polynomial IOP is defined similarly to a (public-coin) interactive proof, except that in every round the prover is allowed to send the truth table of a large polynomial, and the verifier can query a few points from each polynomial. The notion was proposed concurrently in [21] and [24]. Essentially the same notion appears also in [38] (called *Probabilistically Checkable Interactive Proof w.r.t. Encoded Provers* therein).

of integer lattices which, to the best of our knowledge, have not been used before in this context.[4] See Sect. 2.3 for details.

2. **Improved Assumptions and Simplicity:** Setting aside the gap in the security proof, we also significantly improve the assumptions that the DARK scheme relies on. The improvement in assumptions stems from a *simpler* (and conceptually more appealing) extraction procedure that we describe. This improvement applies to the two main variants of the DARK scheme. In more detail:
   (a) The first variant of the original DARK scheme uses RSA groups, while relying on the *strong RSA assumption* and the *adaptive root assumption*. The former assumption, while not new, is relatively strong, whereas the latter is a new assumption, due to Wesolowski [46], which is not yet well understood (note that both assumptions are known to hold in the generic group model [16,27]).

       In contrast, when instantiating our scheme in this setting, we only need to rely on the hardness of factoring (products of safe primes).

   (b) In order to obtain an *unstructured* common random string, Bünz et al. also give a construction that uses class groups of an imaginary quadratic field. This construction relies on both the aforementioned adaptive root assumption (for class groups) *and* a new assumption that they introduce on class groups called the 2-strong RSA assumption. The class-group based construction is also more complex than their construction using RSA groups.

       In contrast, our construction works equally well for both groups and we can instantiate it using class groups while assuming only the hidden order assumption (which is weaker than the adaptive root assumption [15]). See also [1,43] for a comparison between these assumptions.
3. **Small Space Polynomial Commitments:** We show that the commitment and evaluation protocols in (our variant of) the DARK scheme can be implemented in time roughly $\tilde{O}(2^n)$ (i.e., quasi-linear in the description of the polynomial) *and space* $\mathrm{poly}(n)$ (i.e., poly-logarithmic in the description), given (multi-pass) streaming access to the evaluations of the polynomial on the Boolean hypercube. Crucially, (and in contrast to the scheme of [11]) the verifier in our evaluation proofs runs in time $\mathrm{poly}(n)$. See Sect. 2.4 for the ideas underlying our space-efficient implementation.
4. **Statistical Proof of Exponentiation over General Groups:** We improve and generalize a recent elegant *proof-of-exponentiation* protocol due to Pietrzak [37]. In a proof of exponentiation protocol, the goal is for the prover to convince the verifier that the triplet $(g, h, T) \in \mathbb{G} \times \mathbb{G} \times \mathbb{N}$ satisfies the relation $h = g^{2^T}$, where $\mathbb{G}$ is a group of unknown order.[5] Pietrzak constructs such a protocol in which the prover runs in time roughly $T$ and the verifier

[4] We emphasize that we use lattice theory to show that our *group* based construction is secure. In particular all of our hardness assumptions are group based.

[5] Since the order of $\mathbb{G}$ is not known, one cannot simply compute $2^T$ modulo the group order and then exponentiate.

runs in time roughly $\log(T)$ (which is exponentially faster than the direct computation via repeated squaring). Pietrzak uses his protocol to construct a simple *verifiable delay function* [14], based on the Time-Lock puzzles of Rivest, Shamir and Wagner [39].

Pietrzak's protocol is designed specifically for the group $\mathsf{QR}_N^+$ of (signed) quadratic residues modulo an integer $N$, which is the product of two safe primes. Pietrzak [37, Section 6.1] points out that the protocol can also be extended to class groups, but with two caveats. First, this extension is only *computationally* sound and second, it requires an additional assumption from the class group (namely, that it is hard to find elements of small order). This is in contrast to Pietrzak's protocol for $\mathsf{QR}_N^+$ which provides statistical security and without relying on any assumption. We note that a different protocol, due to Wesolowski [46], gives a proof of exponentiation over groups in which the adaptive root assumption holds (which plausibly includes class groups), but also only achieves computational soundness and requires a (strong) hardness assumption. Wesolowski's protocol is used as sub-routine within the DARK scheme.

As an additional contribution, which we find to be of independent interest, we show a modification of Pietrzak's protocol that works over *general* groups of unknown order (including class groups) while preserving *statistical* security and without relying on any assumption. By replacing Wesolowski's protocol within the DARK scheme with our new extension, we obtain that the evaluation proofs for our polynomial commitment are *proofs* (rather than arguments) of knowledge.

### 1.2 Additional Related Works

*Polynomial Commitments.* Polynomial commitment schemes were introduced by Kate et al. [32]. As discussed above, such commitments allow one to commit to a polynomial and later answer evaluation queries while proving consistency with the commitment.

There are several variants of polynomial commitments include privately verifiable schemes [32,36], publicly-verifiable schemes with trusted setup [21], and zero-knowledge schemes [47]. More recently, much focus has been on obtaining publicly-verifiable schemes without a trusted setup [5,7,17,21,33,34,42,45,47,48]. In all but one prior work, the space complexity of the committer is proportional to the description size of the polynomial. The only exception is the aforementioned work of Block et al. [11] who build a commitment scheme for multilinear polynomials (based on [17,19]), where the committer's space complexity is *poly-logarithmic* in the description of the polynomial, assuming that the committer is given multi-pass streaming access to its description. As mentioned above, a key drawback of their work is that the verification is linear in the size of the polynomial.

Lastly, we mention that classical works on low degree testing (à la [40]) as well as more recent works [5,6,8] can be used to construct polynomial-commitments by Merkle hashing the entire truth table of the polynomial (and using a self-correction procedure or protocol).

*Privately Verifiable Proofs.* The question of constructing proof systems in which the prover is efficient both in terms of time *and* space was first raised by Bitansky and Chiesa [10], who constructed a time- and space-efficient (or in their terminology *complexity preserving*) interactive argument for any problem in **NP** based on *fully homomorphic encryption*. Holmgren and Rothblum [31] constructed *non-interactive* time- and space-efficient arguments for **P** based on the (sub-exponential) learning with errors assumption. The protocols of [10,31] are *privately verifiable*, meaning that only a designated verifier (who knows the randomness used to sampled the verifier messages) is able to verify the proof.

*Proofs by Recursive Composition.* An alternative approach to *publicly verifiable* time- and space-efficient arguments is by recursively composing SNARKs for **NP** [9,44]. Recursive composition requires both the prover and verifier to make non-black-box usage of an "inner" verifier for a different SNARK, leading to large computational overhead. Several recent works [18,20,25] attempt to solve the inefficiency problems with recursive composition, but at additional expense to the underlying cryptographic assumptions. In particular, these works rely on hash functions that are modeled as random oracles in the security proof, despite being used in a *non-black-box* way by the honest parties. Security thus cannot be reduced to a simple computational hardness assumption, even in the random oracle model. Moreover, the practicality of the schemes crucially requires usage of a novel hash function (e.g., Rescue [2]) with algebraic structure designed to maximize the *efficiency* of non-black-box operations. Such hash functions have endured far less scrutiny than standard SHA hash functions, and the algebraic structure could potentially lead to a security vulnerability.

We also mention a recent work of Ephraim et al. [28] which uses recursive composition to address the related question of implementing the prover in small *depth* (i.e., parallel time).

*Multi-Prover Proofs.* Block et al. [11] gave the first *publicly-verifiable* time- and space-efficient arguments for **NP** but as noted above (and in contrast to Theorem 1.1), the verification time is linear in the computation. Bitansky and Chiesa [10], as well as Blumberg et al. [13], construct time- and space-efficient *multi-prover* interactive proof, that is, soundness only holds under the assumption that the provers do not collude. Justifying this assumption in practice seems difficult and indeed multi-prover interactive proofs are usually only used as building blocks toward more complex systems.

### 1.3 Organization

We give overviews of our proof techniques in Sect. 2. Preliminaries are in Sect. 3. In Sect. 4 we formally state our results and in Sect. 5 we describe our polynomial commitment scheme. The rest of the technical sections are deferred to the full version [12].

## 2 Technical Overview

We start, in Sect. 2.1 with an exposition of (a variant of) the DARK polynomial commitment scheme of [21]. Then, in Sect. 2.2 we describe a gap in their security proof. In Sect. 2.3 we show how to modify their protocol in order to resolve this gap (and simultaneously simplify the extraction procedure and relax the cryptographic assumptions). Then, in Sect. 2.4 we describe our small space implementation and lastly, in Sect. 2.5 we describe our improved proof of exponentiation protocol.

### 2.1 Overview of the DARK Scheme

We start with an overview of the DARK polynomial commitment scheme. The main scheme constructed in [21] was for *univariate* polynomials. However, for our applications it will be more useful to consider a variant of their scheme for (multi-variate) *multilinear* polynomials.[6] We emphasize that the gap in the security proof (to be discussed shortly) also applies to the original DARK scheme.

*DARK Commitments: Encoding Polynomials by Large Integers.* Let $\mathbb{F} = \mathbb{F}_p$ be a finite field of prime order $p$. Recall that a multilinear polynomial $P : \mathbb{F}^n \to \mathbb{F}$ can be specified by its evaluations on the Boolean hypercube. Thus, in order to commit to the polynomial $P$, we will look at the sequence of values $(P(\mathbf{b}))_{\mathbf{b} \in \{0,1\}^n}$. In order to commit to this sequence Bünz et al. construct a large integer $\mathcal{Z}(P)$ that encodes it, by looking at this sequence as a base $q$ representation of an integer, for some $q \gg p$. That is, $\mathcal{Z}(P) = \sum_{\mathbf{b} \in \{0,1\}^n} q^{\mathbf{b}} \cdot P(\mathbf{b})$, where $\mathbf{b}$ is interpreted as an integer in the natural way.

The commitment to the polynomial $P$ is simply $c = g^{\mathcal{Z}(P)}$, where $g$ is a random element of the hidden-order group $\mathbb{G}$ specified as part of the $\mathsf{CRS}$. We say that the integer $Z$ is *consistent* with the multilinear polynomial $P$ if, looking at the base $q$ representation of $Z$, *and reducing each digit modulo* $p$, we get the sequence $(P(\mathbf{b}))_{\mathbf{b} \in \{0,1\}^n}$. Observe that since $q \gg p$, there are many integers $Z$ that are consistent with a given polynomial $P$ (where one of these integers is $\mathcal{Z}(P)$). Nevertheless, the commitment is *binding* since finding two different integers that are consistent with the same commitment reveals a multiple of the order of $g$, which we assumed is computationally infeasible.

We will rely on the fact that this commitment scheme is homomorphic, in the following sense: given integers $Z_1$ and $Z_2$ that are consistent with the polynomial $P_1$ and $P_2$, and have sufficiently small digits in their base $q$ representation, it holds that $Z_1 + Z_2$ is consistent with the polynomial $P_1 + P_2 \pmod{p}$. This is also true for scalar multiplication: if $\alpha$ is sufficiently small then $\alpha Z_1$ is consistent with $\alpha \cdot P_1 \pmod{p}$. However, the assumption that the digits are small is crucial for the homomorphisms to work, and jumping ahead, this will be the source of the gap in the proof.

[6] It is worth mentioning that [21] also present a variant of their scheme for multivariate polynomials. This variant is somewhat different from the one described here and is obtained via a reduction to the univariate case.

*Evaluation Proofs.* Suppose that the committer wants to prove that $P(\boldsymbol{\zeta}) = \gamma$, for some $\boldsymbol{\zeta} = (\zeta_1, \dots, \zeta_n) \in \mathbb{F}^n$ and $\gamma \in \mathbb{F}$. More precisely, we will show an interactive protocol that is a *proof of knowledge* of an integer $Z$ that is consistent with a polynomial $P$ such that that $C = g^Z$ and $P(\boldsymbol{\zeta}) = \gamma$.

Let $P_0, P_1 : \mathbb{F}^{n-1} \to \mathbb{F}$ be the $(n-1)$-variate polynomials defined as $P_0(\cdot) = P(0, \cdot)$ and $P_1(\cdot) = P(1, \cdot)$. The prover first generates these two polynomials, and the corresponding commitments $c_0 = g^{\mathcal{Z}(P_0)}$ and $c_1 = g^{\mathcal{Z}(P_1)}$. Also, let $\gamma_0 = P_0(\zeta_2, \dots, \zeta_n)$ and $\gamma_1 = P_1(\zeta_2, \dots, \zeta_n)$. As its first message, the prover sends $(c_0, c_1, \gamma_0, \gamma_1)$. The verifier now checks that:

1. $\gamma = \zeta_1 \cdot \gamma_1 + (1 - \zeta_1) \cdot \gamma_0$. This equation should indeed hold since $P(\boldsymbol{\zeta}) = \zeta_1 \cdot P_1(\zeta_2, \dots, \zeta_n) + (1 - \zeta_1) \cdot P_0(\zeta_2, \dots, \zeta_n)$.
2. The verifier also checks that $c_0 \cdot (c_1)^{q^{N/2}} = c$, where $N := 2^n$. This should hold since

$$c_0 \cdot (c_1)^{q^{N/2}} = g^{\mathcal{Z}(P_0)} \cdot g^{q^{N/2} \cdot \mathcal{Z}(P_1)} = g^{\mathcal{Z}(P_0) + q^{N/2} \cdot \mathcal{Z}(P_1)} = g^{\mathcal{Z}(P)},$$

where the last equality follows from the fact that

$$\begin{aligned} \mathcal{Z}(P_0) + q^{N/2} \cdot \mathcal{Z}(P_1) &= \sum_{\mathbf{b} \in \{0,1\}^{n-1}} q^{\mathbf{b}} \cdot P_0(\mathbf{b}) + q^{N/2} \cdot \sum_{\mathbf{b} \in \{0,1\}^{n-1}} q^{\mathbf{b}} \cdot P_1(\mathbf{b}) \\ &= \sum_{\mathbf{b} \in \{0,1\}^{n}} q^{\mathbf{b}} \cdot P(\mathbf{b}) = \mathcal{Z}(P), \end{aligned}$$

where the arithmetic is over the integers and we leverage the homomorphic properties of the commitment. Note that actually computing the value $(c_1)^{q^{N/2}}$ is too expensive for the verifier.[7] Thus, rather than computing it directly, this value is supplied by the prover who then proves its correctness using Wesolowski's [46] proof of exponentiation protocol.

Observe that we have replaced the single claim that we had about the tuple $(c, \boldsymbol{\zeta}, \gamma)$ with two separate claims $(c_0, \boldsymbol{\zeta}', \gamma_0)$ and $(c_1, \boldsymbol{\zeta}', \gamma_1)$, where $\boldsymbol{\zeta}' = (\zeta_2, \dots, \zeta_n)$, on $(n-1)$-variate polynomials so that if the original claim were true then the two resulting claims are true, whereas if the original claim is false then intuitively, at least one of the new claims is false.

Since we cannot afford to recurse on both claims, the next idea is to combine them into a single claim, using a random linear combination. In more detail, the verifier chooses a random coefficient[8] $\alpha \in \mathbb{F}$ and sends this coefficient to the prover. Consider now a new commitment

$$c' = c_0 \cdot (c_1)^{\alpha} = g^{\mathcal{Z}(P_0) + \alpha \cdot \mathcal{Z}(P_1)}. \tag{1}$$

[7] Computing this value directly by exponentiation takes time roughly $N = 2^n$ (using the standard repeated squaring trick) whereas we seek $\mathrm{poly}(n)$ time verification. Note that since the group's order is not known, one cannot first compute $q^{N/2}$ modulo the group order, and only then exponentiate.

[8] Looking ahead, it actually makes more sense to choose $\alpha$ from $\{0, \dots, 2^{\lambda} - 1\}$ where $\lambda$ is a statistical security parameter (independent of the field size). We ignore this here and simply follow the presentation in [21].

At first glance, $c'$ looks like a commitment to the (multilinear) polynomial $P'(\cdot) = P_0(\cdot) + \alpha \cdot P_1(\cdot)$. This is not exactly true since the operations in the exponent in Eq. (1) are over the integers rather than over the field $\mathbb{F}_p$. Nevertheless, it is indeed the case that when interacting with the honest prover, $c' = g^Z$, for an integer $Z$ that is *consistent* with $P'$. The verifier would therefore like to check that $c' = g^Z$ such that $Z$ is consistent with a polynomial $P'$ such that $P'(\boldsymbol{\zeta}') = \gamma'$, where $\gamma' \equiv \gamma_0 + \alpha \cdot \gamma_1 \pmod p$.

The parties have therefore reduced the instance $(c, \boldsymbol{\zeta}, \gamma)$ to $(c', \boldsymbol{\zeta}', \gamma')$, of smaller dimension (since the new instance corresponds to a polynomial on $n-1$ variables). At the bottom of the recursion (i.e., when the number of variables is 0), the parties are in the following situation - both hold a commitment $C_0 \in \mathbb{G}$ and a value $\gamma_0 \in \mathbb{F}_p$ and the claim is that $C_0 = g^{Z_0}$ such that $Z_0 = \gamma_0 \pmod p$. This can be checked by having the prover send $Z_0$ and the verifier explicitly checking that this value is consistent with $\gamma_0$ (and with $C_0$).

*Bounding the Blowup in Coefficients.* Note that as the protocol progresses, the magnitude of the digits in the base $q$ representation of the integers grows. However, this growth is bounded - in every iteration the main source of growth is multiplication by $\alpha$ and so the growth is bounded by roughly a factor of $p$ per iteration. Thus, by setting $q \gg p^n$ we ensure the growth of the coefficients does not break the homomorphism as the protocol progresses. This suffices for completeness. For soundness (or rather knowledge soundness), we actually need a larger bound on $q$ and have the verifier check in the base of the recursion that $Z_0 \leqslant p^n$. Loosely speaking, this is done so that a cheating prover cannot use integers with large digits to violate the homomorphism.

## 2.2 A Gap in the Proof

We need to show that the above scheme is an argument-of-knowledge.[9] Loosely speaking this means that for every polynomial-time prover strategy $\mathcal{P}$ there exists a polynomial-time extractor $\mathcal{E}$ so that for every input $(c, \boldsymbol{\zeta}, \gamma)$, if $\mathcal{P}$ convinces $V$ to accept with non-negligible probability, then $\mathcal{E}^{\mathcal{P}}$ outputs an integer $Z$ such that $g = c^Z$ and $Z$ is consistent with a polynomial $P$ such that $P(\boldsymbol{\zeta}) = \gamma$.

The extractor works recursively. Let us therefore assume that we have an extractor for the $(n-1)$-variate case and attempt to construct an extractor for the $n$-variate case. Thus, we are given a commitment $c$, a point $\boldsymbol{\zeta} \in \mathbb{F}^n$ a value $\gamma \in \mathbb{F}$ and a prover that convinces the verifier to accept with non-negligible probability. For sake of this overview however, let us pretend that the prover succeeds with probability close to 1.

The high-level idea for extraction is as follows. First, let the prover send its first message which is $(c_0, c_1, \gamma_0, \gamma_1)$. At this point our extractor continues the

[9] We note that [21] only aim to show that the protocol is an *argument* of knowledge (and this is inherent to their approach). Jumping ahead we mention that the evaluation proof in our variant of DARK will actually be a *proof* of knowledge (i.e., extraction is guaranteed even wrt computationally unbounded provers).

interaction with two uniformly random choices of $\alpha$ for the verifier, which we denote by $\hat{\alpha}$ and $\tilde{\alpha}$. This defines two claim triplets: $(\hat{c}, \boldsymbol{\zeta}', \hat{\gamma})$ and $(\tilde{c}, \boldsymbol{\zeta}', \tilde{\gamma})$, where:

$$\hat{c} = c_0 \cdot (c_1)^{\hat{\alpha}} \qquad\qquad \tilde{c} = c_0 \cdot (c_1)^{\tilde{\alpha}}$$
$$\hat{\gamma} \equiv \gamma_0 + \hat{\alpha} \cdot \gamma_1 \pmod p \qquad\qquad \tilde{\gamma} \equiv \gamma_0 + \tilde{\alpha} \cdot \gamma_1 \pmod p.$$

Since these two claims correspond to the $(n-1)$-variate case, we can now recursively run our extractor (twice) to obtain integers $\hat{Z}$ and $\tilde{Z}$ that are consistent with the respective claims. Namely, $\hat{Z}$ (resp., $\tilde{Z}$) is consistent with a polynomial $\hat{P}$ (resp., $\tilde{P}$) such that $\hat{P}(\boldsymbol{\zeta}') = \hat{\gamma}$ (resp., $\tilde{P}(\boldsymbol{\zeta}') = \tilde{\gamma}$), and $g^{\hat{Z}} = \hat{c}$ (resp., $g^{\tilde{Z}} = \tilde{c}$).

Consider the following linear-system, over the rationals, with unknowns $Z_0$ and $Z_1$.

$$\hat{Z} = Z_0 + \hat{\alpha} \cdot Z_1 \qquad\qquad \tilde{Z} = Z_0 + \tilde{\alpha} \cdot Z_1$$

Note that since $\hat{\alpha}$ and $\tilde{\alpha}$ are random, with overwhelming probability this system has a (unique) solution over the rationals:

$$Z_0 = \frac{\hat{\alpha} \cdot \tilde{Z} - \tilde{\alpha} \cdot \hat{Z}}{\hat{\alpha} - \tilde{\alpha}} \qquad\qquad Z_1 = \frac{\hat{Z} - \tilde{Z}}{\hat{\alpha} - \tilde{\alpha}} \tag{2}$$

An immediate difficulty that arises is that this solution may not be integral (i.e., $Z_0$ and $Z_1$ are not integers). However, Bünz et al. show that finding a fractional solution violates their hardness assumptions. Thus, (under the foregoing assumptions) we can safely assume that $Z_0$ and $Z_1$ are integers.

At this point we would like to combine $Z_0$ and $Z_1$ into $Z = Z_0 + q^{2^{n-1}} Z_1$, which serves as a valid output for the extractor. A question that arises however is whether $Z_0$ and $Z_1$ have bounded digits in their base $q$ representation. This is crucial since, as discussed above, if the digits are large the homomorphism breaks. Bünz et al. claim that it is indeed the case that $Z_0$ and $Z_1$ have small coefficients by observing that both the numerators and denominators in Eq. (2) consist of relatively small integers and so their quotient is small. While the claim that the quotient itself is small is indeed valid, it does not necessarily mean that *the base $q$ representation of the quotient has small digits*. Indeed, as demonstrated by the following example, this is not necessarily true and is the source of the gap in the DARK extraction procedure.

*Example 2.1.* Suppose that $q$ is odd and consider the integers $a = q + 1$ and $b = 2$ (in case $q$ is even a similar example with $a = q$ and $b = 2$ works). Note that the base $q$ representation of both only has small digits. However, $a/b$ has a digit of magnitude $(q + 1)/2$. Using such large digits breaks the homomorphism within a couple of steps.

We refer the reader to Lemma 8 in the full version of DARK [22] for the exact location of the gap in the proof. Specifically, in the third paragraph in that proof, it is claimed that $f_L(X)$ has small entries by the triangle inequality, but this does not account for the division by $\Delta_\alpha$ in the definition of $f_L$. This division can entirely break the claimed bounds on the base $q$ representation of $f_L$.

### 2.3 Resolving the Gap

Unfortunately, we do not know how to resolve the gap in the extraction procedure of [21]. Rather, we show how to modify the scheme and construct an extractor for our modified scheme.

As our first step, for a reason that will be made clear momentarily, rather than handling a single claim $(c, \boldsymbol{\zeta}, \gamma)$, we construct an interactive proof that handles a bundle of $\lambda$ claims $\{(c_i, \boldsymbol{\zeta}, \gamma_i)\}_{i\in[\lambda]}$, using the same evaluation point $\boldsymbol{\zeta} \in \mathbb{F}^n$, and where $\lambda$ is an auxiliary statistical security parameter. These $\lambda$ claims do not have to be distinct, so to solve the original problem $(c, \boldsymbol{\zeta}, \gamma)$ we can simply consider $\lambda$ copies of it. Thus, our goal is to construct a proof-of-knowledge of integers $Z_1, \dots, Z_\lambda$ that are consistent, respectively, with polynomials $P_1, \dots, P_\lambda$ so that $P_i(\boldsymbol{\zeta}) = \gamma_i$, for every $i \in [\lambda]$.

We follow the divide and conquer approach of [21]. Namely, using a similar type of interaction we split each one of the $\lambda$ claims $(c_i, \boldsymbol{\zeta}, \gamma_i)$ into two claims each on an $(n-1)$-variate polynomial. At this point, we have, overall, $2\lambda$ claims on $(n-1)$-variate polynomials and we would like to reduce these to just $\lambda$ claims so that we can recurse. Denote this set of claims by $\{(c'_i, \boldsymbol{\zeta}', \gamma'_i)\}_{i\in[2\lambda]}$ (note that the indexing intentionally ignores the source for each one of these claims).

Let us first describe how we generate a single claim from these $2\lambda$ claims. The verifier chooses a random subset $S \subseteq [2\lambda]$ and sends $S$ to the prover. Consider now the new claim $(\bar{c}, \boldsymbol{\zeta}', \bar{\gamma})$, where $\bar{c} = \prod_{i\in S} c'_i$ and $\bar{\gamma} \equiv \sum_{i\in S} \gamma'_i \pmod{p}$. If the original claims were true then with probability 1 the new claim is true, whereas, *intuitively*, if at least one of the original claims was false then with probability 1/2 the new claim is false[10]. We therefore repeat this process $\lambda$ times to derive $\lambda$ claims so that if one of the original claims was false, then, with all but $2^{-\lambda}$ probability, one of the new claims will be false.

To actually make this argument work, we need to construct an extractor. As suggested above, the extractor can rewind the computation a constant $r$ number of times to deduce a linear-system, analogous to that of Sect. 2.2, but now with $r \cdot \lambda$ equations and $\lambda$ variables, where the coefficients are uniformly random 0/1 values.

Similarly to the situation in Sect. 2.2, it is clear that this linear-system is full rank (over the rationals) but it is not a priori clear that the solution is integral, nor that its base $q$ representation has small digits. Nevertheless, we show that for *random* Boolean matrices this is indeed the case. This fact, which turns out to be non-trivial to prove, is summarized in the following lemma:

**Lemma 2.2 (Informally Stated, see [12]).** *If* $\mathbf{A}$ *is uniformly random in* $\{0,1\}^{\lambda\times r\cdot\lambda}$ *for* $r \geq 5$*, then with all but* $2^{-\Omega(\lambda)}$ *probability* $\mathbf{A}$ *has a right-inverse* $\mathbf{B} \in \mathbb{Z}^{r\cdot\lambda\times\lambda}$*. Moreover, the inverse matrix* $\mathbf{B}$ *can be found in* $\mathrm{poly}(\lambda)$ *time and its entries have bit length at most* $\mathrm{poly}(\lambda)$*.*

[10] This is not actually precise since there are many polynomials that are consistent with the $c'_i$'s and so the claim could be true wrt some of these polynomials. This is dealt with formally by showing *knowledge soundness* (i.e., constructing an extractor).

Note that the fact that the inverse matrix $\mathbf{B}$ of our linear-system has relatively small integral coefficients (independent of $q$) is crucial since it means that our solution is integral and has small digits in base $q$.

Our proof of Lemma 2.2 leverages ideas from the theory of integer lattices, see the full version for details [12]. Having found the desired solution to the specified linear-system, our extractor can proceed in the extraction similarly to the extraction in DARK. This concludes the high-level description of our resolution of the gap in the DARK scheme.

*Remark 2.3.* Note that our approach not only resolved the gap in the DARK extraction, but also unconditionally avoided the possibility of the linear-system having a non-integral solution. This simultaneously simplifies the definitions and proofs and lets us avoid an undesirable reliance on additional, poorly understood, cryptographic assumptions.

*Remark 2.4.* For sake of convenience, we used $\lambda$ repetitions in our analysis. We remark however that the number of repetitions here is a *statistical security parameter*. Namely, it bounds even a computationaly *unbounded* adversary's success probability by $2^{-\Omega(\lambda)}$. Thus, in practice it may be best to differentiate between this parameter and the cryptographic parameter that corresponds to the size of the group.

## 2.4 Small Space Implementation

Having resolved the gap in the security proof, we now turn out attention to implementing the polynomial commitment in small space.

When considering sublinear space algorithms it is important to specify how the input is given (since the algorithm cannot simply copy the input to its work tape, see [30] for a comprehensive discussion). For our context, the most natural choice is for our small space algorithms to be given *multi-pass streaming access* to the description of the multilinear polynomial. That is, the evaluations of the polynomial on the Boolean hypercube are written on the read-only input tape. The algorithm can process the input tape from left-to-right, or choose to reset the machine head to the beginning of the tape. The reason that this choice is natural is that when constructing our argument-system, we will need to apply this commitment to a transcript of a computation. Such a transcript can be generated in a (resettable) streaming manner by simply executing the computation.

With that in mind, let us first consider our commitment algorithm. Recall that we are given as input the stream of values $\{P(\mathbf{b})\}_{\mathbf{b}\in\{0,1\}^n}$, where $P : \mathbb{F}^n \to \mathbb{F}$ is a multilinear polynomial and we need to produce the commitment $g^{\mathcal{Z}(P)} = g^V$, where

$$V = \sum_{\mathbf{b}\in\{0,1\}^n} q^{\mathbf{b}} \cdot P(\mathbf{b}).$$

Note that we cannot compute $V$ directly and then exponentiate. This is because even storing $V$ requires $2^n$ bits. Rather, we will leverage the fact that $V$ appears only in the exponent and compute $g^V$ directly.

We do so by iterating through $\mathbf{b}$ in lexicographic order while maintaining, as we go along, two variables $C$ and $D$. We will maintain the invariant that at the start of the $\mathbf{b}$-th iteration $D = g^{q^{\mathbf{b}}}$ and $C = g^{V_{<\mathbf{b}}}$, where $V_{<\mathbf{b}} = \sum_{\mathbf{b}'<\mathbf{b}} q^{\mathbf{b}'} \cdot P(\mathbf{b}')$. To do so we:

- Initialize $C$ as the group's identity element and $D = g$.
- To update $C$ and $D$ from $\mathbf{b}$ to $\mathbf{b}+1$, we set $C \leftarrow C \cdot D^{P(\mathbf{b})}$ and $D \leftarrow D^q$ (using repeated squaring).

It is not hard to see that the invariant is indeed maintained. Given the value of $D$ in the last iteration, it is easy to generate the commitment $g^V$.

Implementing the evaluation proofs is more subtle. The key challenge here is that throughout the recursion, the prover needs to deal with the intermediate polynomials that are defined throughout the recursion, but only has streaming access to the *original* base polynomial. Needless to say, we cannot afford to explicitly store the intermediate polynomials since this would introduce $2^{\Omega(n)}$ space usage.

Thus, we need, for sake of space efficiency, to open up the recursion and work directly with our original stream $P$. At first glance, one would hope that using the polynomial $P$ we can emulate streaming access to the intermediate polynomials that we encounter. Unfortunately, we do not know how to do that. Rather, in order to commit or evaluate some intermediate polynomial $Q$, we show that as we process the base polynomial $P$, each value that we encounter, has some partial contribution to $Q$ (with coefficients that depend on the verifier's random challenges). The crucial observation is that both the commitment to $Q$ and evaluation are *linear* and therefore commute. This means that we do not have to process the partial contributions of each entry of $Q$ in sequence. Furthermore, we show that the coefficients of these entries in the linear combination can either be produced individually in small space (when evaluating the intermediate polynomials) or generated as a stream in small space (when producing commitments).

### 2.5 Generalizing Pietrzak's Proof of Exponentiation Protocol

Our Proof of Exponentiaion (PoE) protocol builds on Pietrzak's PoE protocol [37]. We therefore start by recalling his protocol and then proceed to describe our improvement.

*Pietrzak's* PoE *protocol.* Let $\mathbb{G}$ be a group and $q \in \mathbb{Z}$. Recall that the prover wishes to prove that $y = x^{q^{2^t}}$ for some $x, y \in \mathbb{G}$ and $t \in \mathbb{N}$. As a shorthand, we will use $T = 2^t$ and denote the claim $y = x^{q^T}$ by the tuple $(x, y, T)$ and refer to this as a claim of size $T$ (because its validity can be easily checked by performing $T$ repeated squarings). To proceed with the proof, the prover first sends a single group element $\mu = x^{q^{T/2}}$, which implicitly defines two sub-claims $(x, \mu, T/2)$ and $(\mu, y, T/2)$ of size $T/2$ each. Note that if $(x, y, T)$ is true then both claims $(x, \mu, T/2)$ and $(\mu, y, T/2)$ must also be true: $y = x^{q^T}$ and $\mu = x^{q^{T/2}}$

implies that $y = \mu^{q^{T/2}}$. However, intuitively, if $(x, y, T)$ is false then for any (even maliciously generated) $\mu$, at least one of the sub-claims must be false. Instead of recursing on both subclaims, the prover combines the two subclaims into a single claim of size $T/2$. The new claim $(x', y', T/2)$ is computed by taking a, verifier specified, random linear combination of the two claims. That is,

$$x' = x^r \cdot \mu \quad \text{and} \quad y' = \mu^r \cdot y,$$

where $r \leftarrow \mathbb{Z}_{2^\lambda}$ is sampled by the verifier. It is easy to see that if $(x, \mu, T/2)$ and $(\mu, y, T/2)$ are true then $(x', y', T/2)$ is also true, and Pietrzak (relying on $QR_N^+$ not having small order subgroups) shows that if one of $(x, \mu, T/2)$ or $(\mu, y, T/2)$ is false, then with overwhelming probability, over the choice of $r$, the claim $(x', y', T/2)$ is false. Now, the prover and verifier recurse on the $T/2$-sized claim, halving the size every time and eventually ending up in the base case $(T = 1)$ where the verifier just needs to check whether $y = x^q$ (which can be done in poly$(\lambda)$ time).

*Our New* PoE *protocol.* As mentioned above the main downside of Pietrzak's protocol is that statistical soundness can only be proved for groups where small order subgroups do not exist. This difficulty arises since we are taking random linear combinations of *group* elements rather than *field* elements. In particular, if one of the group elements has small order, then the random linear combination does not have the desired effect.

Our approach for resolving this difficulty is inspired by our resolution for the gap in the DARK scheme (see Sects. 2.2 and 2.3). We will maintain more instances throughout the interaction and take random subset sums of all of these instances rather than random linear combinations of two instances. Interestingly, this simple idea gets us quite a bit of mileage - simplifying the analysis, improving the assumptions (e.g., in the case of class groups) and generalizing the result to general groups.

In more detail, rather than handling a single claim $(x, y, T)$, we will show a protocol for checking $\lambda$ claims $\{(x_i, y_i, T)\}_{[i \in [\lambda]}$ all sharing the same exponent parameter $T$. Note that the single claim case can be easily reduced to this more general setting by simply setting $x_1 = \cdots = x_\lambda = x$ and $y_1 = \cdots = y_\lambda = y$.

Analogous to Pietrzak's protocol, our prover sends the sequence of values $\mu = (\mu_1, \ldots, \mu_\lambda) \in \mathbb{G}^\lambda$ as its first message, where $\mu_i = x_i^{q^{T/2}}$, for every $i \in [\lambda]$. This decomposes the $\lambda$ claims $\{(x_i, y_i, T)\}_{i \in [\lambda]}$ into two sets of $\lambda$ claims each $\{(x_i, \mu_i, T/2)\}_{i \in [\lambda]}$ and $\{(\mu_i, y_i, T/2)\}_{i \in [\lambda]}$. It will be convenient however to think of these as a single set of claims $\{(z_i, w_i, T/2)\}_{i \in [2\lambda]}$. Note that if the original set of claims was not true, then no matter what values $\{\mu_i\}_{i \in [\lambda]}$ the prover sends, at least one of the claims in $\{(z_i, w_i, T/2)\}_{i \in [2\lambda]}$ must be false.

*Reducing Claims via Subset Products.* We show a simple and general method for reducing the number of claims. Let us first see how to produce a single new claim from these $2\lambda$ claims. The verifier chooses at random a set $S \subseteq [2\lambda]$ and sends $S$ to the prover. Consider the claim $(z', w', T/2)$ where:

$$z' = \prod_{i \in S} z_i \quad \text{and} \quad w' = \prod_{i \in S} w_i.$$

Observe that if all the original claims were true (i.e., $w_i = z_i^{q^{T/2}}$, for every $i \in [\lambda]$) then $(z')^{q^{T/2}} = \prod_{i \in S} z_i^{q^{T/2}} = \prod_{i \in S} w_i = w'$ and so the resulting claim holds. On the other hand, if even just one of the original claims is false then:

$$\Pr_S \left[ (z')^{q^{T/2}} = w' \right] = \Pr_S \left[ \prod_{i \in S} z_i^{q^{T/2}} = \prod_{i \in S} w_i \right] = \Pr_S \left[ \prod_{i \in S} u_i = \mathbf{1}_{\mathbb{G}} \right] \leqslant 1/2,$$

where $u_i = z_i^{q^{T/2}} \cdot w_i^{-1}$ for every $i \in [2\lambda]$, the group's identity element is denoted by $\mathbf{1}$ and the inequality follows from the simple principle that a random subset product of a sequence of group elements, not all of which are equal to $\mathbf{1}$, is equal to $\mathbf{1}$ with probability at most $1/2$.

To get $2^{-\lambda}$ error probability, we simply repeat this process $\lambda$ times to get a new sequence of $\lambda$ claims, each of size $T/2$. We have thus reduced our $\lambda$ size $T$ claims to $\lambda$ size $T/2$ claims. We can continue recursing as in Pietrzak's protocol until $T = 1$ in which case the verifier can solve the problem by itself.

*Remark 2.5.* We remark that our technique introduces a factor of $\lambda$ overhead in the communication complexity as compared to Pietrzak's protocol. This is due to the fact that the prover has to send $\lambda$ group elements per round.

Similarly to Remark 2.4, $\lambda$ is a *statistical* (rather than computational) security parameter and relatively small values of $\lambda$ may suffice. Moreover, we believe that it is possible to "interpolate" between our approach and that of [37] by considering the minimal sub-group size of $\mathbb{G}$ and using coefficients of suitably larger magnitude in our choice of the random matrix.

## 3 Preliminaries

We let "$\circ$" denote the string concatenation operator and let $\epsilon$ denote the empty string; that is, for any string $s$ it holds that $s = s \circ \epsilon = \epsilon \circ s$.

Let $S$ be a finite, non-empty set. We let $x \leftarrow S$ denote sampling an element $x$ uniformly at random from $S$. For any $N \in \mathbb{N}$, we let $S^N$ denote the set of all sequences of length $N$ containing elements of $S$, and note that $S^0 := \{\epsilon\}$. As usual, we make the convention that if $j > k$ then $\sum_{i=j}^{k} a_i = 0$ and $\prod_{i=j}^{k} a_i = 1$.

We let $\mathbb{F}_p$ denote a finite field of prime cardinality $p$, and often use lower-case Greek letters to denote elements of $\mathbb{F}$, e.g., $\alpha \in \mathbb{F}$. We use boldface lowercase letters to denote binary vectors, e.g. $\mathbf{b} \in \{0,1\}^n$. For bit strings $\mathbf{b} \in \{0,1\}^n$, we naturally associate $\mathbf{b}$ with integers in the set $\{0, 1, \ldots, 2^n - 1\}$; i.e., $\mathbf{b} \equiv \sum_{i=1}^{n} b_i \cdot 2^{i-1}$. We assume that $\mathbf{b} = (b_n, \ldots, b_1)$, where $b_n$ is the most significant bit and $b_1$ is the least significant bit. For bit string $\mathbf{b} \in \{0,1\}^n$ and $\sigma \in \{0,1\}$ we let $\sigma\mathbf{b}$ (resp., $\mathbf{b}\sigma$) denote the string $(\sigma \circ \mathbf{b}) \in \{0,1\}^{n+1}$ (resp., $(\mathbf{b} \circ \sigma) \in \{0,1\}^{n+1}$). We use boldface lowercase to denote $\mathbb{F}$ vectors, e.g., $\boldsymbol{\alpha} \in \mathbb{F}^n$. For $(\alpha_n, \ldots, \alpha_1) =$

$\boldsymbol{\alpha} \in \mathbb{F}^n$, we refer to $\alpha_n$ as the most significant field element and $\alpha_1$ as the least significant field element. For two equal length vectors $\mathbf{u}, \mathbf{v}$, we let $\mathbf{u} \odot \mathbf{v}$ denote the coordinate-wise product of $\mathbf{u}$ and $\mathbf{v}$. We let uppercase calligraphic letters denote sequences and let corresponding lowercase letters to denote its elements, e.g., $\mathcal{Y} = (y_{\mathbf{b}})_{\mathbf{b} \in \{0,1\}^n} \in \mathbb{F}^N$ is a sequence of $N$ elements in $\mathbb{F}$. Often, for $\mathbf{b} \in \{0,1\}^n$, we let $\mathcal{Y}_{\mathbf{b}}$ denote the value $y_{\mathbf{b}}$.

We use upper case letters to denote matrices, e.g., $M \in \mathbb{Z}^{m \times n}$. For a matrix $M$ of dimension $m \times n$, we let $M(i,*)$ and $M(*,j)$ denote the $i^{th}$ row and $j^{th}$ column of $M$, respectively. For row vector $\mathbf{u}$ of length $m$ and column vector $\mathbf{v}$ of length $n$, we let $\mathbf{u} \cdot M$ and $M \cdot \mathbf{v}$ denote the standard matrix-vector product.

*Non-standard Notation.* We are also interested in matrix-vector "exponents". Let $\mathbb{G}$ be some group, $M \in \mathbb{Z}^{m \times n}$, $\mathbf{u} = (u_1, \ldots, u_m) \in \mathbb{G}^{1 \times m}$, and $\mathbf{v} = (v_1, \ldots, v_n)^\top \in \mathbb{G}^{n \times 1}$. We let $\mathbf{u} \star M$ and $M \star \mathbf{v}$ denote a matrix-vector exponent, defined as

$$(\mathbf{u} \star M)_j = \prod_{i=1}^{m} u_i^{M(i,j)} \qquad (M \star \mathbf{v})_{i'} = \prod_{j'=1}^{n} v_{j'}^{M(i',j')},$$

for every $j \in [n]$ and every $i' \in [m]$. Note that $\mathbf{u} \star M \in \mathbb{G}^{1 \times n}$ and $M \star \mathbf{v} \in \mathbb{G}^{m \times 1}$.

For vector $\mathbf{x} \in \mathbb{Z}^n$ and group element $g \in \mathbb{G}$, we abuse notation and define $g^{\mathbf{x}} := (g^{x_1}, \ldots, g^{x_n})$. Finally, for $k \in \mathbb{Z}$ and vector $\mathbf{u} \in \mathbb{G}^n$, we let $\mathbf{u}^k$ denote the vector $(u_1^k, \ldots, u_n^k) \in \mathbb{G}^n$.

### 3.1 Multilinear Polynomials

An $n$-variate polynomial $f \colon \mathbb{F}^n \to \mathbb{F}$ is multilinear if the individual degree of each variable in $f$ is at most 1.

**Fact 3.1.** *An multilinear polynomial $f : \mathbb{F}^n \to \mathbb{F}$ (over a finite field $\mathbb{F}$) is uniquely defined by its evaluations over the Boolean hypercube. Moreover, for every $\boldsymbol{\zeta} \in \mathbb{F}^n$ it holds that $f(\boldsymbol{\zeta}) = \sum_{\mathbf{b} \in \{0,1\}^n} f(\mathbf{b}) \cdot \prod_{i=1}^{n} \chi(b_i, \zeta_i)$, where $\chi(b, \zeta) = b \cdot \zeta + (1-b) \cdot (1-\zeta)$.*

As a short hand, we will often denote $\prod_{i=1}^{n} \chi(b_i, \zeta_i)$ by $\overline{\chi}(\mathbf{b}, \boldsymbol{\zeta})$ for $n = |\mathbf{b}| = |\boldsymbol{\zeta}|$.

*Notation for Multilinear Polynomials.* Throughout this work, we represent $n$-variate, multilinear polynomials $f$ by the $N$-sized sequence $\mathcal{Y}$ containing evaluations of $f$ over the Boolean hypercube. That is, $\mathcal{Y} := (f(\mathbf{b}))_{\mathbf{b} \in \{0,1\}^n}$, and denote the evaluation of the multilinear polynomial defined by $\mathcal{Y}$ on $\boldsymbol{\zeta}$ as $\mathsf{ML}(\mathcal{Y}, \boldsymbol{\zeta}) := \sum_{\mathbf{b}} \mathcal{Y}_{\mathbf{b}} \cdot \overline{\chi}(\mathbf{b}, \boldsymbol{\zeta})$. Furthermore, we also consider the evaluation of a multilinear polynomial defined by some integer sequence $\mathcal{Z} \in \mathbb{Z}^N$. For any $\boldsymbol{\zeta} \in \mathbb{F}_p$ for prime $p$, we define $\mathsf{ML}(\mathcal{Z}, \boldsymbol{\zeta}) := \sum_{\mathbf{b}} (\mathcal{Z}_{\mathbf{b}} \bmod p) \cdot \overline{\chi}(\mathbf{b}, \boldsymbol{\zeta})$.

### 3.2 Groups of Hidden Order

We start by defining the notion of a *group sampler*.

**Definition 3.2 (Group Sampler).** *A PPT algorithm $\mathcal{G}$ is a* group sampler *if for every $\lambda \in \mathbb{N}$, $\mathcal{G}$ on input $1^\lambda$ samples a description*[11] *$\mathbb{G}$ of a group of size at most $2^\lambda$. As a shorthand, we denote this random process by $\mathbb{G} \leftarrow \mathcal{G}(1^\lambda)$, and by $g \leftarrow \mathbb{G}$ denote the process of sampling a random group element from $\mathbb{G}$ and assigning it to $g$.*

*Furthermore, we say that $\mathcal{G}$ is* public-coin *if the output of $\mathcal{G}$ (i.e., the group description) is a uniformly random string.*

Here, we will focus only on group samplers $\mathcal{G}$ for which the Hidden Order Assumption holds, which, informally, requires that it be computationally hard to find (a multiple of) the order of a random group element of $\mathbb{G} \leftarrow \mathcal{G}(1^\lambda)$.

**Assumption 3.3 (Hidden Order Assumption).** *The* Hidden Order Assumption *holds for $\mathcal{G}$ if for every polynomial-size family of circuits $A = \{A_\lambda\}_{\lambda \in \mathbb{N}}$:*

$$\Pr\left[g^a = 1 \wedge a \neq 0 : \mathbb{G} \leftarrow \mathcal{G}(1^\lambda), g \leftarrow \mathbb{G}, a \leftarrow A_\lambda(\mathbb{G}, g)\right] \leq \mathsf{negl}(\lambda). \quad (3)$$

*Candidates for $\mathcal{G}$.* In this work we consider two main candidates for $\mathcal{G}$ where the Hidden Order Assumption is believed to hold:

1. RSA group: the multiplicative group $\mathbb{Z}_N^*$ of integers modulo a product $N = P \cdot Q$ for large random primes $P$ and $Q$. Here, the Hidden Order Assumption holds assuming the hardness of factoring $N$ when it is a product of safe primes. This group can be sampled by choosing random primes and specifying their product. However, this is private-coin type generation and it is not clear how to generate the group in a public way (this corresponds to the well-studied problem of generating hard factoring instances using only public-coins).
2. Class group: the class group of an of imaginary quadratic order. Here, the Hidden Order Assumption (in fact, even much stronger assumptions) are believed to hold (see, e.g., [21,46]). The main feature of such class groups is that there is a way to sample the group description using only public-coins. These are, to the best of our knowledge, the only known public-coin hidden order groups. We refer the reader to [21] for details.

### 3.3 Interactive Games and Proof Systems

**Definition 3.4 (Merlin-Arthur Games).** *Let $r$ be a positive integer.*

*An* $\mathsf{MA}[2r]$ game *(or just an* MA *game*[12] *if $r$ is unspecified) is a tuple $\mathcal{G} = (1^r, 1^\ell, W)$, where $\ell \in \mathbb{Z}^+$ and $W \subseteq \{0,1\}^*$ is a set, called the* win predicate*, that*

[11] The group description includes a poly($\lambda$) description of the identity element, and poly($\lambda$) size circuits checking membership in the group, equality, performing the group operation and generating a random element in the group.

[12] MA stands for Merlin-Arthur proofs [3] (differing from Arthur-Merlin proofs in that the prover (Merlin) sends the first message).

*is represented as a boolean circuit. The integer $r$ is called the* number of rounds *of $\mathcal{G}$ and $\{0,1\}^\ell$ is called the* challenge space.

*If $\mathcal{G} = (1^r, 1^\ell, W)$ is an* $\mathsf{MA}[2r]$ *game and $P : \{0,1\}^* \to \{0,1\}^*$ is a function, then the* value of $\mathcal{G}$ with respect to $P$ *is denoted and defined as*

$$v[P](\mathcal{G}) \overset{\mathsf{def}}{=} \Pr_{\beta_1,\ldots,\beta_r \leftarrow \{0,1\}^\ell} \left[ (\alpha_1, \beta_1, \ldots, \alpha_r, \beta_r) \in W \right],$$

*where each $\alpha_i$ denotes $P(\beta_1, \ldots, \beta_{i-1})$. The* value *of $\mathcal{G}$, denoted $v(\mathcal{G})$, is* $\sup_P \{v[P](\mathcal{G})\}$.

**Definition 3.5 (Game Transcripts).** *If $\mathcal{G} = \left(1^r, 1^\ell, W\right)$ is an* $\mathsf{MA}[2r]$ *game, then a* transcript for $\mathcal{G}$ *is a tuple $\tau = (\alpha_1, \beta_1, \ldots, \alpha_r, \beta_r)$ with each $\beta_i \in \{0,1\}^\ell$ and $\alpha_i \in \{0,1\}^*$. If $\tau$ is contained in $W$, then it is said to be an* accepting transcript *for $\mathcal{G}$. If for a function $P : \{0,1\}^* \to \{0,1\}^*$, $\alpha_i = P(\beta_1, \ldots, \beta_{i-1})$ for each $i \in [r]$, then $\tau$ is said to be* consistent *with $P$. If $\tau$ is both an accepting transcript for $\mathcal{G}$ and consistent with $P$, we say simply that $\tau$ is an* accepting transcript for $(P, \mathcal{G})$.

**Definition 3.6 (MA Verifiers).** *For a function $r : \mathbb{Z}^+ \to \mathbb{Z}^+$ and a language $\mathcal{L}$, an* $\mathsf{MA}[2r]$ verifier for $\mathcal{L}$ *is a polynomial-time algorithm $V$, where:*

- *$V$ maps any string $x \in \{0,1\}^*$ to an* $\mathsf{MA}\big[2r(|x|)\big]$ *game.*[13]
- *The* completeness *of $V$ is a function $c : \mathbb{Z}^+ \to [0,1]$, defined as*

$$c(n) \overset{\mathsf{def}}{=} \min_{x \in \mathcal{L} \cap \{0,1\}^n} v\big(V(x)\big).$$

- *The* soundness error *of $V$ is a function $s : \mathbb{Z}^+ \to [0,1]$, defined as*

$$s(n) \overset{\mathsf{def}}{=} \max_{x \in \{0,1\}^n \setminus \mathcal{L}} v\big(V(x)\big).$$

**Definition 3.7 (Witness-Extended Emulation (cf. [35])).** *An* MA *verifier $V$ has (statistical)* witness-extended $\epsilon(\cdot)$-emulation *with respect to a relation $\mathcal{R}$ if there exists an expected polynomial-time oracle algorithm $\mathcal{E}$ such that for all $P : \{0,1\}^* \to \{0,1\}^*$ and all $x \in \{0,1\}^*$, if we sample $(\tau, w) \leftarrow \mathcal{E}^P(x)$, then:*

- *$\tau$ is distributed uniformly at random on the set of all possible transcripts between $V(x)$ and $P$.*
- *With all but $\epsilon(|x|)$ probability, if $\tau$ is an accepting transcript for $V(x)$ then $(x, w) \in \mathcal{R}$.*

**Definition 3.8.** *An* MA *verifier $V$ has* statistical witness-extended emulation *with respect to a relation $\mathcal{R}$ if it has statistical witness-extended $\epsilon$-emulation (as per Definition 3.7) for some negligible function $\epsilon$.*

[13] In particular, this definition implies there is a polynomial in $n$ that bounds the length of any accepting transcript for $V(x)$ when $x \in \{0,1\}^n$.

### 3.4 Multilinear Polynomial Commitment

Polynomial commitment schemes, introduced by Kate et al. [32] and generalized in [19,21,41,45], are a cryptographic primitive that allows one to commit to a polynomial of bounded degree and later provably reveal evaluations of the committed polynomial. Since we consider only multilinear polynomials, we tailor our definition to them.

*Convention.* In defining the syntax of various protocols, we use the following convention for any list of arguments or returned tuple $(a, b, c; d, e)$ – variables listed before semicolon are known both to the prover and verifier whereas the ones after are only known to the prover. In this case, $a, b, c$ are public whereas $d, e$ are secret. In the absence of secret information the semicolon is omitted.

**Definition 3.9 (Multilinear Polynomial Commitment Scheme).** *A* multilinear polynomial commitment scheme *is a tuple of protocols* $(\mathsf{Setup}, \mathsf{Com}, \mathsf{isValid}, \mathsf{Eval})$ *such that*

1. $pp \leftarrow \mathsf{Setup}(1^\lambda, p, 1^n)$*: takes as input the security parameter* $\lambda \in \mathbb{N}$ *and outputs public parameter* $pp$ *that allows to support* $n$*-variate multilinear polynomials over* $\mathbb{F} = \mathbb{F}_p$ *for some prime* $p$*.*
2. $(C; d) \leftarrow \mathsf{Com}(pp, \mathcal{Y})$*: takes as input public parameters* $pp$ *and a description of a multilinear polynomial* $\mathcal{Y} = (y_{\mathbf{b}})_{\mathbf{b} \in \{0,1\}^n}$ *and outputs a commitment* $C$ *and a (secret) decommitment* $d$*.*
3. $b \leftarrow \mathsf{isValid}(pp, C, \mathcal{Y}, d)$*: takes as input* $pp$*, a commitment* $C$*, a description of the multilinear polynomial* $\mathcal{Y}$ *and a decommitment* $d$*, and returns a decision bit* $b \in \{0, 1\}$*.*
4. $\mathsf{Eval}(pp, C, \boldsymbol{\zeta}, \gamma; \mathcal{Y}, d)$*: is a public-coin interactive proof system* $(P, V)$ *for the relation:*

$$\mathcal{R}_{\mathsf{ml}} = \Big\{(pp, C, \boldsymbol{\zeta}, \gamma; \mathcal{Y}, d) : \mathsf{isValid}(pp, C, \mathcal{Y}, d) = 1 \wedge \gamma = \mathsf{ML}(\mathcal{Y}, \boldsymbol{\zeta})\Big\}, \qquad (4)$$

*where* $V$ *is an* MA *verifier (as per Definition 3.6) where* $P$ *is the honest strategy for* $V$*. Note that the verifier in this proof-system gets as input the public parameters* $pp$*, commitment* $C$*, evaluation point* $\boldsymbol{\zeta} \in \mathbb{F}^n$ *and claimed evaluation* $\gamma \in \mathbb{F}$*, and the prover additionally receives the full description of the polynomial* $\mathcal{Y}$ *and the decommitment* $d$*.*

*We require the following three properties from the scheme* $(\mathsf{Setup}, \mathsf{Com}, \mathsf{isValid}, \mathsf{Eval})$*:*

1. ***Perfect Correctness:*** *for all primes* $p$*,* $\lambda \in \mathbb{N}$*,* $n \in \mathbb{N}$ *and all* $\mathcal{Y} \in \mathbb{F}_p^{2^n}$ *and* $\boldsymbol{\zeta} \in \mathbb{F}_p^n$*,*

$$\Pr\left[1 = \mathsf{Eval}(pp, C, \mathcal{Z}, \gamma; \mathcal{Y}, d) : \begin{array}{c} pp \leftarrow \mathsf{Setup}(1^\lambda, p, 1^n), \\ (C; d) \leftarrow \mathsf{Com}(pp, \mathcal{Y}),\ \gamma = \mathsf{ML}(\mathcal{Y}, \boldsymbol{\zeta}) \end{array}\right] = 1\ .$$

2. ***Computational Binding:*** *for every polynomial-sized family of circuits* $A = \{A_\lambda\}_{\lambda \in \mathbb{N}}$ *the following holds*

$$\Pr\left[ (b_0 = b_1 = 1) \wedge (\mathcal{Y}_0 \neq \mathcal{Y}_1) : \begin{array}{c} pp \leftarrow \mathsf{Setup}(1^\lambda, p, 1^N) \\ (C, \mathcal{Y}_0, \mathcal{Y}_1, d_0, d_1) \leftarrow A_\lambda(pp) \\ b_0 \leftarrow \mathsf{isValid}(pp, C, \mathcal{Y}_0, d_0) \\ b_1 \leftarrow \mathsf{isValid}(pp, C, \mathcal{Y}_1, d_1) \end{array} \right] \leq \mathsf{negl}(\lambda) \ .$$

3. ***Witness-Extended Emulation:*** *For* $\mathsf{Eval} = (P, V)$*,* $V$ *has (statistical) witness-extended emulation for the relation* $\mathcal{R}_{\mathsf{ml}}$ *(defined in Eq. (4)).*

*Remark 3.10.* We note that our definition of polynomial commitment scheme is stronger than the ones used in the literature (see, e.g., [5,7,11,21,33,34,42,45,47,48]), in that we require $\mathsf{Eval}$ to have *statistical* soundness (rather than computational). As a result we show soundness for every pair $(x, pp)$.

A key ingredient in our efficient argument-systems is polynomial commitments that can be generated in a time *and space* efficient way. We call such polynomial commitments *streamable.*

**Definition 3.11 (Streamable Multilinear Polynomial Commitment Scheme).** *A* streamable multilinear polynomial commitment scheme *is a multilinear polynomial commitment scheme (as per Definition 3.9) with the following efficiency properties for n-variate multilinear polynomials over* $\mathbb{F}_p$ *for some prime* $p \leq 2^\lambda$*:*

1. *The commitment output by* $\mathsf{Com}$ *is of size* $n \cdot \mathrm{poly}(\lambda)$*, and assuming multi-pass streaming access to the description of the polynomial, the commitment can be implemented in time* $2^n \cdot \mathrm{poly}(n, \lambda)$ *and space* $\mathrm{poly}(n, \lambda)$*.*
2. *The communication complexity of the* $\mathsf{Eval}$ *protocol is* $n \cdot \mathrm{poly}(\lambda)$ *and the receiver of* $\mathsf{Eval}$ *runs in time* $\mathrm{poly}(n, \lambda)$*. Assuming multi-pass streaming access to the description of the polynomial, the committer of* $\mathsf{Eval}$ *can be implemented in time* $2^n \cdot \mathrm{poly}(n, \lambda)$ *and space* $\mathrm{poly}(n, \lambda)$*.*

## 4 Our Results

In this section we formally state our main results. All analyses, protocols, and proofs are deferred to the full version [12] (unless otherwise stated).

*Time- and Space-efficient Arguments.* Our first main result is a time- and space-efficient public-coin zero-knowledge argument-system.

**Theorem 4.1.** *Assume the existence of a group sampler for which the hidden order assumptions holds (see Assumption 3.3). Then, there exists a public-coin zero-knowledge argument-system for any* **NP** *relation verifiable by a time* $T$ *space* $S$ *random access machine with the following complexity.*

1. *The protocol has perfect completeness and neligible soundness error.*
2. *The number of rounds is* $O(\log T)$.
3. *The communication complexity is* $\mathrm{poly}(\lambda, \log T)$.
4. *The prover runs in time* $T \cdot \mathrm{poly}(\lambda, \log T)$ *and space* $S \cdot \mathrm{poly}(\lambda, \log T)$.
5. *The verifier runs in time* $|x| \cdot \mathrm{poly}(\lambda, \log T)$, *for a given input* $|x|$.

Theorem 4.1 relies on a new polynomial commitment scheme discussed next.

*Streamable Polynomial Commitments.* The core component of our time- and space-efficient arguments is a new polynomial commitment scheme for multilinear polynomials where the committer can be implemented in small space and verification is only poly-logarithmic.

**Theorem 4.2.** *Assume the existence of a group sampler for which the hidden order assumptions holds. Then, there exists a streamable multilinear polynomial commitment scheme* $(\mathsf{Setup}, \mathsf{Com}, \mathsf{isValid}, \mathsf{Eval})$ *(as per Definition 3.11) over finite field* $\mathbb{F}$ *of prime-order* $p$ *with the following efficiency guarantees:*

1. $\mathsf{Com}$ *outputs a commitment of size* $\mathrm{poly}(\lambda)$ *bits, runs in time* $2^n \cdot \mathrm{poly}(n, \lambda, \log(p))$ *and space* $n + O(\log(p)) + \mathrm{poly}(\lambda)$, *and uses a single pass over the stream;*
2. $\mathsf{Eval}$ *has* $O(n)$ *rounds and communcation complexity* $\mathrm{poly}(n, \lambda, \log(p))$;
3. *The committer of* $\mathsf{Eval}$ *runs in time* $2^n \cdot \mathrm{poly}(n, \lambda, \log(p))$ *and space* $n \cdot \mathrm{poly}(\lambda, \log(p))$, *and uses* $O(n)$ *passes over the stream; and*
4. *The receiver of* $\mathsf{Eval}$ *runs in time* $\mathrm{poly}(n, \lambda, \log(p))$.

We present our scheme in Sect. 5.

*Proof-of-Exponentiation.* Our polynomial commitment scheme relies on a new Proof-of-Exponentiation ($\mathsf{PoE}$) protocol, which may be of independent interest.

For some group $\mathbb{G}$ and base $q \in \mathbb{Z}$ consider the language

$$\mathcal{L}_{\mathbb{G},q} = \left\{ (x, y, t) \in \mathbb{G} \times \mathbb{G} \times \mathbb{N} \; : \; x^{q^{2^t}} = y \right\}. \tag{5}$$

Note that this problem can be solved in time roughly $2^t$ (by repeated squaring), but for some groups it is conjectured to not be solvable in significantly less time (even when leveraging parallelization). Indeed, an instantiation of this language using RSA groups underlies the original time-lock puzzle construction by Rivest, Shamir and Wagner [39]. This problem has also been used recently for constructing *verifiable delay functions* (VDFs). We show a extension of a recent protocol due to Pietrzak [37] that works for general groups.

**Theorem 4.3.** *Let* $\mathbb{G}$ *be a group whose elements have* $O(\log(|\mathbb{G}|))$*-bit descriptions, and whose group operations take time* $\mathrm{polylog}(|\mathbb{G}|)$, *and let* $q \in \mathbb{N}$. *There exists a perfectly correct, statistically sound public-coin interactive-proof for* $\mathcal{L}_{\mathbb{G},q}$ *with the following efficiency properties for exponent parameter* $t$:

1. *The communication complexity is* $O(t\lambda^2 + t\lambda \log(|\mathbb{G}|))$ *and there are* $t$ *rounds.*
2. *The prover runs in time* $2^t \cdot \mathrm{poly}(\log(q), \log(|\mathbb{G}|), \lambda)$ *and uses space* $O(\lambda \cdot \log(|\mathbb{G}|)) + \log(t) + \log(q) + \lambda^2$
3. *The verifier runs in time* $t \cdot \mathrm{poly}(\log(|\mathbb{G}|), \log(q), \lambda)$.

# 5 Multilinear Polynomial Commitment Scheme in Hidden Order Groups

We describe our commitment scheme $(\mathsf{Setup}, \mathsf{Com}, \mathsf{isValid}, \mathsf{Eval})$ for multilinear polynomials $f\colon \mathbb{F}^n \to \mathbb{F}$ over some field $\mathbb{F}$ of prime-order $p$ which is specified as an input to $\mathsf{Setup}$. Throughout the section, we work with the description $\mathcal{Y} := (f(\mathbf{b}))_{\mathbf{b}\in\{0,1\}^n} \in \mathbb{F}^{2^n}$ of the multilinear polynomial $f$. First, in Sect. 5.1 we describe how to encode $\mathcal{Y}$ as an integer. Then, in Sect. 5.2 we describe our polynomial commitment scheme.

## 5.1 Encoding Multilinear Polynomials as an Integer

One key portion of our scheme is encoding the sequence $\mathcal{Y}$, which defines our multilinear polynomial, as an integer. We do so by using a technique first introduced by [21]. Towards this, we first describe an encoding scheme for *integer* sequences. For any $N = 2^n$ and an odd integer $q \in \mathbb{N}$, let $\mathsf{Enc}_q\colon \mathbb{Z}^N \to \mathbb{Z}$ be the function that encodes a sequence of integers $\mathcal{Z} \in \mathbb{Z}^N$ as[14]

$$\mathsf{Enc}_q(\mathcal{Z}) := \sum_{\mathbf{b}\in\{0,1\}^n} q^{\mathbf{b}} \cdot \mathcal{Z}_{\mathbf{b}},$$

where $q^{\mathbf{b}}$ interprets $\mathbf{b}$ (an $n$-bit string) as the naturally corresponding integer in the set $\{0, 1, \ldots, N-1\}$. To decode an integer $v \in \mathbb{Z}$, we output its base-$q$ representation where, for convenience, the base-$q$ digits of $v$ are integers in the range $[-q/2, q/2)$. We refer to the decoding function as $\mathsf{Dec}_q$.

Our $\mathsf{Enc}_q$ scheme has two homomorphic properties which we leverage to design our polynomial commitment. First, $\mathsf{Enc}_q(\cdot)$ is a linear homomorphism over $\mathbb{Z}$; that is, for any $\mathcal{Z}, \mathcal{Z}' \in \mathbb{Z}^N$ and $\alpha, \beta \in \mathbb{Z}$, it holds that $\alpha \cdot \mathsf{Enc}_q(\mathcal{Z}) + \beta \cdot \mathsf{Enc}_q(\mathcal{Z}') = \mathsf{Enc}_q(\alpha \cdot \mathcal{Z} + \beta \cdot \mathcal{Z}')$. Second, $\mathsf{Enc}_q(\cdot)$ satisfies a restricted form of multiplicative homomorphism; that is, for any $d \in \mathbb{N}$, we have $q^d \cdot \mathsf{Enc}_q(\mathcal{Z}) = \mathsf{Enc}_q((0^d, \mathcal{Z}))$.

*Encoding Bounded Integer Sequences.* In fact, looking ahead, we are interested in encoding only sequences of bounded integers. For some $B \in \mathbb{R}_{\geqslant 1}$, we let $\mathbb{Z}(B) := \{z \in \mathbb{Z}\colon -B \leqslant z < B\}$ be the set of integers whose absolute value is bounded by $B$. Then, to encode integer sequences in $\mathbb{Z}(B)^N$, we consider the restriction of $\mathsf{Enc}_q$ to the set $\mathbb{Z}(B)^N$. Notice that by definition, for any $\mathcal{Z} \in \mathbb{Z}(B)^N$, we have that $\mathsf{Enc}_q(\mathcal{Z}) \in \mathbb{Z}(B \cdot (q^N - 1)/(q-1))$. We remark that while $\mathsf{Enc}_q$ is not injective over all integer sequences (as integer sequences $(1+q, 0)$ and $(1, 1)$ both encode to the integer $1 + q$), the restriction of $\mathsf{Enc}_q$ to the set $\mathbb{Z}(q/2)^N$ *is* injective. We capture this in the following fact:

**Fact 5.1** **([21, Fact 1]).** *Let $q$ be any odd integer and let $N \in \mathbb{N}$. For any $v \in \mathbb{Z}(q^N/2)$, there exists a unique sequence $\mathcal{Z} \in \mathbb{Z}(q/2)^N$ such that $v = \mathsf{Enc}_q(\mathcal{Z})$. Furthermore, $\mathcal{Z} = \mathsf{Dec}_q(v)$.*

[14] This encoding is valid for sequences of arbitrary length, but we restrict to powers of two for convenience.

*Proof.* For any sequence $\mathcal{Z} \in \mathbb{Z}(q/2)^N$, by definition of $\mathsf{Dec}_q$ we observe that $\mathsf{Dec}_q(\mathsf{Enc}_q(\mathcal{Z})) = \mathcal{Z}$. This implies that (restriction of) $\mathsf{Enc}_q$ (to $\mathbb{Z}(q/2)^N$) is injective. Furthermore, the cardinality of sets $\mathbb{Z}(q^N/2)$ and $\mathbb{Z}(q/2)^N$ are equal. Therefore, for every $v \in \mathbb{Z}(q^N/2)$, $\mathsf{Dec}_q(v)$ is the unique sequence in $\mathbb{Z}(q/2)^N$ that encodes to $v$.

Similar to $\mathsf{Enc}_q$, $\mathsf{Dec}_q$ also satisfies some homomorphic properties: for integers $z_1, z_2$, we have that $\mathsf{Dec}_q(z_1 + z_2) = \mathsf{Dec}_q(z_1) + \mathsf{Dec}_q(z_2)$ as long as $z_1, z_2$ encode sequences whose elements are bounded by $q/4$. For our security proof, we will use the following more general statement , which we prove in the full version.

**Claim 5.2.** *Let $\ell, q, N \in \mathbb{N}$ such that $q$ is odd, and let $B_1, B_2 \geq 1$ be such that $B_1 \cdot B_2 \leq q/(2\ell)$. Then, for every $\alpha_1, \ldots, \alpha_\ell \in \mathbb{Z}(B_1)$, and integers $z_1, \ldots, z_\ell \in \mathbb{Z}(q^N/2)$ such that $\mathsf{Dec}_q(z_i) \in \mathbb{Z}(B_2)^N$,*

$$\mathsf{Dec}_q\Big(\sum_{i\in[\ell]} \alpha_i \cdot z_i\Big) = \sum_{i\in[\ell]} \alpha_i \cdot \mathsf{Dec}_q(z_i). \tag{6}$$

*Remark 5.3.* Looking ahead, the correctness of our extractor (to show security for our polynomial commitment scheme) relies crucially on Claim 5.2. The main issue with [21] is that their extractor relies on a variant of Claim 5.2 (formulated below) which is false. Lemma 8 in the full version [22] of [21] uses the following claim to argue correctness of the extracted integer decommitments $f_L$ and $f_R$.

**Claim 5.4 (False claim implicit in** [22, Lemma 8]**).** *For $p, q, N \in \mathbb{N}$ such that $2 \leq p \leq q$ where $q$ is odd. For every $\alpha \in \mathbb{Z}(p)$ and $z \in \mathbb{Z}(q^N/2)$ such that $\alpha \mid z$, $\mathsf{Dec}_q(z/\alpha) = \mathsf{Dec}_q(z)/\alpha$.*

We note that $z, z/\alpha \in \mathbb{Z}(q^N/2)$, by Fact 5.1 $\mathsf{Dec}_q(z), \mathsf{Dec}_q(z/\alpha) \in \mathbb{Z}(q/2)^N$. But, $\mathsf{Dec}_q(z)/\alpha$ may not be an integer sequence. Counter-example: for $z = 1+q, \alpha = 2$, we have $\mathsf{Dec}_q(z) = (1, 1)$ but $\mathsf{Dec}_q(z)/2$ is not an integer sequence.

*Encoding $\mathcal{Y}$.* To encode $\mathcal{Y} \in \mathbb{F}^N$ where $\mathbb{F}$ is a field of prime-order $p$, we first define a lifting function $[\![\cdot]\!] \colon \mathbb{F} \to \mathbb{Z}(p/2)$ in the natural way. That is, for any $\alpha \in \mathbb{F}$, we define $[\![\alpha]\!]$ to be the unique integer in $\mathbb{Z}(p/2)$ such that $[\![\alpha]\!] \equiv \alpha \bmod p$. We then define $\mathsf{Enc}_q(\mathcal{Y})$ as $\mathsf{Enc}_q(\mathcal{Y}) := \sum_{\mathbf{b}\in\{0,1\}^n} q^{\mathbf{b}} \cdot [\![\mathcal{Y}_{\mathbf{b}}]\!]$.

### 5.2 Scheme

Our polynomial commitment scheme is parameterized by three components: (a) the encoding scheme $(\mathsf{Enc}_q, \mathsf{Dec}_q)$ defined in Sect. 5.1, (b) A group sampler $\mathcal{G}$ for which the Hidden Order Assumption holds (see Sect. 3.2 for a discussion on candidates), and (c) a perfectly correct, statistically sound $\mathsf{PoE}$ protocol (we present one such protocol over arbitrary groups the full version). We now present all algorithms $(\mathsf{Setup}, \mathsf{Com}, \mathsf{isValid}, \mathsf{Eval})$ for the polynomial commitment scheme.

$\underline{\mathsf{Setup}(1^\lambda, p, 1^n)}$*:* On input security parameter $1^\lambda$, a prime $p$, and the number of polynomial variables $1^n$, the algorithm $\mathsf{Setup}$ samples group description $\mathbb{G} \leftarrow \mathcal{G}(1^\lambda)$, samples $g \leftarrow \mathbb{G}$, sets $q := q(n, p, \lambda) \in \mathbb{N}$, and outputs public parameters $pp = (q, g, \mathbb{G})$. We require that $q$ be odd such that $q > p \cdot 2^{n \cdot \mathrm{poly}(\lambda)}$.

$\underline{\mathsf{Com}(pp, \mathcal{Y})}$*:* On input $pp = (q, g, \mathbb{G})$ output by $\mathsf{Setup}$ and sequence $\mathcal{Y}$, $\mathsf{Com}$ computes a commitment to the sequence $\mathcal{Y}$ as $C = g^{\mathsf{Enc}_q(\mathcal{Y})}$. The output of $\mathsf{Com}$ is the commitment $C$ and secret decommitment $\mathcal{Z} = ([\![\mathcal{Y}_{\mathbf{b}}]\!])_{\mathbf{b} \in \{0,1\}^n} \in \mathbb{Z}(p/2)^N$.

$\underline{\mathsf{isValid}(pp, C, \mathcal{Y}, \mathcal{Z})}$*:* On inputs $pp = (q, g, \mathbb{G})$, $C$ output by $\mathsf{Com}$, committed sequence $\mathcal{Y} \in \mathbb{F}^N$ and decommitment $\mathcal{Z} \in \mathbb{Z}^N$ for $N = 2^n$, the algorithm $\mathsf{isValid}$ outputs a decision bit. $\mathsf{isValid}$ outputs 1 if and only if (1) $\mathcal{Z} \subseteq \mathbb{Z}(q/2)^N$; (2) $\mathcal{Y} \equiv \mathcal{Z} \bmod p$; and (3) $C = g^{\mathsf{Enc}_q(\mathcal{Z})}$. Otherwise, $\mathsf{isValid}$ outputs 0.

$\underline{\mathsf{Eval}(pp, C, \boldsymbol{\zeta}, \gamma; \mathcal{Y}, \mathcal{Z})}$*:* On input $pp = (q, g, \mathbb{G})$, $C \in \mathbb{G}$, $\boldsymbol{\zeta} \in \mathbb{F}^n$, $\gamma \in \mathbb{F}$, $\mathcal{Y} \in \mathbb{F}^N$ and $\mathcal{Z} \in \mathbb{Z}^N$ for $N = 2^n$, $\mathsf{Eval}$ is an interactive protocol $(P, V)$ for the relation,

$$\mathcal{R}_{\mathsf{ml}} = \Big\{(pp, C, \boldsymbol{\zeta}, \gamma; \mathcal{Y}, \mathcal{Z}) : \mathsf{isValid}(pp, C, \mathcal{Y}, \mathcal{Z}) = 1 \wedge \gamma = \mathsf{ML}(\mathcal{Y}, \boldsymbol{\zeta})\Big\}, \quad (7)$$

where on common input $(pp, C, \boldsymbol{\zeta}, \gamma)$, $P$ tries to convince $V$ that it knows a committed sequence $\mathcal{Y} \in \mathbb{F}^N$ and an integer sequence $\mathcal{Z} \in \mathbb{Z}^N$ such that $\mathsf{isValid}(pp, C, \mathcal{Y}, \mathcal{Z}) = 1$ and $\gamma$ is the evaluation of the multilinear polynomial defined by $\mathcal{Y}$ at evaluation point $\boldsymbol{\zeta} = (\zeta_n, \ldots, \zeta_1)$; that is, $\gamma \overset{?}{=} \mathsf{ML}(\mathcal{Y}, \boldsymbol{\zeta})$. More specifically, both the committer and receiver in $\mathsf{Eval}$ first make $\lambda$ many copies of the statement $(C, \boldsymbol{\zeta}, \gamma; \mathcal{Z})$ as $(\mathbf{C}, \boldsymbol{\zeta}, \boldsymbol{\gamma}; Z)$, where $\mathbf{C} = (C, \ldots, C) \in \mathbb{G}^\lambda$, $\boldsymbol{\gamma} = (\gamma, \ldots, \gamma) \in \mathbb{F}^\lambda$, and $Z \in \mathbb{Z}^{\lambda \times N}$ is a matrix such that $Z(i, \mathbf{b}) := \mathcal{Z}_{\mathbf{b}}$ for every $i \in [\lambda]$ and $\mathbf{b} \in \{0,1\}^n$. The committer and receiver then run the subroutine $\mathsf{MultiEval}$, presented in Algorithm 1.

$\mathsf{MultiEval}$ is a recursive protocol which given the statement $(\mathbf{C}, \boldsymbol{\zeta}, \boldsymbol{\gamma}; Z)$ proves that $\gamma_i = \mathsf{ML}(Z(i, *), \boldsymbol{\zeta})$ and $C_i = \mathsf{Com}(Z(i, *))$ for every $i \in [\lambda]$, where $Z(i, *) \in \mathbb{Z}^{1 \times N}$ is the $i$þrow of $Z$. This is done via a divide and conquer approach. Let $P_i \colon \mathbb{F}^n \to \mathbb{F}$ be the multilinear polynomial defined by row $i$ of matrix $Z$ for every $i \in [\lambda]$. For presentation, we focus on the polynomial $P_1$. To prove that $\gamma_1 = P_1(\boldsymbol{\zeta})$ and $C_1 = \mathsf{Com}(P_1) = g^{\mathsf{Enc}_q(P_1)}$, the committer first splits $P_1$ into it's "left" and "right" halves, defined by $P_{1,\mathsf{L}}(\cdot) = P_1(\cdot, 0)$ and $P_{1,\mathsf{R}}(\cdot, 1)$. Then it computes evaluations of these polynomials at the point $\boldsymbol{\zeta}' = (\zeta_n, \ldots, \zeta_2)$ to obtain $\gamma_{1,\mathsf{L}} = P_{1,\mathsf{L}}(\boldsymbol{\zeta}')$ and $\gamma_{1,\mathsf{R}} = P_{1,\mathsf{R}}(\boldsymbol{\zeta}')$ (Line 5). Similarly, the committer also computes commitments $C_{1,\mathsf{L}} = g^{\mathsf{Enc}_q(P_{1,\mathsf{L}})}$ and $C_{1,\mathsf{R}} = g^{\mathsf{Enc}_q(P_{1,\mathsf{R}})}$ (Line 6). The claims $(\gamma_{1,\mathsf{L}}, \gamma_{1,\mathsf{R}})$ and $(C_{1,\mathsf{L}}, C_{1,\mathsf{R}})$ are then sent to the receiver. If indeed the committer defined $P_{1,\mathsf{L}}$ and $P_{1,\mathsf{R}}$ correctly, then $\gamma_1 = \gamma_{1,\mathsf{L}} \cdot (1 - \zeta_1) + \gamma_{1,\mathsf{R}} \cdot \zeta_1$ (Line 8) and $C_{1,\mathsf{L}} \cdot C_{1,\mathsf{R}}^{q^T} = C_1$ for $T = 2^{n-1}$. Since checking $C_{1,\mathsf{L}} \cdot C_{1,\mathsf{R}}^{q^T} = C_1$ directly is too costly to the receiver, the committer and prover run a *proof of exponent* protocol $\mathsf{PoE}$ to prove that equality holds (Line 9). The committer does simultaneously this for all polynomials $P_i$. The receiver then specifies random linear combinations $U \leftarrow \{0,1\}^{\lambda \times 2\lambda}$ (Line 10). The committer and receiver then obtain a set of $\lambda$ new evaluations $\gamma_i' = \sum_{j \in [\lambda]} U(i, j) \cdot \gamma_{j,\mathsf{L}} + U(i, 2j) \cdot \gamma_{j,\mathsf{R}}$ and $\lambda$ new

```
Algorithm 1: $\mathsf{MultiEval}(\mathbf{C}, k, \boldsymbol{\zeta}, \boldsymbol{\gamma}; Z)$
   Input  : $\mathbf{C} \in \mathbb{G}^\lambda$, $k \in \mathbb{N}$, $\boldsymbol{\zeta} \in \mathbb{F}^n$, $\boldsymbol{\gamma} \in \mathbb{F}^\lambda$, and $Z \in \mathbb{Z}^{\lambda \times 2^{n-k}}$.
   Output : Accept or reject.
1  if $k = n$ then
2      $P$ sends $Z \in \mathbb{Z}^\lambda$ to $V$.
3      $V$ outputs accept if and only if $\|Z\|_\infty \leq p(2\lambda)^n$, $\boldsymbol{\gamma} \equiv Z \bmod p$, and $\mathbf{C} = g^Z$.
4  else
5      $P$ computes
           $\boldsymbol{\gamma}_\mathsf{L} = \sum_{\mathbf{b} \in \{0,1\}^{n-k-1}} (Z(*, 0\mathbf{b}) \bmod p) \cdot \prod_{j=1}^{n-k-1} \chi(b_j, \zeta_{j+k+1})$
           $\boldsymbol{\gamma}_\mathsf{R} = \sum_{\mathbf{b} \in \{0,1\}^{n-k-1}} (Z(*, 1\mathbf{b}) \bmod p) \cdot \prod_{j=1}^{n-k-1} \chi(b_j, \zeta_{j+k+1})$
6      $P$ computes
           $\mathbf{C}_\mathsf{L} = g^{\boldsymbol{\ell}}$    where $\boldsymbol{\ell} = \sum_{\mathbf{b} \in \{0,1\}^{n-k-1}} q^{\mathbf{b}} \cdot Z(*, 0\mathbf{b})$
           $\mathbf{C}_\mathsf{R} = g^{\mathbf{r}}$    where $\mathbf{r} = \sum_{\mathbf{b} \in \{0,1\}^{n-k-1}} q^{\mathbf{b}} \cdot Z(*, 1\mathbf{b})$
7      $P$ sends $(\boldsymbol{\gamma}_\mathsf{L}, \boldsymbol{\gamma}_\mathsf{R})$ and $(\mathbf{C}_\mathsf{L}, \mathbf{C}_\mathsf{R})$ to $V$.
8      $V$ checks $\boldsymbol{\gamma} \overset{?}{=} \boldsymbol{\gamma}_\mathsf{L} \cdot (1 - \zeta_{k+1}) + \boldsymbol{\gamma}_\mathsf{R} \cdot \zeta_{k+1}$.
9      $P$ and $V$ run $\mathsf{PoE}(\mathbf{C}_\mathsf{R}, \mathbf{C}/\mathbf{C}_\mathsf{L}, q, n-k-1, \lambda)$ which is a proof showing
       $\mathbf{C}_\mathsf{R}(i)^{q^{2^{n-k-1}}} = \mathbf{C}(i)/\mathbf{C}_\mathsf{L}(i)$ for every $i \in [\lambda]$. Here, $\mathbf{C}/\mathbf{C}_\mathsf{L}$ denotes
       coordinate-wise division of the elements of $\mathbf{C}$ by the elements of $\mathbf{C}_\mathsf{L}$.
10     $V$ samples $U = [U_\mathsf{L} \| U_\mathsf{R}] \leftarrow \{0,1\}^{\lambda \times 2\lambda}$ and sends $U$ to $P$, where
       $U_\mathsf{L}, U_\mathsf{R} \in \{0,1\}^{\lambda \times \lambda}$.
11     $P$ and $V$ compute
           $\boldsymbol{\gamma}' = U_\mathsf{L} \cdot \boldsymbol{\gamma}_\mathsf{L} + U_\mathsf{R} \cdot \boldsymbol{\gamma}_\mathsf{R}$    $\mathbf{C}' = (U_\mathsf{L} \star \mathbf{C}_\mathsf{L}) \odot (U_\mathsf{R} \star \mathbf{C}_\mathsf{R})$
12     For $Z_\mathsf{L}, Z_\mathsf{R} \in \{0,1\}^{\lambda \times 2^{n-k-1}}$ such that $Z = [Z_\mathsf{L} \| Z_\mathsf{R}]$, $P$ computes
           $Z' = U_\mathsf{L} \cdot Z_\mathsf{L} + U_\mathsf{R} \cdot Z_\mathsf{R}$.
13     return $\mathsf{MultiEval}(\mathbf{C}', k+1, \boldsymbol{\zeta}, \boldsymbol{\gamma}'; Z')$
```

committments $C'_i = \prod_{j \in [\lambda]} (C_{j,\mathsf{L}})^{U(i,j)} \cdot (C_{j,\mathsf{R}})^{U(i,2j)}$ (Line 11). This also defines new matrix $Z' = U_\mathsf{L} \cdot Z_\mathsf{L} + U_\mathsf{R} \cdot Z_\mathsf{R}$ (Line 12) for $U = [U_\mathsf{L} \| U_\mathsf{R}]$ and $Z = [Z_\mathsf{L} \| Z_\mathsf{R}]$. If the committer is honest, then the polynomial $P'_1$ defined by the row $Z'(1, *)$

satisfies $\gamma_1' = P_1'(\boldsymbol{\zeta}')$ and $C_1' = g^{\mathsf{Enc}_q(P_1')}$ (and similarly for all other polynomials $P_i'$ defined by row $Z'(i,*)$). The committer and receiver recurse via the above $\lambda$-to-$2\lambda$-to-$\lambda$ reduction until the matrix $Z$ is a single column; at this point, $Z$ is sent to the receiver. The receiver checks (Line 3) if the entries of $Z$ are appropriately bounded, if the final vector $\boldsymbol{\gamma} \equiv Z (\bmod p)$, and if $\mathbf{C} = g^Z = (g^{Z_1}, \ldots, g^{Z_\lambda})$.

*Remark 5.5.* For simplicity of presentation, we let the (computational) security parameter $\lambda_c$ given as input to $\mathsf{Setup}$ to be equal to the statistical security parameter $\lambda_s$ given to $\mathsf{Eval}$. However, they may be set differently: $\lambda_c$ needs to be set so that $2^{\lambda_c^\epsilon}$ is larger than the running time of the adversary (generally, $\lambda_c = 2048$ for RSA groups to have security against $2^{80}$ time adversaries). However, $\lambda_s$ needs to set so that the success probability of the adversary (we want to tolerate) is upperbounded by $2^{-\Omega(\lambda_s)}$, in fact, even relatively small values of $\lambda_s$ would be sufficient for security, and offer qualitatively more efficient implementations.

**Acknowledgments.** Alexander R. Block was supported in part by NSF grant CCF-1910659. Pratik Soni was supported in part by the NSF award 1916939, DARPA SIEVE program, a gift from Ripple, a DoE NETL award, a JP Morgan Faculty Fellowship, a PNC center for financial services innovation award, and a Cylab seed funding award. Ron Rothblum was supported in part by a Milgrom family grant, by the Israeli Science Foundation (Grants No. 1262/18 and 2137/19), and grants from the Technion Hiroshi Fujiwara cyber security research center and Israel cyber directorate. Alon Rosen is supported in part by ISF grant No. 1399/17 and Project PROMETHEUS (Grant 780701).

## References

1. Cash for RSA assumptions. https://rsa.cash/rsa-assumptions/
2. Aly, A., Ashur, T., Ben-Sasson, E., Dhooghe, S., Szepieniec, A.: Design of symmetric-key primitives for advanced cryptographic protocols. Cryptology ePrint Archive, Report 2019/426 (2019). https://eprint.iacr.org/2019/426
3. Babai, L., Moran, S.: Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. JCSS **36**(2), 254–276 (1988). https://doi.org/10.1016/0022-0000(88)90028-1. https://www.sciencedirect.com/science/article/pii/0022000088900281
4. Ben-Or, M., et al.: Everything provable is provable in zero-knowledge. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 37–56. Springer, New York (1990). https://doi.org/10.1007/0-387-34799-2_4
5. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Fast Reed-Solomon interactive oracle proofs of proximity. In: Chatzigiannakis, I., Kaklamanis, C., Marx, D., Sannella, D. (eds.) 45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, Prague, Czech Republic, 9–13 July 2018. LIPIcs, vol. 107, pp. 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018). https://doi.org/10.4230/LIPIcs.ICALP.2018.14
6. Ben-Sasson, E., Carmon, D., Ishai, Y., Kopparty, S., Saraf, S.: Proximity gaps for Reed-Solomon codes. In: FOCS (2020)
7. Ben-Sasson, E., Goldberg, L., Kopparty, S., Saraf, S.: DEEP-FRI: sampling outside the box improves soundness. Cryptology ePrint Archive, Report 2019/336 (2019). https://eprint.iacr.org/2019/336

8. Ben-Sasson, E., Goldberg, L., Kopparty, S., Saraf, S.: DEEP-FRI: sampling outside the box improves soundness. In: Vidick, T. (ed.) ITCS (2020)
9. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: Recursive composition and bootstrapping for SNARKS and proof-carrying data. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC, pp. 111–120. ACM Press, June 2013. https://doi.org/10.1145/2488608.2488623
10. Bitansky, N., Chiesa, A.: Succinct arguments from multi-prover interactive proofs and their efficiency benefits. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 255–272. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_16
11. Block, A.R., Holmgren, J., Rosen, A., Rothblum, R.D., Soni, P.: Public-coin zero-knowledge arguments with (almost) minimal time and space overheads. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 168–197. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64378-2_7
12. Block, A.R., Holmgren, J., Rosen, A., Rothblum, R.D., Soni, P.: Time- and space-efficient arguments from groups of unknown order. Cryptology ePrint Archive, Report 2021/358 (2021). https://eprint.iacr.org/2021/358
13. Blumberg, A.J., Thaler, J., Vu, V., Walfish, M.: Verifiable computation using multiple provers. Cryptology ePrint Archive, Report 2014/846 (2014). http://eprint.iacr.org/2014/846
14. Boneh, D., Bonneau, J., Bünz, B., Fisch, B.: Verifiable delay functions. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 757–788. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_25
15. Boneh, D., Bünz, B., Fisch, B.: A survey of two verifiable delay functions. Cryptology ePrint Archive, Report 2018/712 (2018). https://eprint.iacr.org/2018/712
16. Boneh, D., Bünz, B., Fisch, B.: Batching techniques for accumulators with applications to IOPs and stateless blockchains. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 561–586. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26948-7_20
17. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 327–357. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_12
18. Bowe, S., Grigg, J., Hopwood, D.: Halo: recursive proof composition without a trusted setup. Cryptology ePrint Archive, Report 2019/1021 (2019). https://eprint.iacr.org/2019/1021
19. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy, pp. 315–334. IEEE Computer Society Press, May 2018. https://doi.org/10.1109/SP.2018.00020
20. Bünz, B., Chiesa, A., Mishra, P., Spooner, N.: Recursive proof composition from accumulation schemes. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 1–18. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64378-2_1
21. Bünz, B., Fisch, B., Szepieniec, A.: Transparent SNARKs from DARK compilers. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 677–706. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_24
22. Bünz, B., Fisch, B., Szepieniec, A.: Transparent snarks from DARK compilers. IACR Cryptology ePrint Archive 2019, 1229 (20200226:080105 (posted 26-Feb-2020 08:01:05 UTC)). https://eprint.iacr.org/2019/1229

23. Bünz, B., Fisch, B., Szepieniec, A.: Personal Communication (2021)
24. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.: Marlin: preprocessing zkSNARKs with universal and updatable SRS. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 738–768. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_26
25. Chiesa, A., Ojha, D., Spooner, N.: Fractal: post-quantum and transparent recursive proofs from holography. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 769–793. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_27
26. Damgård, I., Fujisaki, E.: A statistically-hiding integer commitment scheme based on groups with hidden order. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 125–142. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36178-2_8
27. Damgård, I., Koprowski, M.: Generic lower bounds for root extraction and signature schemes in general groups. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 256–271. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_17
28. Ephraim, N., Freitag, C., Komargodski, I., Pass, R.: SPARKs: succinct parallelizable arguments of knowledge. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 707–737. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_25
29. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12
30. Goldreich, O.: Computational Complexity - A Conceptual Perspective. Cambridge University Press, New York (2008). https://doi.org/10.1017/CBO9780511804106
31. Holmgren, J., Rothblum, R.: Delegating computations with (almost) minimal time and space overhead. In: Thorup, M. (ed.) 59th FOCS, pp. 124–135. IEEE Computer Society Press, October 2018. https://doi.org/10.1109/FOCS.2018.00021
32. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 177–194. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_11
33. Kattis, A., Panarin, K., Vlasov, A.: RedShift: transparent SNARKs from list polynomial commitment IOPs. Cryptology ePrint Archive, Report 2019/1400 (2019). https://eprint.iacr.org/2019/1400
34. Lee, J.: Dory: efficient, transparent arguments for generalised inner products and polynomial commitments. Cryptology ePrint Archive, Report 2020/1274 (2020). https://eprint.iacr.org/2020/1274
35. Lindell, Y.: Parallel coin-tossing and constant-round secure two-party computation. J. Cryptol. **16**(3), 143–184 (2003). https://doi.org/10.1007/s00145-002-0143-7
36. Papamanthou, C., Shi, E., Tamassia, R.: Signatures of correct computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 222–242. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_13
37. Pietrzak, K.: Simple verifiable delay functions. In: Blum, A. (ed.) ITCS 2019. LIPIcs, vol. 124, pp. 60:1–60:15, January 2019. https://doi.org/10.4230/LIPIcs.ITCS.2019.60
38. Reingold, O., Rothblum, G.N., Rothblum, R.D.: Constant-round interactive proofs for delegating computation. In: Wichs, D., Mansour, Y. (eds.) 48th ACM STOC, pp. 49–62. ACM Press, June 2016. https://doi.org/10.1145/2897518.2897652

39. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto. Technical report, USA (1996)
40. Rubinfeld, R., Sudan, M.: Robust characterizations of polynomials with applications to program testing. SIAM J. Comput. **25**(2), 252–271 (1996). https://doi.org/10.1137/S0097539793255151
41. Setty, S.: Spartan: efficient and general-purpose zkSNARKs without trusted setup. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 704–737. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56877-1_25
42. Setty, S., Lee, J.: Quarks: quadruple-efficient transparent zkSNARKs. Cryptology ePrint Archive, Report 2020/1275 (2020). https://eprint.iacr.org/2020/1275
43. Tomescu, A.: Cryptographic assumptions in hidden-order groups. https://alinush.github.io/2020/11/05/cryptographic-assumptions-in-hidden-order-groups.html
44. Valiant, P.: Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 1–18. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78524-8_1
45. Wahby, R.S., Tzialla, I., Shelat, A., Thaler, J., Walfish, M.: Doubly-efficient zkSNARKs without trusted setup. In: 2018 IEEE Symposium on Security and Privacy, pp. 926–943. IEEE Computer Society Press, May 2018. https://doi.org/10.1109/SP.2018.00060
46. Wesolowski, B.: Efficient verifiable delay functions. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part III. LNCS, vol. 11478, pp. 379–407. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17659-4_13
47. Wijesekera, P., et al.: The feasibility of dynamically granted permissions: aligning mobile privacy with user preferences. In: 2017 IEEE Symposium on Security and Privacy, pp. 1077–1093. IEEE Computer Society Press, May 2017. https://doi.org/10.1109/SP.2017.51
48. Zhang, J., Xie, T., Zhang, Y., Song, D.: Transparent polynomial delegation and its applications to zero knowledge proof. In: 2020 IEEE Symposium on Security and Privacy, pp. 859–876. IEEE Computer Society Press, May 2020. https://doi.org/10.1109/SP40000.2020.00052