Transparent SNARKs from DARK Compilers

Benedikt Bünz[1] benedikt@cs.stanford.edu
Ben Fisch[1] bfisch@cs.stanford.edu
Alan Szepieniec[2] alan@nervos.org

[1]Stanford University, Findora Foundation
[2]Nervos Foundation

Abstract

We construct a new polynomial commitment scheme for univariate and multivariate polynomials over finite fields, with logarithmic size evaluation proofs and verification time, measured in the number of coefficients of the polynomial. The underlying technique is a *Diophantine Argument of Knowledge* (DARK), leveraging integer representations of polynomials and groups of unknown order. Security is shown from the strong RSA and the adaptive root assumptions. Moreover, the scheme does not require a trusted setup if instantiated with class groups. We apply this new cryptographic compiler to a restricted class of algebraic linear IOPs, which we call *Polynomial IOPs*, to obtain doubly-efficient public-coin interactive arguments of knowledge for any NP relation with succinct communication. With linear preprocessing, the online verifier's work is logarithmic in the circuit complexity of the relation.

There are many existing examples of Polynomial IOPs (PIOPs) dating back to the first PCP (BFLS, STOC'91). We present a generic compilation of any PIOP using our DARK polynomial commitment scheme. In particular, compiling the PIOP from PLONK (GWC, ePrint'19), an improvement on Sonic (MBKM, CCS'19), yields a public-coin interactive argument with quasi-linear preprocessing, quasi-linear (online) prover time, logarithmic communication, and logarithmic (online) verification time in the circuit size. Applying Fiat-Shamir results in a SNARK, which we call **Supersonic**.

Supersonic is also concretely efficient with 10KB proofs and under 100ms verification time for circuits with 1 million gates (estimated for 120-bit security). Most importantly, this SNARK is *transparent*: it does not require a trusted setup. We obtain zk-SNARKs by applying a hiding variant of our polynomial commitment scheme with zero-knowledge evaluations. Supersonic is the first complete zk-SNARK system that has both a practical prover time as well as asymptotically *logarithmic* proof size and verification time.

1 Introduction

Since the landmark discoveries of *interactive proofs* (IPs) [GMR85] and *probabilistically checkable proofs* (PCPs) [BFLS91, ALM+92] in the 90s, there has been tremendous development in the area of proof systems whereby a prover establishes the correct performance of an arbitrary computation in a way that can be verified much more efficiently than performing the computation itself. Such proof systems are *succinct* if they also have a low communication cost between the prover and the verifier, *i.e.*, the transcript of the protocol is much smaller than a witness to the computation. There are also *zero knowledge* variants of these efficient proof systems, beginning with ZK-IPs [BGG+88] and ZK-PCPs [Kil92], in which the computation may involve secret information and the prover demonstrates correct performance without leaking the secrets. As a toy example, one could prove that a chess position is winning for white without actually revealing the winning moves themselves. General purpose zero-knowledge proofs [GMW91] can be very expensive in terms of proof size and verification time even for computations that would be easy to perform given the secret inputs (*e.g.*, by proving that one decrypted a file properly without leaking the key or the plaintext). The same techniques that are used to build efficient proof systems for expensive computations are also useful for making zero-knowledge proofs more practical.

In recent years, there has been a surge of industry interest in verifiable outsourced computation [WB15] (such as trustless cloud computing) as well as zero-knowledge proofs. In particular, blockchains use efficient zero-knowledge proofs as a solution for balancing privacy and publicly-verifiable integrity: examples include anonymous transactions in ZCash [BCG+14, Zca, HBHW19] and verifying Ethereum smart contracts over private inputs [Ebe]. In these applications, zero-knowledge proofs are posted to the blockchain ledger as a part of transactions and nodes must verify many proofs in the span of a short period of time. Therefore, succinctness and fast verification are necessary properties for the deployment of such proof systems. Verifiable computation is also being explored as a scaling solution for blockhain transactions [But16], and even as a way to entirely eliminate the need for maintaining historical blockchain data [Lab18].

Following this pragmatic interest, there has also been a surge of research focused on obtaining proof systems with better concrete efficiency characteristics: *succinctness* (the proof size is sublinear in the original computation length T), *non-interactivity* (the proof is a single message), *prover-scalability* (proof generation time scales linearly or quasi-linearly in T), and *verifier-scalability* (verification time is sublinear in T). Proof systems that achieve all of these properties for general NP statements are called SNARGs ("succinct non-interactive arguments"). The proof is called an *argument* when it is only sound assuming the prover is computationally bounded, *i.e.*, *computationally sound* as opposed to statistically sound. Succinct statistically sound proofs are unlikely to exist [GVW02, Wee05].

Currently, there are numerous constructions that achieve different tradeoffs between proof size, proof time, and verification time, but also under different *trust* models as well as cryptographic assumptions. Some constructions also achieve better efficiency by relying on a *preprocessing model* in which a one-time expensive setup procedure is performed in order to generate a compact verification key VK, which is later used to verify proof instances efficiently. Somewhat unfortunately, the best performing proof systems to date (considering proof size and verification time) require a *trusted* preprocessing. These are the pairing-based SNARKs extending from GGPR [GGPR13, SBV+13, BCI+13, BCG+13, Gro16], which have been implemented in numerous libraries [BCG+13, Bow16], and even deployed in live systems such as the ZCash [Zca] cryptocurrency. The trusted setup can be performed via *multi-party computation* (MPC) by a committee of parties, such that trust in only one of the parties is sufficient. This has been done on two occasions for the ZCash blockchain, involving elaborate "ceremonies" to engender public trust in the process [Wil16].

A proof system is called *transparent* if it does not involve any trusted setup. Recent progress has yielded transparent proof systems for special types of computations: zk-STARKs [BBHR19] generate zero-knowledge proofs of size $O(\log^2 T)$ for a uniform computation[1], and the GKR protocol produces interactive proofs with communication $O(d \log T)$ for computations expressed as low-depth circuits of total size T and depth d [GKR08]. In both cases, non-interactivity can be achieved in the random oracle model with the Fiat-Shamir heuristic [FS87, CCH+19]. These transparent proof systems perform significantly worse than SNARKs based on preprocessing. For computations expressed as an arithmetic circuit of 1-million gates, STARKs [BBHR19] report a proof size of 600KB, whereas preprocessing SNARKs achieve 200 bytes [Gro16]. Bulletproofs [BBB+18, BCC+16a] is a transparent zero-knowledge proof system whose proofs are much smaller than those of STARK, but these proofs have a verification time that scales linearly in the size of the circuit; for an arithmetic circuit of one million gates the verification time is close to 1 minute, more than 1,000 times more expensive than verifying a STARK proof for the same computation.

Another thread of research has produced proof systems that remove trust from the circuit preprocessing step, and instead have a *universal* (trusted) setup: a one-time trusted setup that can be reused for *any* computation [MBKM19, XZZ+19, GWC19]. All of these systems build SNARKs by combining an underlying reduction of circuit satisfiability to probabilistic testing of polynomials (with degree at most linear in the circuit size) together with *polynomial commitment schemes*. In a polynomial commitment scheme, a prover commits to a μ-variate polynomial f over $\mathbb{F}$ of total degree at most d with a message that is much smaller than sending all the coefficients of f. The prover can later produce a non-interactive argument that $f(z) = y$ for arbitrary $z \in \mathbb{F}^\mu$ and $y \in \mathbb{F}$. The trusted portion of the universal SNARK is entirely confined to the polynomial commitment scheme's setup. These constructions use

[1] A uniform computation is expressed as a RAM program P and a time bound T on the running time of the program. A uniform computation depends on the size of P's description but not on the time bound T.

variants of the Kate *et al.* commitment scheme for univariate polynomials [KZG10], which requires a trusted setup.

1.1 Summary of contributions

Following the observations of the recent universal SNARK constructions [GWC19, MBKM19, XZZ+19], SNARKs can be built from polynomial commitment schemes where all the trust is confined to the setup of the commitment scheme. The main technical contribution of our work is thus a new polynomial commitment scheme without trusted setup (*i.e.*, a transparent polynomial commitment scheme), which we can use to construct transparent SNARKs. The observation that transparent polynomial commitments imply transparent SNARKs was also implicit in the recent works that build transparent SNARKs from multi-round classical PCPs, and specifically interactive oracle proofs of proximity (IOPPs) [BBHR18]. As a secondary contribution, we present a framework that unifies all existing approaches to constructing SNARKs from polynomial commitments using the language of *interactive oracle proofs* (IOPs) [RRR16, BCS16]. We view polynomial commitment schemes as a compiler for *Polynomial IOPs*, and re-characterize the results of prior works as providing a variety of Polynomial IOPs for NP.

New polynomial commitment scheme We construct a new polynomial commitment scheme for μ-multivariate polynomials of total degree d with optional zero-knowledge arguments of knowledge for correct evaluation that have $O(\mu \log d)$ size proofs and are verifiable in $O(\mu \log d)$ time. The commitment scheme requires a group of unknown order: two candidate instantiations are RSA groups and class groups of an imaginary quadratic order. Using RSA groups, we can apply the scheme to obtain universal preprocessing SNARKs with *constant-size* setup parameters, as opposed to the linear-size parameters from previous attempts. Using class groups, we can remove the trusted setup from trusted-setup SNARKs altogether, thereby making them *transparent*. Our polynomial commitment scheme leverages the power of integer commitments and *Diophantine Arguments of Knowledge* [Lip03]; accordingly, we classify this tool (and others of its kind) as a *DARK* proof system.

Polynomial IOP formalism All SNARK constructions can be viewed as combining an underlying information-theoretic statistically-sound protocol with a "cryptographic compiler" that transforms the underlying protocol into a succinct argument at the cost of computational soundness. We define a *Polynomial IOP* as a refinement of algebraic linear IOPs [IKO07, BCI+13, BBC+19], where in each round of interaction the prover provides the verifier with oracle access to a multivariate polynomial function of bounded degree. The verifier may then query this oracle to evaluate the polynomial on arbitrary points of its choice. The existing universal and transparent SNARK constructions provide a variety of statistically-sound Polynomial IOPs for circuit satisfiability (or RAM programs, in the case of STARKs); these are then cryptographically compiled using some form of a polynomial commitment, typically using Merkle trees or pairing groups.

The linear PCPs underlying GGPR and its successors (*i.e.*, based on QAPs and R1CS) can also be transformed into Polynomial IOPs.[2] This transformation helps highlight the fundamental paradigm shift between constructions of non-transparent non-universal SNARKs that combine linear PCPs and *linear-only encodings* versus the more recent ones based on polynomial commitments: given the lack of efficient[3] *linear function* commitment schemes, the compilation of linear PCPs *necessarily* involves a trusted preprocessing step that preselects the verifier's linear PCP queries, and hides them inside a linear-only encoding. This linear-only encoding forces the prover to homomorphically output an (encoded) linear tranformation of the query, upon which the verifer performs several homomorphic checks (*e.g.*, using pairings). The shift towards Polynomial IOPs, which can be compiled more directly with efficient polynomial commitments, avoids the involvement of a trusted party to place hidden queries in the preprocessing. The only potential need for non-transparent setup is in the instantiation of polynomial commitment itself.

[2]This observation was also implicit in the paper by Ben-Sasson *et al.* introducing the system Aurora [BCR+19].

[3]Lai and Malavota [LM19] provide a n-dimensional *linear-map* commitment based on bilinear pairings, extending techniques in functional commitments [LRY16], but verifying claimed evaluations of the committed function on query points takes $O(n)$.

The precise definition of Polynomial IOPs as a central and standalone notion raises the question about its exact relation to other IOP notions. We present a univariate Polynomial IOP for extracting an indicated coefficient of a polynomial. Furthermore, we present a univariate Polynomial IOP for proving that the inner product between the coefficient vectors of two polynomials equals a given value. This proof system is of independent interest. Together with an offline pre-processing phase during which the correctness of a multivariate polynomial is ascertained, these two tools enable us to show that *any* algebraic linear IOP can be realized with a multivariate Polynomial IOP.

Polynomial IOP compiler We present a generic compilation of any public-coin Polynomial IOP into a doubly-efficient public-coin interactive argument of knowledge using an abstract polynomial commitment scheme. We prove that if the commitment scheme's evaluation protocol has witness-extended emulation, then the compiled interactive argument has this knowledge property as well. If the commitment scheme is hiding and the evaluation is honest-verifier zero knowledge (HVZK), then the compiled interactive argument is HVZK as well. Finally, public-coin interactive arguments may be cryptographically compiled into SNARKs using the Fiat-Shamir heuristic.[4]

New SNARK without Trusted Setup The main practical outcome of this work is a new transparent proof system (Supersonic) for computations represented as arbitrary arithmetic circuits, obtained by cryptographically compiling the Polynomial IOPs underlying Sonic [MBKM19], PLONK [GWC19], and Marlin[CHM+19] using the DARK polynomial commitment scheme. Supersonic improves the proof size by an order of magnitude over STARKs without compromising on verification time. For one million gates, Supersonic's proofs are just 7.8KB and take around 75ms to verify. Using the notation $O_\lambda(\cdot)$ to hide multiplicative factors dependent on the security parameter λ, STARKs have size and verification complexity $O_\lambda(\log^2 T)$ whereas Supersonic has size and verification complexity $O_\lambda(\log T)$. (The additional multiplicative factors dependent on λ are actually better for Supersonic as well.) As a caveat, while the prover time in Supersonic is asymptotically on par with STARKs (*i.e.*, quasilinear in T), the concrete efficiency is much worse due to the use of heavy-weight "crypto operations" over 1200 bit class group elements in contrast to the light-weight FFTs and hash functions in STARKs. Furthermore, Supersonic is not quantum-secure due to its reliance on groups of unknown order, whereas STARKs are a candidate quantum-secure SNARK.

1.2 Related Work

Arguments based on hidden order groups Fujisaki and Okamoto [FO97] proposed homomorphic integer commitment schemes based on the RSA group. They also provide protocols to prove that a list of committed integers satisfy modular polynomial equations as opening a commitment bit by bit. Damgård and Fujisaki [DF02] patched the soundness proof of that protocol and were the first to suggest using class groups of an imaginary quadratic order as a candidate group of unknown order. Lipmaa drew the link between zero-knowledge proofs constructed from integer commitment schemes and Diophantine complexity [Lip03], coining the term *Diophantine Arguments of Knowledge*. Recently, Couteau *et al.* study protocols derived from integer commitments specifically in the RSA group to reduce the security assumptions needed; in the process they develop proofs for polynomial evaluation modulo a prime π that is not initially known to the verifier, in addition to a proof showing that an integer X lies in the range $[a, b]$ by showing that $1 + 4(X - a)(b - X)$ decomposes as the sum of 3 squares [CPP17].

Pietrzak [Pie19] developed an efficient proof of repeated squaring, *i.e.*, proving that $x^{2^T} = y$ with $O(\log T)$ proof size and verification time in order to build a conceptually simple verifiable delay function [BBBF18] based on the RSW time-lock puzzle [RSW96]. Wesolowski [Wes19] improves on this result by proposing a single-round protocol to prove correct repeated squaring in groups of unknown order with a constant size proof. Boneh *et al.* [BBF19] observe that this protocol generalizes to arbitrary exponents (PoE) and develop a proof of knowledge of an integer exponent (PoKE), as well as a zero-knowledge variant.

[4]Security for Fiat-Shamir has been proven secure in the random oracle model for constant-round protocols, for multi-round protocols satisfying *soundness against restoration attacks*, and in some cases using correlation-intractable hash functions [FS87, BCS16, KRR17, CCRR18, CCH+19].

They use both PoE and PoKE to construct efficient accumulators and vector commitment schemes.

Transparent polynomial commitments Whaby *et al.* constructed a transparent polynomial commitment scheme [WTs+18] for multilinear polynomials by combining a matrix commitment of Bootle *et al.* [BCC+16b] with the inner-product argument of Bünz *et al.* [BBB+18]. For polynomials of degree d it has commitments of size $O(\sqrt{d})$ and evaluation arguments with $O(\sqrt{d})$ communication. Zhang *et al.* [ZXZS19] and Kattis *et al.* [KPV19] recently and independently showed how to build a polynomial commitment from FRI (Fast Reed Solomon IOPP) [BBHR18, BSGKS19] The commitment is transparent, has $O(\lambda)$ size commitments and evaluation arguments with $O(\log^2 d)$ communication.

Polynomial IOP formalism In concurrent work Chiesa *et al.* [CHM+19] introduce an information theoretic framework called *algebraic holographic proofs (AHP)*. They also show that with a polynomial commitment scheme an AHP can be compiled to a preprocessing SNARK. The AHP framework is essentially equivalent to our Polynomial IOP framework. In other concurrent work, Chiesa, Ojha, and Spooner show interesting connections between algebraic holographic proofs and recursive proof composition. In the same work, the authors develop an AHP-based transparent SNARK called Fractal [COS19].

2 Technical Overview

This technical overview provides an informal description of our key technical contribution: a polynomial commitment scheme with logarithmic evaluation proofs and verification time. The commitment scheme relies on four separate tools.

1. Integer encoding of polynomials Given a univariate polynomial $f(X) \in \mathbb{Z}_p[X]$ the prover first encodes the polynomial as an integer. Interpreting the coefficients of $f(X)$ as integers in[5] $[0, p)$, define $\hat{f}(X)$ to be the *integer* polynomial with these coefficients. The prover computes $\hat{f}(q) \in \mathbb{Z}$ for some large integer $q \geq p$. This is an injective map from polynomials with bounded coefficients to integers and is also decodable: the coefficients of $f(q)$ can be recovered from the base-q expansion of $\hat{f}(q)$. For example, suppose that $f(X) = 2X^3 + 3X^2 + 4X + 1 \in \mathbb{Z}_5[X]$ and $q = 10$. Then the integer $\hat{f}(10) = 2341$ encodes the polynomial $f(X)$ because its coefficients appear in the decimal expansion of $\hat{f}(10)$.

Note that this encoding is also additively homomorphic, assuming that q is sufficiently large. For example, let $g(X) = 4X^3 + 1X^2 + 3$ such that $\hat{g}(10) = 4103$. Then $\hat{f}(10) + \hat{g}(10) = 6444 = (\hat{g} + \hat{f})(10)$. The more homomorphic operations we want to permit, the larger q needs to be. The encoding additionally permits multiplication by polynomials ($\hat{f}(q) \cdot q^k$ is equal to the encoding of $f(X) \cdot X^k$).

2. Succint integer commitments The integer $x \leftarrow \hat{f}(q)$ encoding a degree d polynomial $f(X)$ lies between q^d and q^{d+1}; in other words, its size is $(d+1)\log_2 q$ bits. The prover commits to x using a *succinct* integer commitment scheme that is additively homomorphic. Specifically, we use exponentiation in a group $\mathbb{G}$ of unknown order: the commitment is the single group element g^x for a base element $\mathsf{g} \in \mathbb{G}$ specified in the setup. (Note that if the order n of $\mathbb{G}$ is known then this is not an integer commitment; g^x could be opened to any integer $x' \equiv x \bmod n$.)

3. Evaluation protocol The evaluation protocol is an interactive argument to convince a verifier that C is an integer commitment to $\hat{f}(q)$ such that $f(z) = y$ at a provided point $z \in \mathbb{Z}_p$. The protocol must be *evaluation binding*: it should be infeasible for the prover to succeed in arguing that $f(z) = y$ and $f(z) = y'$ for $y \neq y'$. The protocol should also be an *argument of knowledge*, which informally means that any prover who succeeds at any point x must "know" the coefficients of the committed f.

As a warmup, we first describe how a prover can efficiently convince a verifier that C is a commitment to an integer polynomial of degree at most d with bounded coefficients. Assume

[5]The choice to represent the coefficients by integers in $[0, p)$ optimizes for clarity, but later on we will in fact choose a balanced set of representatives, *i.e.*, $[-\frac{p-1}{2}; \frac{p-1}{2}]$.

for now that $d = 2^k - 1$. The protocol uses a recursive divide-and-combine strategy. In each step we split $f(X)$ into two degree $d' = \lfloor \frac{d}{2} \rfloor$ polynomials $f_L(X)$ and $f_R(X)$. The left half $f_L(X)$ contains the first $d'+1$ coefficients of $f(X)$ and the right half $f_R(X)$ the second, such that $f(X) = f_L(X) + X^{d'+1} f_R(X)$. The prover now commits to f_L and f_R by computing $\mathsf{C}_L \leftarrow \mathsf{g}^{\hat{f}_L(q)}$ and $\mathsf{C}_R \leftarrow \mathsf{g}^{\hat{f}_R(q)}$. The verifier checks the consistency of these commitments by testing $\mathsf{C}_L \mathsf{C}_R^{q^{d'+1}} = \mathsf{C}$. The verifier then samples random $\alpha \in \mathbb{Z}_p$ and computes $\mathsf{C}' \leftarrow \mathsf{C}_L^{\alpha} \mathsf{C}_R$, which is an integer commitment to $\alpha \hat{f}_L(q) + \hat{f}_R(q)$. The prover and verifier recurse on the statement that C' is a commitment to a polynomial of degree at most d', thus halving the "size" of the statement. After $\log_2(d+1)$ rounds, the commitment C' exchanged between prover and verifier is a commitment to a polynomial of degree 0, *i.e.*, to a scalar $c \in \mathbb{Z}_p$. So C' is of the form $\mathsf{g}^{\hat{c}}$ where $\hat{c}$ is some integer congruent to c modulo p. The prover sends $\hat{c}$ to the verifier directly. The verifier checks that $\mathsf{g}^{\hat{c}} = \mathsf{C}'$ and also that $\hat{c} < q$.[6]

To also show that $f(z) = y$ at a provided point z, the prover additionally sends $y_L = f_L(z) \bmod p$ and $y_R = f_R(z) \bmod p$ in each round. The verifier checks consistency with the claim, *i.e.*, that $y_L + z^{d'+1} y_R = y$, and also computes $y' \leftarrow \alpha y_L + y_R \bmod p$ to proceed to the next round. (The recursive claim is that C' commits to f' such that $f'(z) = y' \bmod p$.) In the final round of recursion, the value of the constant polynomial in z is the constant itself. So in addition to testing $\mathsf{C} = \mathsf{g}^{\hat{c}}$ and $\hat{c} < q$, the verifier also checks that $\hat{c} \equiv y \bmod p$.

4. Outsourcing exponentiation for efficiency The evaluation protocol requires communicating only 2 group elements and 2 field elements per round. However, the verifier needs to check that $\mathsf{C}_L \mathsf{C}_R^{(q^{d'+1})} = \mathsf{C}$, and naïvely performing the exponentiation requires $\Omega(d \cdot \log q)$ work. To reduce this workload, we employ a recent technique for proofs of exponentiation (PoE) in groups of unknown order due to Wesolowski [Wes19] in which the prover computes this exponentiation and the verifier verifies it in essentially constant time. This outsourcing reduces the total verifier time (*i.e.*, of the entire protocol) to a quantity that is logarithmic in d.

3 Preliminaries

3.1 Assumptions

The cryptographic compilers make extensive use of groups of unknown order, *i.e.*, groups for which the order cannot be computed efficiently. Concretely, we require groups for which two specific hardness assumptions hold. First the Strong RSA Assumption [BP97] which roughly states that it is hard to take *arbitrary* roots of *random* elements. Secondly, the much newer Adaptive Root Assumption [Wes19] which is the dual of the Strong RSA Assumption and states that it is hard to take *random* roots of *arbitrary* group elements. Both of these assumptions hold in generic groups of unknown order [DK02, BBF19].

The r-strong RSA assumption as presented below is a parameterization on the Strong RSA assumption. For $r = 1$, our definition is identical to the standard Strong RSA Assumption. Higher values of r allows the adversary to take certain roots efficiently. For $r = 2$, the adversary is efficiently able to take square roots. In class groups of imaginary quadratic order taking square roots is easy. In rth order class groups taking rth roots is easy.

Assumption 1 (r-Strong RSA Assumption)**.** The *r-Strong RSA Assumption* states that an efficient adversary cannot compute ℓth roots for a given random group element, if ℓ not a power of r. Specifically, it holds for *GGen* if for any probabilistic polynomial time adversary $\mathcal{A}$:

$$\Pr\left[\mathsf{u}^{\ell} = \mathsf{g} \wedge \ell \neq r^k, k \in \mathbb{N} : \begin{array}{l} \mathbb{G} \leftarrow \mathit{GGen}(\lambda) \\ \mathsf{g} \xleftarrow{\$} \mathbb{G} \\ (\mathsf{u}, \ell) \in \mathbb{G} \times \mathbb{N} \leftarrow \mathcal{A}(\mathbb{G}, \mathsf{g}) \end{array}\right] \leq \mathsf{negl}(\lambda) \quad .$$

Assumption 2 (Adaptive Root Assumption)**.** The *Adaptive Root Assumption* holds for *GGen* if there is no efficient adversary $(\mathcal{A}_0, \mathcal{A}_1)$ that succeeds in the following task. First, $\mathcal{A}_0$ outputs an element $\mathsf{w} \in \mathbb{G}$ and some st. Then, a random prime ℓ in $\mathsf{Primes}(\lambda)$ is chosen

[6] In the full scheme, the verifier actually checks that $\hat{c} < B$ for a bound $B < q$ that depends on the field size p and the polynomial's maximum degree d

and $\mathcal{A}_1(\ell, \mathsf{st})$ outputs $\mathsf{w}^{1/\ell} \in \mathbb{G}$. For all efficient $(\mathcal{A}_0, \mathcal{A}_1)$:

$$\Pr\left[\mathsf{u}^\ell = \mathsf{w} \neq 1 \ : \ \begin{array}{l} \mathbb{G} \xleftarrow{\$} GGen(\lambda) \\ (\mathsf{w}, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_0(\mathbb{G}) \\ \ell \xleftarrow{\$} \mathsf{Primes}(\lambda) \\ \mathsf{u} \leftarrow \mathcal{A}_1(\ell, \mathsf{st}) \end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

Groups of unknown order. We consider two candidate groups of unknown order. Both have their own upsides and downsides.

RSA Group. In the multiplicative group $\mathbb{Z}_n^*$ of integers modulo a product $n = p \cdot q$ of large primes p and q, computing the order of the group is as hard as factoring n. The Adaptive Root Assumption does not hold for $\mathbb{Z}_n^*$ because $-1 \in \mathbb{Z}_n^*$ can be easily computed and has order two. This can be resolved though by working instead in the quotient group $\mathbb{Z}_n^*/\langle -1 \rangle \cong \mathrm{QR}_n$. The downside of using an RSA group, or more precisely, the group of quadratic residues modulo an RSA modulus, is that this modulus cannot be generated in a publicly verifiable way without exposing the order, and thus requires a trusted setup.

Class Group. The class group of an imaginary quadratic order is defined as the quotient group of fractional ideals by principal ideals of an order of a number field $\mathbb{Q}(\sqrt{\Delta})$, with ideal multiplication. A class group $\mathcal{C}\ell(\Delta)$ is fully defined by its discriminant Δ, which needs to satisfy only public constraints such as $\Delta \equiv 1 \bmod 4$ and $-\Delta$ must be prime. As a result, Δ can be generated from public coins, thus obviating the need for a trusted setup. A group element can be represented by two integers strictly smaller (in absolute value) than $-\Delta$, which in turn is on the same order of magnitude as RSA group elements for a similar security level. We refer the reader to Buchmann and Hamdy's survey [BH01] and Straka's accessible blog post [Str19] for more details.

Working in $\mathcal{C}\ell(\Delta)$ does present an important difficulty: there is an efficient algorithm due to Gauss to compute square roots of arbitrary elements [BS96], and by repetition, arbitrary power of two roots. As a result, such class groups cannot be used to commit to integers but rather to *dyadic rationals*, which are rational numbers whose denominator is a power of two. Additionally, the standard Strong RSA Assumption is broken if computing square roots is easy. We therefore give a weakening of the Strong RSA assumption, called 2-Strong-RSA assumption, which is believed to still hold even if computing square roots is easy. The 2-Strong-RSA assumption assumes that computing non square roots is hard.

3.2 Interactive Arguments of Knowledge

Interactive arguments are *interactive proofs* [GMR85] in which security holds only against a computationally bounded prover. In an interactive argument of knowledge for a relation $\mathcal{R}$, the prover convinces the verifier that it "knows" a witness w for a statement x such that $(x, w) \in \mathcal{R}$. In this paper, *knowledge* means that the argument has *witness-extended emulation*.

Definition 1 (Interactive Argument)**.** Let $(\mathcal{P}, \mathcal{V})$ denote a pair of PPT interactive algorithms and Setup denote a non-interactive setup algorithm that outputs public parameters pp given a security parameter. Both $\mathcal{P}$ and $\mathcal{V}$ have access to pp. Let $\langle \mathcal{P}(\mathsf{pp}, x, w), \mathcal{V}(\mathsf{pp}, x) \rangle$ denote the output of V on input x after its interaction with P, who has witness w. The triple $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ is called an argument for relation $\mathcal{R}$ if for all non-uniform PPT adversaries $\mathcal{A}$ the following properties hold:

- Perfect Completeness.

$$\Pr\left[\begin{array}{c} (x, w) \notin \mathcal{R} \text{ or} \\ \langle \mathcal{P}(\mathsf{pp}, x, w), \mathcal{V}(\mathsf{pp}, x) \rangle = 1 \end{array} \ : \ \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (x, w) \leftarrow \mathcal{A}(\mathsf{pp}) \end{array}\right] = 1$$

- Computational soundness.

$$\Pr\left[\begin{array}{c} \forall w \ (x, w) \notin \mathcal{R} \text{ and} \\ \langle \mathcal{A}(\mathsf{pp}, x, \mathsf{st}), \mathcal{V}(\mathsf{pp}, x) \rangle = 1 \end{array} \ : \ \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (x, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Definition 2 (Witness-extended emulation [GI08, Lin01])**.** Given a public-coin interactive argument tuple $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ and arbitrary prover algorithm $\mathcal{P}^*$, let $\mathsf{Record}(\mathcal{P}^*, \mathsf{pp}, x, \mathsf{st})$ denote the message transcript between $\mathcal{P}^*$ and $\mathcal{V}$ on shared input x, initial prover state st, and pp generated by Setup. Furthermore, let $\mathcal{E}^{\mathsf{Record}(\mathcal{P}^*, \mathsf{pp}, x, \mathsf{st})}$ denote an machine $\mathcal{E}$ with a transcript oracle for this interaction that can be rewound to any round and run again on fresh verifier randomness. The tuple $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ has witness-extended emulation if for every deterministic polynomial time $\mathcal{P}^*$ there exists an expected polynomial time emulator $\mathcal{E}$ such that for all non-uniform polynomial time adversaries $\mathcal{A}$ the following condition holds:

$$\Pr\left[\mathcal{A}(\mathsf{tr}) = 1 : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (x, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ \mathsf{tr} \leftarrow \mathsf{Record}(\mathcal{P}^*, \mathsf{pp}, x, \mathsf{st}) \end{array}\right] \approx$$

$$\Pr\left[\begin{array}{c} \mathcal{A}(\mathsf{tr}) = 1 \text{ and} \\ \mathsf{tr} \text{ accepting} \Rightarrow\ (x, w) \in \mathcal{R} \end{array} : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (x, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ (\mathsf{tr}, w) \leftarrow \mathcal{E}^{\mathsf{Record}(\mathcal{P}^*, \mathsf{pp}, x, \mathsf{st})}(\mathsf{pp}, x) \end{array}\right]$$

Generalized special soundness The following lemma due to Bootle *et al.* [BCC+16b] is a helpful tool for showing that an interactive argument has witness-extended emulation. It reduces the analysis to a generalized version of special soundness.

Consider a public-coin interactive argument with r rounds and verifier challenges sampled from an exponentially large message space. An $(\mathbf{n_1}, ..., \mathbf{n_r})$**-tree of accepting transcripts** for the interactive argument on input x is defined as follows. The root of the tree is labelled with the statement x. The tree has r depth. Each node at depth $i < r$ has n_i children, and each child is labelled with a distinct value for the ith challenge. An edge from a parent node to a child node is labelled with a message from prover to verifier. Every path from the root to a leaf corresponds to an accepting transcript, hence there are $\prod_{i=1}^{r} n_i$ distinct accepting transcripts overall.

Lemma 1 (Generalized Forking Lemma [BCC+16b])**.** Let $(\mathcal{P}, \mathcal{V})$ be an r-round public-coin interactive argument system for a relation $\mathcal{R}$. Let $\mathcal{T}$ be a tree-finder algorithm that, given access to a $\mathsf{Record}(\cdots)$ oracle with rewinding capability, runs in polynomial time and outputs an $(n_1, \ldots, n_r)$-tree of accepting transcripts with overwhelming probability. Let $\mathcal{X}$ be a deterministic polynomial-time extractor algorithm that, given access to $\mathcal{T}$'s output, outputs a witness w for the statement x with overwhelming probability over the coins of $\mathcal{T}$. Then $(\mathcal{P}, \mathcal{V})$ has witness-extended emulation.

We note that our statement of the Generalized Forking Lemma differs from that of Bootle *et al.*, which does not mention a tree-finder $\mathcal{T}$ and which requires $\mathcal{X}$ have success probability one. The restatement is necessary to take into account adversaries $\mathcal{P}^*$ for which the tree-finder $\mathcal{T}$ outputs a $(n_1, \ldots, n_r)$-tree of accepting transcripts with negligible but nonzero probability even when the statement x has no matching witness w. In such cases, no extractor $\mathcal{X}$ with success probability one can exist. Intuitively, the reason why this modification is okay is because the proof constructs an emulator $\mathcal{E}$ that uses $\mathcal{X}$ as a black box. Since $\mathcal{E}$ runs in polynomial time, it cannot distinguish an always-correct $\mathcal{X}$ from an overwhelmingly correct $\mathcal{X}$. If $\mathcal{X}$ should fail, then $\mathcal{E}$ re-runs $\mathcal{T}$ and $\mathcal{X}$.

Zero knowledge We recall the definition of *honest verifier zero-knowledge* (HVZK) for interactive proofs. HVZK only considers simulating the view of a verifier that follows the protocol honestly. The Fiat-Shamir transform compiles public-coin proofs that have HVZK into non-interactive proofs that have statistical zero-knowledge (for malicious verifiers).

Definition 3 (HVZK for interactive arguments)**.** Let $\mathsf{View}_{\langle \mathcal{P}(x,w), \mathcal{V}(x) \rangle}$ denote the view of the verifier in an interactive protocol described in Definition 1 on common input x and prover witness input w. The interactive protocol has δ-statistical honest verifier zero-knowledge if there exists a probabilistic polynomial time algorithm $\mathcal{S}$ such that for every $(x, w) \in \mathcal{R}$, the distribution $\mathcal{S}(x)$ is δ-close to $\mathsf{View}_{\langle \mathcal{P}(x,w), \mathcal{V}(x) \rangle}$ (as distributions over the randomness of $\mathcal{P}$ and $\mathcal{V}$).

3.3 Commitment Schemes

In defining the syntax of the various protocols, we use the following convention with respect to public values (known to both the prover and the verifier) and secret ones (known only

to the prover). In any list of arguments or returned tuple $(a, b, c; d, e)$ those variables listed before the semicolon are public, and those variables listed after it are secret. When there is no secret information, the semicolon is omitted.

Definition 4 (Commitment scheme)**.** A commitment scheme Γ is a tuple $\Gamma = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ of PPT algorithms where:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$ generates public parameters pp;
- $\mathsf{Commit}(\mathsf{pp}; x) \to (c; r)$ takes a secret message x and outputs a public commitment c and (optionally) a secret opening hint r (which might or might not be the randomness used in the computation).
- $\mathsf{Open}(\mathsf{pp}, c, x, r) \to b \in \{0, 1\}$ verifies the opening of commitment c to the message x provided with the opening hint r.

A commitment scheme Γ is **binding** if for all PPT adversaries $\mathcal{A}$:

$$\Pr\left[b_0 = b_1 \neq 0 \ \wedge \ x_0 \neq x_1 \ : \ \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (c, x_0, x_1, r_0, r_1) \leftarrow \mathcal{A}(\mathsf{pp}) \\ b_0 \leftarrow \mathsf{Open}(\mathsf{pp}, c, x_0, r_0) \\ b_1 \leftarrow \mathsf{Open}(\mathsf{pp}, c, x_1, r_1) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

We now extend the syntax to polynomial commitment schemes. The following definition generalizes that of Kate *et. al.* [KZG10] to allow interactive evaluation proofs. It also stipulates that the polynomial's degree be an argument to the protocol, contrary to Kate *et al.* where the degree is known and fixed.

Definition 5. (Polynomial commitment) A polynomial commitment scheme is a tuple of protocols $\Gamma = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$ where $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ is a binding commitment scheme for a message space $R[X]$ of polynomials over some ring R, and

- $\mathsf{Eval}(\mathsf{pp}, c, z, y, d, \mu; f(X)) \to b \in \{0, 1\}$ is an interactive public-coin protocol between a PPT prover $\mathcal{P}$ and verifier $\mathcal{V}$. Both $\mathcal{P}$ and $\mathcal{V}$ have as input a commitment c, points $z, y \in R$, and a degree d. The prover additionally knows the opening of c to a secret polynomial $f(X) \in R[X]$ with $\deg(f(X)) \leq d$. The protocol convinces the verifier that $f(z) = y$. In a multivariate extension of polynomial commitments, the input $\mu > 1$ indicates the number of variables in the committed polynomial and $z \in R^\mu$.

A polynomial commitment scheme is **correct** if an honest committer can successfully convince the verifier of any evaluation. Specifically, if the prover is honest then for all polynomials $f(X) \in R[X]$ and all points $z \in R$,

$$\Pr\left[b = 1 \ : \ \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (c; r) \leftarrow \mathsf{Commit}(\mathsf{pp}, f(X)) \\ y \leftarrow f(z) \\ d \leftarrow \deg(f(X)) \\ b \leftarrow \mathsf{Eval}(\mathsf{pp}, c, z, y, d; f(X), r) \end{array}\right] = 1 \ .$$

A polynomial commitment scheme is **evaluation binding** if no efficient adversary can convince the verifier that the committed polynomial $f(X)$ evaluates to different values $y_0 \neq y_1 \in R$ in the same point $z \in R$. However, our applications require a stronger property called *knowledge soundness*.

Knowledge soundness Any successful prover in the Eval protocol must *know* a polynomial $f(X)$ such that $f(z) = y$ and c is a commitment to $f(X)$. More formally, since Eval is a public-coin interactive argument we define this knowledge property as a special case of witness-extended emulation (Definition 2).

Define the following NP relation given $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$:

$$\mathcal{R}_{\mathsf{Eval}}(\mathsf{pp}) = \left\{ \langle (c, z, y, d), (f(X), r) \rangle : \begin{array}{l} f \in R[X] \text{ and } \deg(f(X)) \leq d \text{ and } f(z) = y \\ \text{and } \mathsf{Open}(\mathsf{pp}, c, f(X), r) = 1 \end{array} \right\}$$

The correctness definition above implies that if $\Gamma = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$ is *correct* then Eval is a correct interactive argument for $\mathcal{R}_{\mathsf{Eval}}(\mathsf{pp})$, with overwhelming probability over

the randomness of Setup. We say that Γ has **witness-extended emulation** if Eval has witness-extended emulation as an interactive argument for $\mathcal{R}_{\mathsf{Eval}}(\mathsf{pp})$.

It is easy to see that witness-extended emulation implies evaluation binding when the $\mathsf{Setup}, \mathsf{Commit}$, and Open part of Γ form a binding commitment scheme. If the adversary succeeds in Eval on both (c, z, y_0, d_0) and (c, z, y_1, d_1) for $y_0 \neq y_1$ or $d_0 \neq d_1$ then the emulator obtains two distinct witnesses $f(X) \neq f'(X)$ such that c is a valid commitment to both. This would contradict the binding property of the commitment scheme.

3.4 Proofs of Exponentiation

Wesolowski [Wes19] introduced a simple yet powerful proof of correct exponentiation ("PoE") in groups of unknown order. A prover can efficiently convince a verifier that a large exponentiation in such a group was done correctly. For instance, the prover wishes to convince the verifier that $\mathsf{w} = \mathsf{u}^x$ for known group elements $\mathsf{u}, \mathsf{w} \in \mathbb{G}$ and exponent $x \in \mathbb{Z}$, and the verifier wants to verify this with much less work than performing the exponentiation. To do this, the verifier samples a large enough prime ℓ at random and the prover provides him with $\mathsf{Q} \leftarrow \mathsf{u}^q$ where $q = \lfloor \frac{x}{\ell} \rfloor$. The verifier then simply computes the remainder $r \leftarrow (x \mod \ell)$ and checks that $\mathsf{Q}^\ell \mathsf{u}^r = \mathsf{w}$. The protocol is an argument for the relation $\mathcal{R}_{\mathsf{PoE}} = \{\langle (\mathsf{u}, \mathsf{w}, x), \varnothing \rangle : \mathsf{u}^x = \mathsf{w}\}$. The proof verification uses just $O(\lambda)$ group operations. When x is $x = q^d$ the verifier can compute $r \leftarrow x \bmod \ell$ using just $\log(d)$ ℓ-bit multiplications.

$\mathsf{PoE}(\mathsf{u}, \mathsf{w}, x)$:

1. $\mathcal{V}$ samples $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$ and sends ℓ to $\mathcal{P}$
2. $\mathcal{P}$ computes quotient q and remainder r such that $x = q\ell + r$ and $r \in \{0, \ldots, \ell - 1\}$
3. $\mathcal{P}$ computes $\mathsf{Q} \leftarrow \mathsf{u}^q$ and sends it to $\mathcal{V}$
4. $\mathcal{V}$ computes $r \leftarrow (x \mod \ell)$ and checks that $\mathsf{Q}^\ell \mathsf{u}^r = \mathsf{w}$
5. **if** check passes **then return** 1 **else return** 0

Lemma 2 (PoE soundness [Wes19])**.** PoE is an argument system for Relation $\mathcal{R}_{\mathsf{PoE}}$ with negligible soundness error, assuming the Adaptive Root Assumption (Assumption 2) holds for *GGen*.

4 Polynomial Commitments from Groups of Unknown Order

4.1 Information-Theoretic Abstraction

Before we present our concrete polynomial commitment scheme based on groups of unknown order, we present the underlying information theoretic protocol that abstracts the concrete cryptographic instantiations. The purpose of this abstraction is two-fold: first, it provides an intuitive stepping stone from which presenting and studying the concrete cryptographic protocol is easier; and second, it opens the door to alternative cryptographic instantiations that provide the same interface but based on alternative hardness assumptions.

Let $[\![*]\!] : \mathbb{Z}_p[X] \to \mathbb{S}$ be a homomorphic commitment function that sends polynomials over a prime field to elements of some set $\mathbb{S}$. Moreover, let $\mathbb{S}$ be equipped with operations $* + * : \mathbb{S} \times \mathbb{S} \to \mathbb{S}$ and $* \cdot * : \mathbb{Z}_p[X] \times \mathbb{S} \to \mathbb{S}$ that accommodate two homomorphisms for $[\![*]\!]$:

- a *linear homomorphism*: $a \cdot [\![f(X)]\!] + b \cdot [\![g(X)]\!] = [\![af(X) + bg(X)]\!]$
- a *monomial homomorphism*: $X^d \cdot [\![f(X)]\!] = [\![X^d f(X)]\!]$.

For now, assume both prover and verifier have oracle access to the function $[\![*]\!]$ and to the operations $* \cdot *$ and $* + *$. (Later on, we will instantiate this commitment function using groups of unknown order and an encoding of polynomials as integers.)

The core idea of the evaluation protocol is to reduce the statement that is being proved from one about a polynomial $f(X)$ of degree d and its evaluation $y = f(z)$, to one about a polynomial $f'(X)$ of degree $d' = \lfloor \frac{d}{2} \rfloor$ and its evaluation $y' = f'(z)$. For simplicity, assume that $d+1$ is a power of 2. The prover splits $f(X)$ into $f_L(X)$ and $f_R(X)$ such that $f(X) = f_L(X) + X^{d'+1} f_R(X)$ and such that both halves have degree at most d'. The prover obtains a random

challenge $\alpha \in \mathbb{Z}_p$ from the verifier and proceeds to prove that $f'(X) = \alpha \cdot f_L(X) + f_R(X)$ has degree d' and that $f'(z) = y' = \alpha y_L + y_R$ with $y_L = f_L(z)$ and $y_R = f_R(z)$.

The proof repeats this reduction by using $f'(X), z, y'$ and d' as the input to the next recursion step. In the final step, $f(X) = f$ is a constant and the verifier checks that $f = y$.

The commitment function binds the prover to one particular polynomial for every commitment held by the verifier. In particular, at the start of every recursion step, the verifier is in possession of a commitment $[\![f(X)]\!]$ to $f(X)$. The prover provides commitments $[\![f_L(X)]\!]$ and $[\![f_R(X)]\!]$, and the verifier checks their soundness homomorphically by testing $[\![f(X)]\!] = [\![f_L(X)]\!] + X^{d'+1} \cdot [\![f_R(X)]\!]$. From these commitments, the verifier can also compute the commitment to $f'(X)$ homomorphically, via $[\![f'(X)]\!] = \alpha \cdot [\![f_L(X)]\!] + [\![f_R(X)]\!]$. In the last step, the verifier checks that the constant polynomial f matches the commitment by computing $[\![f]\!]$ outright.

4.2 Integer Polynomial Encoding

We propose using integer commitments in a group of unknown order as a concrete instantiation of the homomorphic commitment scheme required for the abstract protocol presented in Section 4.1. At the heart of our protocol is thus an encoding of integer polynomials with bounded coefficients as integers, which also has homomorphic properties. Any commitment scheme which is homomorphic over integer polynomials is automatically homomorphic over $\mathbb{Z}_p[X]$ polynomials as well (by reducing integer polynomials modulo p). Polynomials over $\mathbb{Z}_p[X]$ can be lifted to integer polynomials in a canonical way by choosing representatives in $[0, p)$. Therefore, from here on we will focus on building a homomorphic integer encoding of integer polynomials, and how to combine this with a homomorphic integer commitment scheme.

Strawman encoding In order to encode integer polynomials over an odd prime field $\mathbb{F}_p$, we first lift them to the ring of polynomials over the integers by choosing representatives in $[0, p)$. In the technical overview (Section 2) we noted that a polynomial $f \in \mathbb{Z}[X]$ with positive coefficients bounded by q can be encoded as the integer $f(q)$. The coefficients of f can be recovered via the base q decomposition of $f(q)$. This encoding is an injective mapping from polynomials in $\mathbb{Z}[X]$ of degree at most d with positive coefficients less than q to the set $[0, q^{d+1})$. The encoding is also *partially* homomorphic. If f is encoded as $f(q)$ and g is encoded as $g(q)$ where coefficients of both g, f are less than $q/2$, then the base-q decomposition of $f(q) + g(q)$ gives back the polynomial $f + g$. By choosing a sufficiently large $q \gg p$ it is possible to perform several levels of homomorphic operations on encodings.

What goes wrong? Unfortunately, this simple encoding scheme does not quite work yet for the protocol outlined in Section 2. The homomorphic consistency checks ensure that if $[\![f_L(X)]\!]$ is a homomorphic integer commitment to the encoding of $f_L \in \mathbb{Z}[X]$, $[\![f_R(X)]\!]$ is a homomorphic integer commitment to the encoding of $f_R \in \mathbb{Z}[X]$, and both f_L, f_R are polynomials with $q/2$-bounded coefficients, then $[\![f(X)]\!]$ is an integer commitment to the encoding of $f_L + X^{d'} f_R$. (Moreover, if $f_L(z) = y_L \bmod p$ and $f_R(z) = y_R \bmod p$ then $f(z) = y_L + z^{d'} y_R \bmod p$).

However, the validity of $[\![f_L(X)]\!]$ and $[\![f_R(X)]\!]$ are never checked directly. The verifier only sees the opening of the commitment at the bottom level of recursion. If the intermediate encodings use integer polynomials with coefficients larger than $q/2$ the homomorphism is not preserved. Furthermore, even if $[\![f(X)]\!]$ is a commitment to $f^*(q)$ with positive q-bounded coefficients, an adversarial prover could find an integer polynomial g^* that does not have positive q-bounded coefficients such that $g^*(q) = f^*(q)$ and $g^* \not\equiv f^* \bmod p$ (*i.e*, g^* with coefficients greater than q or negative coefficients). The prover could then commit to $g^*_L(q)$ and $g^*_R(q)$, and recurse on $\alpha g^*_L(q) + g^*_R(q)$ instead of $\alpha f^*_L(q) + f^*_R(q)$. This would be non-binding. (For example $f^*(X) = q - 1$ and $g^*(X) = X - 1$, or $f^*(X) = q + 1$ and $g^*(X) = X + 1$).

Inferring coefficient bounds So what can the verifier infer from the opened commitment $[\![f']\!]$ at the bottom level of recursion? The opened commitment is an integer $f' = \alpha f_L + f_R$. From f', the verifier can infer a bound on the absolute value of the coefficients of the integer polynomial $f(X) = f_L + X f_R$, given that f_L and f_R were already committed in the second

to last round. The bound holds with overwhelming probability over the randomness of $\alpha \in [0, p)$. This is reasoned as follows: if $f'_0 \leftarrow \alpha_0 f_L + f_R$ and $f'_1 \leftarrow \alpha_1 f_L + f_R$ such that $\max(|f'_0|, |f'_1|) < q/(2p)$ for some distinct $\alpha_0 \neq \alpha_1$, then $|f_L| \leq |f'_1 - f'_0| < q/p$ and $|f_R| \leq |\alpha_0 f'_1 - \alpha_1 f'_0| < q/2$. If no such pair exists, *i.e.* the bound only holds for a unique α, then there is a negligibly small probability $1/p$ that f' would have passed the bound check.

What about negative coefficients? As shown above, the verifier can infer a bound on the absolute values of f_L and f_R, but still cannot infer that f_L and f_R are both *positive* integers. Moreover, if $f_R > 0$ and $f_L < 0$, then it is still possible that $f_L + q f_R > 0$, and thus that there is a distinct $g \neq f$ with q-bounded positive coefficients such that $g(q) = f(q)$. For example, say $f_R = q/2$ and $f_L = -1$ then $f_L + q f_R = q^2/2 - 1$, and $\alpha f_L + f_R = q/2 - \alpha > 0$ for every $\alpha \in [0, p)$. Yet, also $q^2/2 - 1 = g(q)$ for $g(X) = (q/2 - 1)X + q - 1$.

Ensuring injectivity How can we ensure the encoding scheme is injective over polynomials with either positive/negative coefficients bounded in absolute value? Fortunately, it is a fact that if $|f_L| < q/2$ and $|f_R| < q/2$ then at least one coefficient of g must be larger than $q/2$. In other words, if the prover had committed instead to f^*_L and f^*_R such that $g(X) = f^*_L + X f^*_R$ then the verifier could reject the opening of $\alpha \hat{f}^*_L + \hat{f}^*_R$ with overwhelming probability based on its size.

More generally, for every integer z in the range $B = (-\frac{q^{d+1}}{2}, \frac{q^{d+1}}{2})$ there is a unique degree (at most) d integer polynomial $h(X)$ with coefficients whose absolute values are bounded by $q/2$ such that $h(q) = z$. *We prove this elementary fact below and show how the coefficients of h can be recovered efficiently from z (Fact 1).* If the prover is committed to $h(q)$ at level i of the protocol, there is a unique pair of integers polynomial h_L and h_R with coefficients of absolute value bounded by $q/2$ such that $h_L(q) + q^{\frac{d+1}{2}} h_R(q) = h(q)$, and if the prover recurses on any other h^*_L and h^*_R with larger coefficients then the verifier's bound check at the bottom level of recursion will fail with overwhelming probability.

Optimization with negative coefficients As we have seen, an adversarial prover can commit to polynomials with positive or negative coefficients. As an optimization, we can actually allow the honest prover to encode polynomials using a mixture of negative and positive coefficients as well. A polynomial $f(X) \in \mathbb{Z}_p[X]$ is lifted to an integer polynomial by replacing each coefficient of f with its unique integer representative from $(-p/2, p/2)$ of the same equivalence class modulo p. Also, α can be chosen from $(-p/2, p/2)$, leading to a tighter bound on coefficient growth. This leads us to the following encoding scheme.

Final encoding scheme Let $\mathbb{Z}(b) := \{x \in \mathbb{Z} : |x| \leq b\}$ denote the set of integers with absolute value less than or equal to b. Define $\mathbb{Z}(b)[X] := \{f \in \mathbb{Z}[X] : ||f||_\infty \leq b\}$, the set of integer polynomials with coefficients from $\mathbb{Z}(b)$. (For a polynomial $g \in \mathbb{Z}[X]$ the norm $||g||_\infty$ is the maximum over the absolute values of all individual coefficients of g.)

- **Encoding.** For any integer q, the function $\mathsf{Enc} : \mathbb{Z}(b)[X] \to \mathbb{Z}$ maps $h(X) \mapsto h(q)$. A polynomial $f(X) \in \mathbb{Z}_p[X]$ is first mapped to $\mathbb{Z}(p/2)[X]$ by replacing each coefficient of f with its unique integer representative from $(-p/2, p/2)$ of the same equivalence class modulo p.

- **Decoding.** Decoding works as follows. Define the partial sum $S_k := \sum_{i=0}^{k} f_i q^i$ with $S_{-1} := 0$. Assuming $|f_i| < q/2$ for all i, observe that for any partial sum S_k we have $|S_k| < \frac{q^{k+1}}{2}$. Therefore, when $S_k < 0$ then $S_k \bmod q^{k+1} > q^{k+1}/2$ and when $S_k \geq 0$ then $S_k \bmod q^{k+1} < q^{k+1}/2$. This leads to a decoding strategy for recovering S_k from $y \in \mathbb{Z}$. The decode algorithm sets S_k to $y \bmod q^{k+1}$ if this value is less than $q^{k+1}/2$ and to $q^{k+1} - (y \bmod q^{k+1})$ otherwise. Two consecutive partial sums yield a coefficient of $f(X)$: $f_k = \frac{S_k - S_{k-1}}{q^k} \in \mathbb{Z}(b)$. These operations give rise to the following algorithm.

$\mathsf{Dec}(y \in \mathbb{Z})$:

1. **for each** k **in** $[0, \lfloor \log_q(|y|) \rfloor]$ **do:**
2. $\quad S_{k-1} \leftarrow (y \bmod q^k)$
3. $\quad$ **if** $S_{k-1} > q^k/2$ **then** $S_{k-1} \leftarrow q^k - S_{k-1}$ **end if**
4. $\quad S_k \leftarrow (y \bmod q^{k+1})$
5. $\quad$ **if** $S_k > q^{k+1}/2$ **then** $S_k \leftarrow q^{k+1} - S_k$ **end if**
6. $\quad f_k \leftarrow (S_k - S_{k-1})/q^k$
7. **return** $f(X) = \sum_{k=0}^{\lfloor \log_q(|y|) \rfloor} f_k X^k$

Fact 1. Let q be an odd integer. For any z in the range $B = (-\frac{q^{d+1}}{2}, \frac{q^{d+1}}{2})$ there is a unique degree (at most) d integer polynomial $h(X)$ in $\mathbb{Z}(\frac{q-1}{2})[X]$ such that $h(q) = z$.

Proof. Given any degree (at most) d integer polynomial $f \in \mathbb{Z}(\frac{q-1}{2})$, by construction we see that $\mathsf{Dec}(\mathsf{Enc}(f)) = f$. Therefore, Enc is an injective map from degree (at most) d polynomials in $\mathbb{Z}(\frac{q-1}{2})[X]$ to B. Furthermore, the cardinality of both the domain and range of this map is q^{d+1}. This shows that the map is surjective. In conclusion, the map is bijective. □

Encoding of dyadic rational polynomials. There exists an algorithm to compute square roots of any element of a class group of an imaginary quadratic order, originally described by Gauß (see Bosma and Stevenhagen for a modern description [BS96]). As a result, in such class groups an adversary can also commit to **dyadic rationals** $\mathbb{D} := \{\frac{x}{2^k} : x \in \mathbb{Z} \wedge k \in \mathbb{N}\} \subset \mathbb{Q}$, in addition to integers. When using class groups we therefore need to modify the encoding scheme .

The encoding map is identical, except lifted to the dyadic rationals: $\mathsf{Enc} : \mathbb{D}[X] \to \mathbb{D}$, $g(X) \mapsto g(q)$. The main difference with respect to the integer encoding scheme will be that decoding works for dyadic rationals where *both the numerator and the denominator are bounded*. Let $N \in \mathbb{N}$ be a bound on the absolute value of the numerator and $2^D \in \mathbb{N}$ be a bound on the value of the denominator, and let $\mathbb{D}(N, D) := \{\frac{x}{2^a} \in \mathbb{D} : \ |x| \leq N \wedge 2^a \leq D\}$ denote the set of such bounded dyadic rationals. The encoding scheme is uniquely decodable if $N \cdot 2^a < q/2$.

Note that denominator of $g(q)$ is bounded by $2^{\lfloor \log_2(D) \rfloor}$, D rounded down to the next power of 2. To decode such a dyadic rational, compute the integer $y \leftarrow g(q) \cdot 2^{\lfloor \log_2(D) \rfloor} \in \mathbb{Z}$ and use the decoding algorithm described above to decode a polynomial $f(X)$ in $\mathbb{Z}(q/2)[X]$. From $f(X)$ one derives the polynomial $g(X) \leftarrow \frac{f(X)}{2^{\lfloor \log_2(D) \rfloor}} \in \mathbb{D}(\lceil q/(2D) \rceil, 2^{\lfloor \log_2(D) \rfloor})[X]$ through division. If the integer polynomial encoding is uniquely decodable, then so is the scheme for dyadic rational polynomials. If q is a power of 2, then an adversary can encode Laurent polynomials, *i.e.*, polynomials where some terms have negative powers. In order to disallow negative powers, q must be odd.

4.3 Concrete Polynomial Commitment Scheme

We now instantiate the abstract homomorphic commitment function $[\![*]\!]$. To this end we sample a group of unknown order $\mathbb{G}$, and sample a random element g from this group. Lift the field polynomial $f(X) \in \mathbb{Z}_p[X]$ to an integer polynomial with bounded coefficients, *i.e.*, $\hat{f}(X) \in \mathbb{Z}(\frac{p-1}{2})[X]$ such that $\hat{f}(X) \bmod p = f(x)$. We encode $\hat{f}(X)$ as an integer by evaluating it at a "large enough" integer q. Finally we use exponentiation in $\mathbb{G}$ to commit to the integer. Therefore, $[\![f(X)]\!]$, corresponds to $\mathsf{g}^{\hat{f}(q)}$. This commitment function inherits the homomorphic properties of the integer encoding for a limited number of additions and multiplications-by-constant. The monomial homomorphism for X^d is achieved by raising the group element to the power q^d. To maintain consistency between the prover's witness polynomials and the verifier's commitments, the prover operates on polynomials with integer coefficients $\hat{f}(X), \hat{g}(X)$, *etc.*, without ever reducing them modulo p.

The Setup, Commit and Open functionalities are presented formally below. Note that the scheme is parameterized by p and q.

- $\mathsf{Setup}(1^\lambda)$: Sample $\mathbb{G} \xleftarrow{\$} GGen(\lambda)$ and $\mathsf{g} \xleftarrow{\$} \mathbb{G}$. Return $\mathsf{pp} = (\lambda, \mathbb{G}, \mathsf{g}, q)$.
- $\mathsf{Commit}(\mathsf{pp}; f(X) \in \mathbb{Z}_p[X])$: Compute $\mathsf{C} \leftarrow \mathsf{g}^{\hat{f}(q)}$ and return $(\mathsf{C}; \hat{f}(X))$.

- $\mathsf{Open}(\mathsf{pp}, \mathsf{C}, f(X), \hat{f}(X))$: Check that $\hat{f}(X) \in \mathbb{Z}(q/2)[X]$ and $\mathsf{g}^{\hat{f}(q)} = \mathsf{C}$ and $f(X) = \hat{f}(X) \bmod p$.

Evaluation protocol Using the cryptographic compilation of the information theoretic protocol we get an Eval protocol with logarithmic communication. In every round, however, the verifier needs to check consistency between $[\![f_L(X)]\!], [\![f_R(X)]\!]$ and $[\![f(X)]\!]$. This is done by checking that $\mathsf{C}_L \cdot \mathsf{C}_R^{q^{d'+1}} = \mathsf{C}$. This naive check is highly inefficient as the exponent $q^{d'+1}$ has $O(d)$ bits. To resolve this inefficiency, we utilize a proof of exponentiation (PoE) [Pie19, Wes19] to outsource the computation to the prover. The PoE protocol is an argument that a large exponentiation in a group of unknown order was performed correctly. Wesolowski's PoE [Wes19] is public coin, has constant communication and verification time, and is thus particularly well-suited here.

We now specify subtleties that were previously glossed over. First, we handle the case where $d+1$ is not a power of 2. Whenever $d+1$ is odd in the recursion, the polynomial is shifted by one degree — specifically, $f'(X) = Xf(X)$ and the protocol proceeds to prove that $f'(X)$ has degree bounded by $d' = d+1$ and evaluates to $y' = zy$ at z. The verifier obtains the matching commitment $\mathsf{C}' \leftarrow \mathsf{C}^q$.

Second, the coefficients of $f(X)$ grow by a factor of $\frac{p+1}{2}$ in every recursion step, but eventually the transmitted constant f has to be tested against some bound because if it is *too large* it should be rejected. However, the function interface provides no option to specify the allowable size of coefficients. We therefore define and use a subroutine $\mathsf{EvalBounded}$, which takes an additional argument b and which proves, in addition to what Eval proves, that all coefficients f_i of $f(X)$ satisfy $|f_i| \leq b$. Importantly, b grows by a factor for $\frac{p+1}{2}$ in every recursion step. This subroutine is also useful if commitments were homomorphically combined prior to the execution of $\mathsf{EvalBounded}$. The growth of these coefficients determines a lower bound on q: q should be *significantly* larger than b. Exactly which factor constitutes "significantly" is determined by the knowledge-soundness proof.

In the final round we check that the constant f satisfies $|f| \leq b$ and the protocol's correctness is guaranteed if $b = \frac{p-1}{2}(\frac{p+1}{2})^{\lceil \log_2(d+1) \rceil}$. However, q needs to be even larger than this value in order for extraction to work (and hence, for the proof of witness-extended emulation to go through). In RSA groups, where computing square roots is hard, we need $q > p^{2\log(d+1)+1}$; whereas in class groups where computing square roots is easy, we need $p^{3\log(d+1)+1}$. When this condition is satisfied, we can prove that the original committed polynomial has coefficients smaller than $\frac{q}{2}$. To avoid presenting two algorithms whose only difference is the one constant, we capture this constant explicitly in the variable $\varsigma_{p,d}$ and set its value depending on the context:

$$\varsigma_{p,d} = \begin{cases} p^{\log_2(d+1)} & \text{(in RSA groups)} \\ p^{2\log_2(d+1)} & \text{(in class groups)} \end{cases} .$$

We now present the full, formal Eval protocol below.

$\mathsf{Eval}(\mathsf{pp}, \mathsf{C} \in \mathbb{G}, z \in \mathbb{Z}_p, y \in \mathbb{Z}_p, d \in \mathbb{N}; \tilde{f}(X) \in \mathbb{Z}_p[X]) :$ // $\tilde{f}(X) = \sum_{i=0}^{d} \tilde{f}_i X^i$

1. $\mathcal{P}$ computes $f_i \in [-\frac{p-1}{2}, \frac{p-1}{2}]$ such that $f_i \equiv \tilde{f}_i \bmod p$ for all $i \in [0, d]$.
2. $\mathcal{P}$ computes $f(X) \leftarrow \sum_{i=0}^{d} f_i \cdot X^i \in \mathbb{Z}(\frac{p-1}{2})[X] \subset \mathbb{Z}[X]$
3. $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{EvalBounded}(\mathsf{pp}, \mathsf{C}, z, y, d, \frac{p-1}{2}; f(X))$

$\mathsf{EvalBounded}(\mathsf{pp}, \mathsf{C} \in \mathbb{G}, z \in \mathbb{Z}_p, y \in \mathbb{Z}_p, d \in \mathbb{N}, b \in \mathbb{Z}; f(X) \in \mathbb{Z}(b)[X])$

1. **if** $d = 0$:
2. $\quad\mathcal{P}$ sends $f(X) \in \mathbb{Z}$ to the verifier. // $f = f(X)$ is a constant
3. $\quad\mathcal{V}$ checks that $b \cdot \varsigma_{p,d} < q$// $\varsigma_{p,d} = O(p^{2\log(d)})$ (see Theorem 1 and 2)
4. $\quad\mathcal{V}$ checks that $|f| \leq b$
5. $\quad\mathcal{V}$ checks that $f \equiv y \bmod p$
6. $\quad\mathcal{V}$ checks that $\mathsf{g}^f = \mathsf{C}$
7. $\quad\mathcal{V}$ outputs 1 **if** all checks pass, 0 otherwise.
8. **if** $d + 1$ is odd
9. $\quad d' \leftarrow d + 1, \mathsf{C}' \leftarrow \mathsf{C}^q$, $y' \leftarrow y \cdot z \bmod p$ and $f'(X) \leftarrow X \cdot f(X)$.
10. $\quad\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{EvalBounded}(\mathsf{pp}, \mathsf{C}', z, y', d', b; f'(X))$
11. **else** : // $d \geq 1$ and $d + 1$ is even
12. $\quad\mathcal{P}$ and $\mathcal{V}$ compute $d' \leftarrow \frac{d+1}{2} - 1$
13. $\quad\mathcal{P}$ computes $f_L(X) \leftarrow \sum_{i=0}^{d'} f_i \cdot X^i$ and $f_R(X) \leftarrow \sum_{i=0}^{d'} f_{d'+1+i} \cdot X^i$
14. $\quad\mathcal{P}$ computes $y_L \leftarrow f_L(z) \bmod p$ and $y_R \leftarrow f_R(z) \bmod p$
15. $\quad\mathcal{P}$ computes $\mathsf{C}_L \leftarrow \mathsf{g}^{f_L(q)}$ and $\mathsf{C}_R \leftarrow \mathsf{g}^{f_R(q)}$
16. $\quad\mathcal{P}$ sends $y_L, y_R, \mathsf{C}_L, \mathsf{C}_R$ to $\mathcal{V}$. // See Section 4.5 for an optimization
17. $\quad\mathcal{V}$ checks that $y = y_L + z^{d'+1} \cdot y_R \bmod p$, outputs 0 if check fails.
18. $\quad\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{PoE}(\mathsf{C}_R, \mathsf{C}/\mathsf{C}_L, q^{d'+1})$// Showing that $\mathsf{C}_L \mathsf{C}_R^{(q^{d'+1})} = \mathsf{C}$
19. $\quad\mathcal{V}$ samples $\alpha \xleftarrow{\$} [-\frac{p-1}{2}, \frac{p-1}{2}]$ and sends it to $\mathcal{P}$
20. $\quad\mathcal{P}$ and $\mathcal{V}$ compute $y' \leftarrow \alpha y_L + y_R \bmod p$, $\mathsf{C}' \leftarrow \mathsf{C}_L^{\alpha} \mathsf{C}_R$, $b' \leftarrow b\frac{p+1}{2}$.
21. $\quad\mathcal{P}$ computes $f'(X) \leftarrow \alpha \cdot f_L(X) + f_R(X) \in \mathbb{Z}[X]$ // $\deg(f'(X)) = d'$
22. $\quad\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{EvalBounded}(\mathsf{pp}, \mathsf{C}', z, y', d', b'; f'(X))$

4.4 Security Analysis

Lemma 3. The polynomial commitment scheme is binding for polynomials in $\mathbb{Z}(b)[X]$ for $b < q/2$ if either the Adaptive Root Assumption or the Strong RSA Assumption hold.

Lemma 4. The polynomial commitment scheme is correct for polynomials in $\mathbb{Z}_p[X]$ of degree at most d if $q > p^{\lceil \log_2(d+1) \rceil + 1}$.

The proofs of the previous lemmas are in Appendix A.1 and A.2. Next is the main security theorem, which states that the evaluation protocol has witness-extended emulation. We start with a high-level intuitive overview where we also identify potential obstacles.

Proof idea. The goal is to construct an extractor by recursively computing $f(X)$ from $f'(X)$. In the final round the verifier receives f such that $|f| \leq b$, and therefore the extractor possesses this constant polynomial as well. Working backwards from here, the extractor uses rewinding in every step to find $f_L(X)$ and $f_R(X)$ and thereby finds $f(X) = f_L(X) + X^{d'+1} f_R(X)$. Specifically, in each round the extractor has $f'(X) = \alpha f_L(X) + f_R(X)$. Suppose the extractor also possesses $f''(X) = \alpha' f_L(X) + f_R(X)$. From $f'(X)$, $f''(X)$, α and α' it is easy to compute $f_L(X)$ and $f_R(X)$. The extractor then computes $f(X) = f_L(X) + X^{d'+1} f_R(X)$. A careful analysis shows that if the coefficients of $f'(X)$ are bounded by b then $f_L(X)$ and $f_R(X)$ must have coefficients bounded by $b \cdot p$ in absolute value. Using a similar analysis we can show that $f(z) \bmod p = y$ for the extracted polynomial $f(X)$.

This argument shows that there is an extractor algorithm $\mathcal{X}$ capable of extracting the witness $f(X)$ from a binary tree of accepting transcripts. Moreover, a tree-finding algorithm $\mathcal{T}$ can output such a tree by repeatedly rewinding the prover, running it with fresh verifier randomness each time, and recording the resulting transcripts. As a result, the Generalized Forking Lemma (Lemma 1) applies and establishes that the protocol has witness-extended emulation.

The full proof takes into account the cryptographic compilation of the protocol using the integer encoding and the commitment scheme based on groups of unknown order. Additionally the full proof will need to support dyadic rationals because taking square roots is easy in class groups.

Theorem 1. The polynomial commitment scheme for polynomials in $\mathbb{Z}_p[X]$ of degree at most $d = \mathsf{poly}(\lambda)$, instantiated using $q > p^{2\lceil \log_2(d+1)\rceil+1}$ and *GGen*, has witness extended emulation (Definition 2) if the Adaptive Root Assumption and the Strong RSA Assumption hold for *GGen*.

Theorem 2. Let *GGen* generate groups $\mathbb{G}$ of unknown order such that the order of $\mathbb{G}$ is odd, and such there exists a PPT algorithm for taking square roots in $\mathbb{G}$. The polynomial commitment scheme for polynomials in $\mathbb{Z}_p[X]$ of degree at most $d = \mathsf{poly}(\lambda)$, instantiated using $q > p^{3\lceil \log_2(d+1)\rceil+1}$ and *GGen*, has witness extended emulation (Definition 2) if the Adaptive Root Assumption and the 2-Strong RSA Assumption hold for *GGen*.

The proof of Theorem 2 is nearly identical to the proof of Theorem 1 but the extracted polynomials are polynomials over the dyadic rationals and not over the integers. This requires the bound on q to be larger by a factor of $p^{\log(d+1)}$. Both proofs are presented in the Appendix (A.3 and A.4).

4.5 Optimizations

We present several ideas for optimizing the performance of the Eval protocol.

Precomputation. The prover has to compute powers of g as large as q^d. While this can be done in linear time, this expense can be shifted to a preprocessing phase in which all elements $\mathsf{g}^{q^i}, i \in \{1, \ldots, d_{max}\}$ are computed. Since for coefficient $|f_i| \leq -\frac{p-1}{2}$ this allows the computation of $\mathsf{g}^{f(q)}$ in $O(\lambda d)$ group operations as opposed to $O(\lambda d \log(d))$. In addition to reducing the prover's workload, this optimization enables parallelizing it. The computation of the PoE proofs can simiarly be parallelized. The prover can express each Q as a power of g which enables pre-computation of powers of g and parallelism as described by Boneh *et al.* [BBF19].

The pre-computation also enables the use of multi-exponentiation techniques [Pip80]. Boneh *et al.* [BBF19] and Wesolowski [Wes19] showed how to use these techniques to reduce the complexity of the PoE prover. The largest PoE exponent $q^{\frac{d+1}{2}}$ has $O(\lambda d \log(d))$ bits. Multi-exponentiation can therefore reduce the prover work to $O(\lambda d)$ instead of $O(\lambda d \log(d))$.

Two group elements per round. In each round the verifier has a value C and receives C_L and C_R such that $\mathsf{C}_L \mathsf{C}_R^{q^{d'+1}} = \mathsf{C}$. This is redundant. It suffices that the verifier sends C_R. The verifier could now compute $\mathsf{C}_L \leftarrow \mathsf{C} \cdot \mathsf{C}_R^{-q^{d'+1}}$, but this is expensive as it involves an exponentiation by q^d. Instead, the verifier infers $\mathsf{C}_R^{(q^{d'+1})}$ from the PoE: the prover's message is Q and the verifier can directly compute $\mathsf{C}_R^{q^{d'+1}} \leftarrow \mathsf{Q}^\ell \mathsf{C}_R^r$ for a challenge ℓ and $r \leftarrow q^{d'+1} \bmod \ell$. From this the verifier infers $\mathsf{C}_L \leftarrow \mathsf{C}/\mathsf{C}_R^{q^{d'+1}}$. The security of PoE does not require that $\mathsf{C}_R^{q^{d'+1}}$ be sent before the challenge ℓ as it is uniquely defined by C_R and $q^{d'+1}$. The same optimization can be applied to the non-interactive variant of the protocol.

Similarly the verifier can infer y_L as $y_L \leftarrow y - z^{d'+1} y_R$. This reduces the communication to two group elements per round and 1 field element. Additionally the prover sends f which has roughly the size of $\log(d+1)$ field elements, which increases the total communication to roughly $2\log(d)$ elements in $\mathbb{G}$ and $2\log(d)$ elements in $\mathbb{Z}_p$.

Evaluation at multiple points The protocol and the security proof extend naturally to the evaluation in a vector of points $\boldsymbol{z}$ resulting in a vector of values $\boldsymbol{y}$, where both are members of $\mathbb{Z}_p^k$. The prover still sends $\mathsf{C}_L \in \mathbb{G}$ and $\mathsf{C}_R \in \mathbb{G}$ in each round and additionally $\boldsymbol{y}_L, \boldsymbol{y}_R \in \mathbb{Z}_p^k$. In the final round the prover only sends a single integer f such that $\mathsf{g}^f = \mathsf{C}$ and $f \bmod p = y$.

This is significantly more efficient than independent executions of the protocol as the encoding of group elements is usually much larger than the encoding of elements in $\mathbb{Z}_p$. Using the optimization above, the marginal cost with respect to k of the protocol is a single

element in $\mathbb{Z}_p$. If $\lambda = \lceil \log_2(p) \rceil$ is 120, then this means evaluating the polynomial at an additional point increases the proof size by only $15 \log(d+1)$ bytes.

Joining Evals. In many applications such as compiling polynomial IOPs to SNARKs (see Section 5) multiple polynomial commitments need to be evaluated at the same point z. This can be done efficiently by taking a random linear combination of the polynomials and evaluating that combination at z. The prover simply sends the evaluations of the individual polynomials and then a single evaluation proof for the combined polynomials. The communication cost for evaluating m polynomials at 1 point is still linear in m but only because the evaluation of each polynomial at the point is being sent. The size of the eval proof, however, is independent of m. Taking a random linear combination does increase the bound on q slightly, as shown in Theorem 3 which is presented below.

$$\mathcal{R}_{\mathsf{JE}}(\mathsf{pp}) = \left\{ \langle (\mathsf{C}_1, \mathsf{C}_2, z, y_1, y_2, d), (f_1(X), f_2(X)) \rangle : \begin{array}{l} \mathsf{C}_1, \mathsf{C}_2 \in \mathbb{G} \\ z, y_1, y_2 \in \mathbb{Z}_p \\ f_1(X), f_2(X) \in \mathbb{Z}(b) \\ (\mathsf{C}_1, z, y_1, d) \in \mathcal{R}_{\mathsf{Eval}}(\mathsf{pp}) \\ (\mathsf{C}_2, z, y_2, d) \in \mathcal{R}_{\mathsf{Eval}}(\mathsf{pp}) \end{array} \right\}$$

$\mathsf{JoinedEval}(\mathsf{pp}, \mathsf{C}_1, \mathsf{C}_2, z, y_1, y_2, d; f_1(X), f_2(X)) :$ // $f_1(X), f_2(X) \in \mathbb{Z}(\frac{p-1}{2})[X]$
Statement: $(\mathsf{pp}, \mathsf{C}_1, \mathsf{C}_2, z, y_1, y_2, b, d) \in \mathcal{R}_{\mathsf{JE}}$

1. $\mathcal{V}$ samples $\alpha \xleftarrow{\$} [-\frac{p-1}{2}, \frac{p-1}{2}]$ and sends it to $\mathcal{P}$
2. $\mathcal{P}$ and $\mathcal{V}$ compute $\mathsf{C}' \leftarrow \mathsf{C}_1^{\alpha} \mathsf{C}_2$ and $y' \leftarrow \alpha \cdot y_1 + y_2 \bmod p$
3. $\mathcal{P}$ computes $f'(X) \leftarrow \alpha f_1(X) + f_2(X)$
4. $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{EvalBounded}(\mathsf{pp}, \mathsf{C}', z, y', d, \frac{p^2-1}{4}; f(X))$

Theorem 3. The protocol $\mathsf{JoinedEval}$ is an interactive argument for the relation $\mathcal{R}_{\mathsf{JE}}$ and has perfect completeness and witness extended emulation if the Strong RSA and Order Assumption hold for *GGen* and if $q > (p-1)(\frac{p^2-1}{2})^{\lceil \log_2(d+1) \rceil + 1}$ (e.g., $q > p^{2\log_2(d+1)+3}$).

The proof is presented in Appendix A.5.

We can additionally combine this optimization with the previous optimization of evaluating a single polynomial at different points. This allows us to evaluate m polynomials at k points with very little overhead. The prover groups the polynomials by evaluation points and first takes linear combinations of the polynomials with the same evaluation point and computes y_1 to y_k using the same linear combinations. Then it takes another combination of the joined polynomials. In each round of the Eval protocol the prover sends $y_{L,1}$ through $y_{L,k}$, i.e. one field element per evaluation point and computes $y_{R,1}$ through $y_{R,k}$. In the final step the prover sends f and the verifier can check whether the final y values are all equal to $f \bmod p$. This enables an Eval proof of m, degree d polynomials at k points using only $2\log_2(d+1)$ group elements and $(1+k)\log_2(d+1)$ field elements.

Evaluating the polynomial over multiple fields The polynomial commitment scheme is highly flexible. For example it does not specify a prime field $\mathbb{Z}_p$ or a degree d in the setup. It instead commits to an integer polynomial with bounded coefficients. That integer polynomial can be evaluated modulo arbitrary primes which are exponential in the security parameter λ as the soundness error is proportional to its inverse. Note that q also needs large enough such that the scheme is secure for the given prime p and degree d (see Theorem 1). The second condition, however, can be relaxed. A careful analysis shows that the challenges α just need to be sampled from an exponential space, *e.g.*, $[-2^{\lambda}, 2^{\lambda}]$. So as long as $q > p \cdot 2^{\lambda \cdot 2\lceil \log_2(d+1) \rceil}$ for RSA groups or $q > p \cdot 2^{\lambda \cdot 3\lceil \log_2(d+1) \rceil}$ for class groups one can evaluate degree d polynomial with coefficients bounded by 2^{λ} over any prime field.

Additionally the proof elements $\mathsf{C}_L, \mathsf{C}_R \in \mathbb{G}$ are independent from the field over which the polynomial is evaluated. This means that it is possible to evaluate a committed polynomial $f(X) \in \mathbb{Z}(b)$ over two separate fields $\mathbb{Z}_p$ and $\mathbb{Z}_{p'}$ in parallel using only $2\log(d+1)$ group elements.

4.6 Multivariate Commitment Scheme

We can extend our polynomial commitment scheme to multivariate polynomials. The idea is simply to use higher degrees of q to encode the next indeterminate. The protocol is linear in the number of variables and logarithmic in the total degree of the polynomial. For simplicity we only present a protocol for μ-variate polynomials where the degree in each variable is d. The protocol extends naturally to different degrees per variable.

Encoding Let $q_i = q^{(d+1)^i}$ then $\hat{f}(q_1, \ldots, q_\mu) \in \mathbb{Z}$ is an encoding of the multivariate polynomial $f(X_1, \ldots, X_\mu)$ with maximum degree d. We use $\mathsf{Dec}_{Multi}(f(q), \mu, d)$ to denote the decoding of an μ-variate polynomial with degree exactly d in each variable. The decoding algorithm simply uses the univariate decoding algorithm described in Section 4.2 to decode a univariate polynomial $\hat{h}(X)$ of degree $(d+1)^\mu - 1$. Then it associates each monomial of the univariate polynomial with a degree vector $(d_1, \ldots, d_\mu)$ of the multivariate polynomial. The coefficient of the ith monomial becomes the coefficient of the $(d_1, \ldots, d_\mu)$-monomial, where $(d_1, \ldots, d_\mu)$ is the base-$(d+1)$ decomposition of i.

Protocols Using this encoding we can naturally derive the multivariate commitment scheme and Eval protocol. The Eval protocol computes the univariate polynomials $f(q_1, \ldots, q_{\mu-1}, X_\mu)$ and then uses the univariate eval protocol to reduce the claim from a claim about an μ-variate polynomial to one about an $(\mu-1)$-variate one. At the final step the prover opens the now constant polynomial and the verifier can check the claim. For example, the protocol would reduce a bivariate (say X and Y) cubic polynomial to a univariate one (in Y) in two rounds of interaction and then reduce the degree of Y using another two rounds.

$\mathsf{MultiSetup}(1^\lambda)$:

1. $\mathbb{G} \xleftarrow{\$} GGen(\lambda)$
2. $\mathsf{g} \xleftarrow{\$} \mathbb{G}$
3. **return** $\mathsf{pp} = (\lambda, \mathbb{G}, \mathsf{g})$

$\mathsf{MultiCommit}(\mathsf{pp}; f(X_1, \ldots, X_\mu) \in \mathbb{Z}(\frac{p-1}{2})[X_1, \ldots, X_\mu] \subset \mathbb{Z}[X_1, \ldots, X_\mu])$:

1. $d \leftarrow \deg(f)$ // For simplicity assume $f(X_1, \ldots, X_n)$ has degree d in each variable
2. $q_i \leftarrow q^{(d+1)^{i-1}}$ for each $i \in [\mu]$
3. $\mathsf{C} \leftarrow \mathsf{g}^{f(q_1, \ldots, q_\mu)}$
4. **return** $(\mathsf{C}; f(X_1, \ldots, X_\mu))$

$\mathsf{MultiEval}(\mathsf{pp}, \mathsf{C} \in \mathbb{G}, \boldsymbol{z} \in \mathbb{Z}_p^\mu, y \in \mathbb{Z}_p, d, \mu, b \in \mathbb{N}; f(X_1, \ldots, X_\mu) \in \mathbb{Z}(b)[X_1, \ldots, X_\mu])$:

1. **if** $\mu = 1$
2. $\quad \mathcal{P}$ and $\mathcal{V}$ run $\mathsf{EvalBounded}(\mathsf{pp}, \mathsf{C}, z_1, y, d, b, x; f(X_1))$
3. **else**
4. $\quad$ Let $\hat{f}(X_\mu) \leftarrow f(q_1, \ldots, q_{\mu-1}, X_\mu)$
5. $\quad$ Let $\mathsf{pp}_\mu \leftarrow \{\lambda, \mathbb{G}, \mathsf{g}, p, q_\mu\}$
6. $\quad \mathcal{P}$ and $\mathcal{V}$ run the univariate $\mathsf{EvalBounded}(\mathsf{pp}_\mu, \mathsf{C}, z_\mu, y, d, q_\mu; \hat{f}(X))$
7. $\quad\quad$ **except:** when $d = 0$, f is not sent; instead the protocol returns its input at this point, *i.e.*, (C', y', b') along with the prover's witness $f'(X_1, \ldots, X_{\mu-1}) = \mathsf{Dec}_{Multi}(f, \mu - 1, d)$ (Lines 2-6 of $\mathsf{EvalBounded}$).
8. $\quad \boldsymbol{z}' \leftarrow (z_1, \ldots, z_{\mu-1}) \in \mathbb{Z}_p^{\mu-1}$
9. $\quad \mathcal{P}$ and $\mathcal{V}$ run $\mathsf{MultiEval}(\mathsf{pp}, C', \boldsymbol{z}', y', d, \mu - 1, b'; f')$

We only prove security under the strong RSA assumption. The security proof, however, directly extends to groups where taking square roots is easy under the 2-Strong-RSA Assumption. In that case $q > p^{3\mu \log_2(d+1)+1}$ suffices.

Theorem 4 (Multivariate Eval)**.** The polynomial commitment scheme for multi-variate polynomials consisting of protocols $(\mathsf{MultiSetup}, \mathsf{MultiCommit}, \mathsf{MultiEval})$ has perfect correctness and witness extended emulation if the Adaptive Root Assumption and the Strong RSA Assumption hold for $GGen$ for μ-variate polynomials of degree d and if $d^\mu = \mathsf{poly}(\lambda)$ if $q > p^{2\mu \log_2(d+1)+1}$.

Proof. Perfect correctness follows from the correctness of the univariate commitment scheme and the fact that the coefficients of the witness polynomial in the honest execution are less than $\frac{p-1}{2}p^{\mu\lceil\log(d+1)\rceil} < q/2$.

To show witness extended emulation we use the forking lemma (Lemma 1) and build a polynomial time extractor algorithm $\mathcal{X}_{\mathsf{MultiEval}}$ that given a binary tree of transcripts of depth $\mu \cdot \lceil\log(d+1)\rceil$, extracts a witness. Each node corresponds to a different challenge α as described in the forking lemma. The tree consists of at most $(d+1)^\mu = \mathsf{poly}(\lambda)$ transcripts. Lemma 2 states that the probability that an adversary can create any accepting transcript for which the PoE can't be replaced by a direct check is negligible under the Adaptive Root Assumption. We can therefore invoke the lemma to replace all PoE executions with direct verification checks that $\mathsf{C}_L\mathsf{C}_R^{q^{d'}+1} = \mathsf{C}$.

In constructing $\mathcal{X}_{\mathsf{MultiEval}}$ we use the extractor $\mathcal{X}_{\mathsf{Eval'}}$ described in the proof of Theorem 1. $\mathcal{X}_{\mathsf{Eval'}}$ computes, given a tree of transcripts for Eval' a valid witness of Eval' or a fractional root of g or an element of known order in $\mathbb{G}$. We construct $\mathcal{X}_{\mathsf{MultiEval}}$ recursively invoking $\mathcal{X}_{\mathsf{Eval'}}$ once per degree μ. The probability that a polynomial time adversary and a polynomial time extractor $\mathcal{X}_{\mathsf{Eval'}}$ can produce a fractional root or an element of known order in $\mathbb{G}$ is negligible under the strong-RSA and the the adaptive root assumptions. From hence on we will consider the case where neither of these events happen.

We use the superscript (i) to denote the inputs to MultiEval where $\mu = i$. If $\mu = 1$ then the extractor $\mathcal{X}_{\mathsf{Eval'}}$ directly extracts $f^{(1)}(X) \in \mathbb{Z}(b)$, a univariate degree d polynomial with coefficients bounded by $b = \frac{p-1}{2}p^{2\lceil\log_2(d+1)\rceil}$ and such that $f(z) = y \bmod p$. Note that $q/2 > b$ so the extraction succeeds.

For $\mu > 1$, let's assume that $f^{(\mu-1)}(X_1, \ldots, X_{\mu-1}) \in \mathbb{Z}(b)$ is an extracted $\mu - 1$ variate polynomial with degree d in each variable such that $f^{(\mu-1)}(z_1, \ldots, z_{\mu-1}) \bmod p = y'$. Let $f' \leftarrow \mathsf{Enc}_{Multi}(f^{(\mu-1)}(X_1, \ldots, X_{\mu-1}) \in \mathbb{Z}$ be the encoding of $f^{(\mu-1)}(X_1, \ldots, X_n)$, such that $\mathsf{C}^{(\mu-1)} \leftarrow \mathsf{g}^{f'}$. Note that f' is equivalent to an encoding of a univariate degree $(d+1)^{\mu-1}$ polynomial with the same coefficients as the multivariate polynomial. Let $g^{(\mu-1)}(X) = \mathsf{Dec}(f') \in \mathbb{Z}(b)[X]$ be that polynomial. Using $g^{(\mu-1)}(X)$ as the witness the extractor $\mathcal{X}_{\mathsf{Eval'}}$ extracts a univariate degree $(d+1)^\mu$ polynomial $g^{(\mu)}(X)$ with coefficients in $\mathbb{Z}(b \cdot p^{\lceil\log(d+1)\rceil})$. Let $f'' \leftarrow g^{(\mu)}(q)$ be the encoding of $g^{(\mu)}$ such that $\mathsf{C}^{(\mu)} = \mathsf{g}^{f''}$. Note that using the multivariate decoding algorithm f'' also encodes a μ-variate degree d polynomial, i.e. $f^{(\mu)}(X_1, \ldots, X_\mu) \leftarrow \mathsf{Dec}_{Multi}(f'', \mu, d)$. The coefficient on X_i in $g^{(\mu)}(X)$ is the coefficient of the monomial in $f^{(\mu)}$ with degree vector defined by the base-$(d+1)$ decomposition of i, i.e. $\prod_{j=1}^{\mu} X_j^{\lfloor i/(d+1)^{j-1}\rfloor \bmod d+1}$. Note that the extraction additionally guarantees that the polynomial evaluation is correct, i.e. $f(z_1, \ldots, z_\mu) \bmod p = y$.

The final extracted polynomial has coefficients in $\mathbb{Z}(\frac{p-1}{2}p^{2\mu\lceil\log_2(d+1)\rceil})$. Since $q > p^{2\mu\lceil\log_2(d+1)\rceil+1}$ both the univariate and the multivariate decoding succeed and the extractor extracts a valid μ-variate degree d witness polynomial. □

4.7 Hiding Commitments and Zero-Knowledge Evaluation

Many applications, such as the construction of ZK-SNARKs, require a polynomial commitment scheme where an evaluation leaks no information about the committed polynomial beyond its value at the queried point. To provide this we show how to build a hiding polynomial commitment along with a zero-knowledge evaluation protocol.

We start by defining what it means for a polynomial commitment scheme to be *hiding*:

Definition 6. A commitment scheme $\Gamma = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ is **hiding** if for all probabilistic polynomial time adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, the probability of distinguishing between commitments of different messages is negligible:

$$\left| 1 - 2 \cdot \Pr\left[\hat{b} = b \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ m_0, m_1, \mathsf{st} \leftarrow \mathcal{A}_0(\mathsf{pp}) \\ b \xleftarrow{\$} \{0,1\} \\ (c; r) \leftarrow \mathsf{Commit}(\mathsf{pp}, m_b) \\ \hat{b} \leftarrow \mathcal{A}_1(\mathsf{st}, c) \end{array} \right] \right| \leq \mathsf{negl}(\lambda) \quad .$$

If the property holds for all algorithms then we say that the commitment is *statistically* hiding.

Hiding Polynomial Commitment We make the polynomial commitment described in Section 4 hiding by adding a degree $d+1$ term with a large random coefficient. Let $B \geq |\mathbb{G}|$ be a publicly known upper bound on the order of $\mathbb{G}$. We will choose the blinding coefficient between 0 and $B \cdot 2^\lambda$. Formally, the hiding commitment algorithm is described as follows:

- $\mathsf{CommitH}(f(X) \in \mathbb{Z}_p[X]) \to (\mathsf{C}; \hat{f}(X), d, r)$. Lift $f(X) \in \mathbb{Z}_p[X]$ to $\hat{f}(X) \in \mathbb{Z}(\frac{p-1}{2})[X]$ and select random integer $r \xleftarrow{\$} [0, B \cdot 2^\lambda)$. Compute $d \leftarrow \deg(f(X))$ and $\mathsf{C} \leftarrow \mathsf{g}^{\hat{f}(q)+q^{d+1} \cdot r}$ and return commitment C with secret opening information $\hat{f}(X), d, r$.

Let $\mathsf{h} \leftarrow \mathsf{g}^{q^{d+1}}$. To argue that C is hiding, it suffices to show that C is computationally indistinguishable from a random element of $\langle \mathsf{g} \rangle$, the cyclic group generated by g. In a setting with trusted-setup, in which the trusted party has a trapdoor to compute the order of g (e.g. RSA groups), the trusted party can select q such that g and h generate the same subgroup. In this case, h^r for $r \xleftarrow{\$} [0, B \cdot 2^\lambda)$ has statistical distance at most $2^{-\lambda}$ from uniform in $\langle \mathsf{g} \rangle$, so long as $B \geq |\langle \mathsf{g} \rangle|$.

Unfortunately, in a setting without a trusted setup, h might only generate a subgroup of $\langle \mathsf{g} \rangle$. The commitment then becomes computationally hiding under a *Subgroup Indistinguishability Assumption*[7] [BG10]: our precise assumption is that no efficient adversary can distinguish a random element of $\langle \mathsf{h} \rangle$ from $\langle \mathsf{g} \rangle$ for any non-trivial $\mathsf{h} \in \langle \mathsf{g} \rangle$. For simplicity, Theorem 5 assumes that g and h generate the same group.

If the condition $\langle \mathsf{g} \rangle = \langle \mathsf{g}^{q^{d+1}} \rangle$ cannot be guaranteed then the security proof must be modified to show that an adversary that can efficiently distinguish commitments with non-negligible probability must be able to distinguish between random elements sampled from $\langle \mathsf{g} \rangle$ and random elements sampled from $\langle \mathsf{g}^{q^{d+1}} \rangle$. The informal proof sketch is as follows. Suppose there exists a non-uniform distinguisher $D_{f,g}$ that can distinguish freshly generated commitments to f and g with non-negligible probability, then the non-uniform adversary $\mathcal{A}_{f,g}$ may be constructed as follows: upon receipt of g' sampled either from $\langle \mathsf{g} \rangle$ or from $\langle \mathsf{g}^{q^{d+1}} \rangle$, it sends $\mathsf{g}^{f(q)}\mathsf{g}'$ and $\mathsf{g}^{g(q)}\mathsf{g}'$ to the distinguisher $D_{f,g}$. In the case that g' was sampled from $\langle \mathsf{g}^{q^{d+1}} \rangle$ this is a statistical simulation of a pair of commitments to f and g respectively, hence the distinguisher should succeed with non-negligible probability. In the case that g was sampled from $\langle \mathsf{g} \rangle$, the pair is actually statistically indistinguishable, and thus the distinguisher must fail. Thus, $\mathcal{A}_{f,g}$ is able to distinguish from which group g' was sampled with non-negligible probability, contradicting the subgroup indistinguishability assumption.

Theorem 5. The commitment scheme $\Gamma = (\mathsf{Setup}, \mathsf{CommitH}, \mathsf{Open})$ is statistically hiding if $B \gg |\mathbb{G}|$ and if $\langle \mathsf{g} \rangle = \langle \mathsf{g}^{q^{d+1}} \rangle$; and it is binding if the commitment described in Section 4.3 is binding.

Proof. The hiding commitment is a commitment to a degree $d+1$ polynomial. It therefore directly inherits the binding property from the non-hiding scheme.

To show hiding, we use the fact that the uniform distributions $[0, b]$ and $[a, a+b]$ have statistical distance $\frac{a}{b}$, *i.e.*, the probability that any algorithm can distinguish the distributions from a single sample is less than $\frac{a}{b}$. Similarly $\mathsf{C} \leftarrow \mathsf{g}^{f(q)+rq^{d+1}}$ for $r \xleftarrow{\$} [0, B \cdot 2^\lambda)$ has statistical distance at most $2^{-\lambda}$ from a uniform element generated by g if $B \geq |\langle \mathsf{g} \rangle|$. This means that two polynomial commitments can be distinguished by any algorithm with probability at most $2^{-\lambda+1}$. □

Zero-Knowledge Evaluation Protocol We now build a zero-knowledge evaluation protocol, which is an Eval protocol for a hiding polynomial commitment. The zero-knowledge protocol shows that the prover must know a degree d polynomial $f(X)$ with bounded coefficients such that $f(z) \bmod p = y$ but does not leak any other information about f. Formally, we will show that the interactive $\mathsf{ZK\text{-}Eval}$ argument is honest verifier zero-knowledge according to Definition 3 by constructing an efficient simulator $\mathcal{S}$ that can generate a distribution of transcripts that is indistinguishable from honestly generated transcripts.

The idea for the $\mathsf{ZK\text{-}Eval}$ protocol is a simple blinding of the polynomial borrowed from Zero-Knowledge Sumcheck [CFS17] and Bulletproofs [BCC$^+$16b, BBB$^+$18]. Let $f(X)$ be the committed polynomial, using the hiding commitment scheme. The prover wants to convince the verifier that $f(z) \bmod p = y$. To do this the prover commits to a degree d polynomial

[7] Brakersi and Goldwasser define subgroup indistinguishability assumptions in a related but slightly different way.

$r(X)$ with random coefficients. The prover also reveals $y' \leftarrow r(z) \bmod p$. The verifier then sends a random challenge c and the prover and verifier can compute a commitment to $s(X) \leftarrow r(X) + c \cdot f(X)$. The random polynomial $r(X)$ ensures that $s(X)$ is distributed statistically close to a random polynomial. The prover could just reveal $s(X)$ and the verifier can check that $s(z) \bmod p = y' + c \cdot y \bmod p$. Instead of sending $s(X)$ in the clear, the prover can additionally just send the commitment randomness to provide the verifier with a non-hiding commitment to $s(X)$. The prover and verifier can then use the standard Eval protocol to efficiently evaluate s at z.

$\mathsf{ZK\text{-}Eval}(\mathsf{pp}, \mathsf{C} \in \mathbb{G}, z \in \mathbb{Z}_p, y \in \mathbb{Z}_p, d \in \mathbb{N}; f(X) \in \mathbb{Z}(b)[X], r \in \mathbb{Z})$:

1. $\mathcal{P}$ samples a random degree d $k(X) \xleftarrow{\$} \mathbb{Z}(\frac{p^2}{4} \cdot 2^\lambda)[X]$ and $r_k \xleftarrow{\$} [0, B \cdot 2^\lambda)$ and computes $\mathsf{R} \leftarrow \mathsf{g}^{k(q)+r_k \cdot q^{d+1}}$ and $y_k \leftarrow k(z) \bmod p$
2. $\mathcal{P}$ sends R and y_k to $\mathcal{V}$
3. $\mathcal{V}$ samples random $c \xleftarrow{\$} [-\frac{p-1}{2}, \frac{p-1}{2}]$ and sends it to $\mathcal{P}$
4. $\mathcal{P}$ computes $s(X) \leftarrow k(X) + c \cdot f(X)$, as well as $r_s \leftarrow r_k + c \cdot r$.
5. $\mathcal{P}$ sends r_s to $\mathcal{V}$
6. $\mathcal{P}$ and $\mathcal{V}$ compute $\mathsf{C}_s \leftarrow \mathsf{R} \cdot \mathsf{C}^c \cdot \mathsf{g}^{-q^{d+1} \cdot r_s}$ and $y_s \leftarrow y_k + c \cdot y \bmod p$ // $\mathsf{C}_s = \mathsf{g}^{s(q)}$
7. $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{EvalBounded}(\mathsf{pp}, \mathsf{C}_s, z, y_s, d, \frac{p^2}{4} \cdot 2^{\lambda+1}; s(X))$ // $s(z) \bmod p = y_s$

Theorem 6. Let Eval have perfect completeness and witness extended emulation for $q > b \cdot \varsigma_{p,d}$. Assuming that $\mathsf{commitH}$ is statistically hiding and both the order assumption and the strong RSA assumption hold for *GGen* the protocol EvalZK has perfect completeness, witness extended emulation and δ-statistical honest-verifier zero-knowledge for $q > 2^\lambda \cdot (p-1)(\frac{p^2-1}{2})^{\lceil \log_2(d+1) \rceil + 1} < p^{2\log_2(d+1)+4}$ and $\delta \leq d \cdot 2^{-\lambda}$.

Proof. A simple application of Theorem 7 and Lemma 8 shows that the protocol maintains witness extended emulation. The extractor extracts $s(X)$ and $s'(X)$ from the Eval protocol for challenges c and c'. We can directly use Lemma 8 to extract the witness $f(X)$ or a break of an assumption from these two transcripts. The bound on q grows by a factor of less than p^2 (or p^3 under the 2-Strong RSA assumption).

To show zero-knowledge, we build the simulator $\mathcal{S}$ as follows. Start with a polynomial $s(X) \xleftarrow{\$} \mathbb{Z}(\frac{p^2-1}{4})[X]$ with uniform random coefficients, and a blinding factor $r_s \xleftarrow{\$} [0, B \cdot 2^{2 \cdot \lambda + 1})$. The simulator $\mathcal{S}$ then chooses a random challenge $c \xleftarrow{\$} (-p/2, p/2)$ and computes $\mathsf{R} = \mathsf{g}^{s(q)+r_s \cdot q^{d+1}} \cdot \mathsf{C}^{-c}$. The simulator then performs the rest of the Eval protocol honestly using $s(X)$ as the witness.

The randomizer r_s is distributed identically to the honest r_s. Given the hiding property of the commitment scheme, R is statistically indistinguishable from any other commitment. Finally the simulated and the honest $s(X)$ have statistical distance at most $2^{-\lambda}$ from a random polynomial. The coefficients of $c \cdot f(x)$ are in $\mathbb{Z}(\frac{p^2}{4})$. The coefficients of the blinding polynomial $s(X)$ are sampled from a range that is larger by a factor 2^λ. So the distribution of coefficients of $s(X) = k(X) + c \cdot f(X)$ is at a statistical distance at most $2^{-\lambda}$ away from the uniform distribution over $\mathbb{Z}(\frac{p^2}{4})$. Since the distributions of simulated and real coefficients are both uniform but merely over different sets, the statistical distance between the simulated $s(X)$ and the real $s(X)$ is at most $d \cdot 2^{-\lambda}$. The evaluation with $\mathsf{EvalBounded}$ cannot leak more than $s(X)$ itself. The views of the simulated and real transcripts are, therefore, δ-close with $\delta \leq d \cdot 2^{-\lambda}$. Consequently, the protocol has δ-statistically honest verifier zero-knowledge. □

4.8 Performance

The polynomial commitment scheme has logarithmic proof size and verifier time in the degree d of the committed polynomial. It has highly batchable proofs and it is possible to evaluate n degree d polynomials at k points using only $2\log_2(d+1)$ group elements and $(k+1)\log_2(d+1)$ field elements (see Section 4.5). Note that this means the proof size is independent of n and linear in k but with a small constant ($15\log(d)$ bytes). We describe the performance of our scheme for different settings in Table 1.

Operation	$\|\mathsf{pp}\|$	Prover	Verifier	Communication
$\mathsf{Commit}(f(X))$	$1\ \mathbb{G}$	$O(\lambda d \log(d))\mathbb{G}$	-	$1\mathbb{G}$
$\mathsf{Commit}(f(X))$	$d\ \mathbb{G}$	$O(\frac{\lambda d}{\log(d)})\mathbb{G}$	-	$1\mathbb{G}$
$f(z) = y \in \mathbb{Z}_p$	$1\ \mathbb{G}$	$O(\lambda \log(d) d)\mathbb{G}$	$O(\lambda \log(d))\mathbb{G}$	$2\log(d)\mathbb{G}$ $+2\log(d)\mathbb{Z}_p$
$f(z) = y \in \mathbb{Z}_p$	$d\ \mathbb{G}$	$O(\lambda d)\mathbb{G}$	$O(\lambda \log(d))\mathbb{G}$	$2\log(d)\mathbb{G}$ $+2\log(d)\mathbb{Z}_p$
$f(\boldsymbol{z}) = \boldsymbol{y} \in \mathbb{Z}_p^k$	$d\ \mathbb{G}$	$O(\lambda d)\mathbb{G}$	$O(\lambda \log(d))\mathbb{G}$	$2\log(d)\mathbb{G}$ $+(k+1)\log(d)\mathbb{Z}_p$
$f(z) = y, g(z) = y' \in \mathbb{Z}_p$	$d\ \mathbb{G}$	$O(\lambda d)\mathbb{G}$	$O(\lambda \log(d))\mathbb{G}$	$2\log(d)\mathbb{G}$ $+2\log(d)\mathbb{Z}_p$

Table 1: $\mathbb{G}$ denotes the size of a group element for communication and a single group operation for computation. $\mathbb{Z}_p$ denotes the size of a field element, *i.e.*, λ bits. $|\mathsf{pp}|$ is the size of the public parameters (which is greater than one $\mathbb{G}$ when preprocessing is used), and d the degree of the polynomial. Rows 3-6 are for Eval proofs of different statements.

4.9 Comparison to Other Polynomial Commitment Schemes

4.9.1 Based on Pairings

The polynomial commitment by Kate *et al.* [KZG10] has evaluation proofs that consist of only a single element in a bilinear group and verifying an evaluation requires only a single pairing computation. However, this asymptotically optimal performance comes at the cost of a trusted setup procedure that outputs a structured reference string whose size is linear in the degree of the polynomial. Our DARK polynomial commitment scheme requires no trusted setup but pays for this reduced trust requirement with a proof size and verification work that scale logarithmically in the degree of the polynomial.

In the multivariate setting, our scheme is logarithmic in the total number of coefficients: $\mu \log(d)$ for a μ-variate polynomial of degree d in each variable. The multivariate extension of Kate *et al.*'s commitment scheme [ZGK+17] evaluation proofs consist of μ group elements.

4.9.2 Based on Discrete Logarithms

Bulletproofs [BCC+16b, BBB+18] is a proof system based on prime order groups in which the discrete logarithm is hard. As a core component it relies on an inner product argument which can be used as a polynomial commitment (see [WTs+18]). The polynomial commitment has logarithmic evaluation proofs with great constants. Unfortunately, the verifier time is linear in the size of the polynomial, i.e. $(d+1)^\mu$ for a μ- variate degree d polynomial. The more general version of the commitment [BCC+16b] can also give evaluation proofs with square root verifier time and square root proof size.

4.9.3 Based on Merkle Trees of Reed-Solomon Codewords

The FRI protocol [BBHR18] is an efficient interactive oracle proof (IOP) that a committed oracle is close to a Reed-Solomon codeword, meaning that the prover commits to large sequences of field elements and the verifier queries only a few specific elements rather than reading the entire sequence. The abstract functionality is cryptographically compiled with a Merkle tree, which results in constant-size commitments and element queries that are logarithmic in the length of the codeword, *i.e.*, the size of the oracle. FRI has been used in multiple recent zero-knowledge proof systems such as STARK [BBHR19], Aurora [BCR+19], and Fractal [COS19].

Since this oracle is a Reed-Solomon codeword, it represents the evaluations of a low-degree polynomial f on an evaluation set $S \subset \mathbb{F}$. In order to be used as a polynomial commitment scheme, the protocol needs to permit querying the polynomial outside of the evaluation set. DEEP-FRI [BGKS19] shows that this is possible and two recent works [ZXZS19, KPV19] makes the connection explicit by building a polynomial commitment scheme from FRI. This FRI-based polynomial commitment scheme have evaluation proofs of size and verifier time $O(\lambda \log^2(d))$ where λ is the security parameter and $d = \deg(f)$. To date, no extension to multivariate polynomials exists for FRI. The commitment relies only on symmetric cryptography and is plausibly quantum resistant.

4.9.4 Comparison

In Table 2 we give a comparison between different polynomial commitment schemes in the literature. In particular, we evaluate the size of the reference string ($|\mathsf{pp}|$), the prover and

verifier time, as well as the size of the evaluation proof ($|\pi|$). Column 2 indicates whether the setup is transparent, *i.e.*, whether the reference string is structured. The symbol $\mathbb{G}_U$ denotes a group of unknown order, $\mathbb{G}_B$ a group with a bilinear map (pairing), and $\mathbb{G}_P$ a group with prime (and known) order. Furthermore, EXP refers to exponentiation of a λ bit number in these groups, and H is either the size of a hash output, or the time it takes to compute a hash, depending on context.

Note that even when precise factors are given, the numbers should be interpreted as estimates. For example we chose to not display smaller order terms. Note also that the prover time for the group based schemes could be brought down by a log factor when using multi-exponentiation techniques.

Scheme	Transp.	$\lvert\mathsf{pp}\rvert$	Prover	Verifier	$\lvert\pi\rvert$
DARK *(this work)*	yes	$O(1)$	$O(d^\mu \mu \log(d))$ EXP	$3\mu \log(d)$ EXP	$2\mu \log(d)$ $\mathbb{G}_U$
Based on Pairings	no	d^μ $\mathbb{G}_B$	$O(d^\mu)$ EXP	μ $\mathsf{Pairing}$	μ $\mathbb{G}_B$
[BCC$^+$16b, $\sqrt{\cdot}$]	yes	$\sqrt{d^\mu}\mathbb{G}_P$	$O(d^\mu)$ EXP	$O(\sqrt{d^\mu})\mathsf{EXP}$	$O(\sqrt{d^\mu})$ $\mathbb{G}_P$
Bulletproofs	yes	$2d^\mu\mathbb{G}_P$	$O(d^\mu)$ EXP	$O(d^\mu)\mathsf{EXP}$	$2\mu \log(d)$ $\mathbb{G}_P$
FRI-based ($\mu = 1$)	yes	$O(1)$	$O(\lambda d)$ H	$O(\lambda \log^2(d))$ H	$O(\lambda \log^2(d))$ H

Table 2: Comparison table between different polynomial commitment schemes for an μ-variate polynomial of degree d.

5 Transparent SNARKs via Polynomial IOPs

5.1 Algebraic Linear IOPs

An *interactive oracle proof (IOP)* [BCS16, RRR16] is a multi-round interactive PCP: in each round of an IOP the verifier sends a message to the prover and the prover responds with a polynomial length proof, which the verifier can query via random access. A t-round ℓ-query IOP has t rounds of interaction in which the verifier makes exactly ℓ queries in each round. Linear IOPs [BBC$^+$19] are defined analogously except that in each round the prover sends a *linear* PCP [IKO07], in which the prover sends a single proof vector $\boldsymbol{\pi} \in \mathbb{F}^m$ and the verifier makes *linear queries* to π. Specifically, the PCP gives the verifier access to an oracle that receives queries of the form $\mathbf{q} \in \mathbb{F}^m$ and returns the inner product $\langle \boldsymbol{\pi}, \mathbf{q} \rangle$.

Bitansky *et al.* [BCI$^+$13] defined a linear PCP to be of *degree* (d_Q, d_V) if there is an explicit circuit of degree d_Q that derives the query vector from the verifier's random coins, and an explicit circuit of degree d_V that computes the verifier's decision from the query responses. In a multi-query PCP, d_Q refers to the maximum degree over all the independent circuits computing each query. Bitansky *et al.* called the linear PCP *algebraic* for a security parameter λ if it has degree $(\mathsf{poly}(\lambda), \mathsf{poly}(\lambda))$. The popular linear PCP based on *Quadratic Arithmetic Programs* (QAPs) implicit in the GGPR protocol [GGPR13] and follow-up works is an algebraic linear PCP with $d_Q \in O(m)$ and $d_V = 2$, where m is the size of the witness.

For the purposes of the present work, we are only interested in the algebraic nature of the query circuit and not the verifier's decision circuit. Of particular interest are linear PCPs where each query-and-response interaction corresponds to the evaluation of a fixed μ-variate degree d polynomial at a query point in $\mathbb{F}^\mu$. This description is equivalent to saying that the PCP is a vector of length $m = \binom{d+\mu}{\mu}$ and the query circuit is the vector of all μ-variate monomials of degree at most d (in some canonical order) evaluated at a point in $\mathbb{F}^\mu$. We call this a (μ, d) *Polynomial PCP* and define *Polynomial IOPs* analogously. As we will explain, we are interested in Polynomial PCPs where $\mu \ll m$ because we can cryptographically compile them into succinct arguments using polynomial commitments, in the same way that Merkle trees are used to compile classical (point) IOPs.

In general, evaluating the query circuit for a linear PCP requires $\Omega(m)$ work. However, a general "bootstrapping" technique can reduce the work for the verifier: the prover expands the verifier's random coins into a full query vector, and then provides the verifier with a second PCP demonstrating that this expansion was computed correctly. It may also help to allow the verifier to perform $O(m)$ work in a one-time preprocessing stage (for instance, to check the correctness of a PCP oracle), enabling it to perform sublinear "online" work when verifying arbitrary PCPs later. We call this a *preprocessing IOP*. In fact, we will see that any t-round (μ, d) algebraic linear IOP can be transformed into a $(t+1)$-round Polynomial

IOP in which the verifier preprocesses (μ, d) Polynomial PCPs, at most one for each distinct query.

We recall the formal definition of public-coin linear IOPs as well an algebraic linear IOPs. Since we are not interested in the algebraic nature of the decision algorithm, we omit specifying the decision polynomial. From here onwards we use algebraic linear IOP as shorthand for algebraic *query* linear IOP.

Definition 7 (Public-coin linear IOP)**.** Let $\mathcal{R}$ be a binary relation and $\mathbb{F}$ a finite field. A t-round ℓ-query public-coin linear IOP for $\mathcal{R}$ over $\mathbb{F}$ with soundness error ϵ and knowledge error δ and query length $\mathbf{m} = (m_1, ..., m_t)$ consists of two stateful PPT algorithms, the *prover* $\mathcal{P}$, and the *verifier* $\mathcal{V} = (\mathcal{Q}, \mathcal{D})$, where the verifier consists in turn of a public deterministic *query generator* $\mathcal{Q}$ and a *decision algorithm* $\mathcal{D}$, that satisfy the following requirements:

Protocol syntax. For each ith round there is a prover state $\mathsf{st}_i^{\mathcal{P}}$ and a verifier state $\mathsf{st}_i^{\mathcal{V}}$. For any common input x and $\mathcal{R}$ witness w, at round 0 the states are $\mathsf{st}_0^{\mathcal{P}} = (x, w)$ and $\mathsf{st}_0^{\mathcal{V}} = x$. In the ith round (starting at $i = 1$) the prover outputs a single[8] proof oracle $\mathcal{P}(\mathsf{st}_{i-1}^{\mathcal{P}}) \to \boldsymbol{\pi}_i \in \mathbb{F}^{m_i}$. The verifier samples public random coins $\mathit{coins}_i \xleftarrow{\$} \{0,1\}^*$ and the query generator computes a query matrix from the verifier state and these coins: $\mathcal{Q}(\mathsf{st}_{i-1}^{\mathcal{V}}, \mathit{coins}_i) \to \mathbf{Q}_i \in \mathbb{F}^{m_i \times \ell}$. The verifier obtains the linear oracle response vector $\boldsymbol{\pi}_i^\top \mathbf{Q}_i = \mathbf{a}_i \in \mathbb{F}^{1 \times \ell}$. The updated prover state is $\mathsf{st}_i^{\mathcal{P}} \leftarrow (\mathsf{st}_{i-1}^{\mathcal{P}}, \mathbf{Q}_i)$ and verifier state is $\mathsf{st}_i^{\mathcal{V}} \leftarrow (\mathsf{st}_{i-1}^{\mathcal{V}}, \mathit{coins}_i, \mathbf{a}_i)$ Finally, $\mathcal{D}(\mathsf{st}_t^{\mathcal{V}})$ returns 1 or 0.

(*Querying prior round oracles*: The syntax can be naturally extended so that in the ith round the verifier may query any oracle, whether sent in the ith round or earlier.)

Argument of Knowledge. As a proof system, $(\mathcal{P}, \mathcal{V})$ satisfies perfect completeness, soundness with respect to the relation $\mathcal{R}$ and with soundness error ϵ, and witness-extended emulation with respect $\mathcal{R}$ with knowledge error δ.

Furthermore, a linear IOP is **stateless** if for each $i \in [t]$, $\mathcal{Q}(\mathsf{st}_{i-1}^{\mathcal{V}}, \mathit{coins}_i) = \mathcal{Q}(i, \mathit{coins}_i)$. It has **algebraic queries** if, additionally, for each $i \in [t]$, the map $\mathit{coins}_i \xmapsto{\mathcal{Q}(i,\cdot)} \mathbf{Q}_i \in \mathbb{F}^{m_i \times \ell}$ decomposes into two maps, $\mathit{coins}_i \xmapsto{\mathcal{Q}_0(i,\cdot)} \boldsymbol{\Sigma}_i \xmapsto{\mathcal{Q}_1(i,\cdot)} \mathbf{Q}_i$, where $\boldsymbol{\Sigma}_i \in \mathbb{F}^{\mu_i \times \ell}$ is a matrix of $\mu_i < m_i$ rows and ℓ and $\mathcal{Q}_1(i, \cdot)$ is described by ℓ μ_i-variate polynomial functions of degree at most $d = \mathsf{poly}(\lambda)$: $\vec{p}_1, \ldots, \vec{p}_\ell : \mathbb{F}^{\mu_i} \to \mathbb{F}^{m_i}$ such that for all $k \in [\ell]$, $\vec{p}_k(\boldsymbol{\sigma}_{i,k}) = \mathbf{q}_{i,k}$, where $\boldsymbol{\sigma}_{i,k}$ and $\mathbf{q}_{i,k}$ denote the kth column of $\boldsymbol{\Sigma}_i$ and $\mathbf{Q}_i$, respectively.

Definition 8 (HVZK for public-coin linear IOPs)**.** Let $\mathsf{View}_{\langle \mathcal{P}(x,w), \mathcal{V}(x) \rangle}(\mathcal{V})$ denote the view of the verifier in the t-round ℓ-query interactive protocol described in Definition 7 on inputs (x, w) with prover algorithm $\mathcal{P}$ and verifier $\mathcal{V}$, consisting of all public-coin challenges and oracle outputs (this view is equivalent to the final state $\mathsf{st}_t^{\mathcal{V}}$). The interactive protocol has δ**-statistical honest-verifier zero-knowledge** if there exists a probabilistic polynomial time algorithm $\mathcal{S}$ such that for every $(x, w) \in \mathcal{R}$, the distribution $\mathcal{S}(x)$ is δ-close to $\mathsf{View}_{\langle \mathcal{P}(x,w), \mathcal{V}(x) \rangle}(\mathcal{V})$ (as distributions over the randomness of $\mathcal{P}$ and random public-coin challenges).

We note that the separation into two maps $\mathit{coins}_i \xmapsto{\mathcal{Q}_0(i,\cdot)} \boldsymbol{\Sigma}_i \xmapsto{\mathcal{Q}_1(i,\cdot)} \mathbf{Q}_i$ subtly relaxes the definition of Bitansky *et al.*, which instead requires that $\mathbf{Q}_i$ be determined via $\vec{p}_1, \ldots, \vec{p}_\ell$ evaluated at a random $\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^{\mu_i}$. The Bitansky *et al.* definition corresponds to the special case that $\mathcal{Q}_0(i, \cdot)$ samples a random element of $\mathbb{F}^{\mu_i}$ based on coins_i. The point is that $\mathcal{Q}_0$ can also do other computations that do not necessarily sample $\boldsymbol{r}$ uniformly, or even output a matrix rather than a vector. The separation into two steps is only meaningful when μ_i is smaller than m_i. The significance to SNARK constructions is that the query can be represented compactly as $\boldsymbol{\Sigma}_i$, and the prover will take advantage of the algebraic map $\mathcal{Q}_1(i, \cdot)$ to demonstrate that $\boldsymbol{\Sigma}_i$ was expanded correctly into $\mathbf{Q}_i$ and applied to the proof oracle π_i. We first present a standalone definition of Polynomial IOPs, and then explain how it is a special case of Algebraic Linear IOPs.

Definition 9 (Public coin Polynomial IOP)**.** Let $\mathcal{R}$ be a binary relation and $\mathbb{F}$ a finite field. Let $\mathbf{X} = (X_1, \ldots, X_\mu)$ be a vector of μ indeterminates. A (μ, d) Polynomial IOP for $\mathcal{R}$ over

[8]The prover may also output more than one proof oracle per round, however this doesn't add any power since two proof oracles of the same size may be viewed as a single (concatenated) oracle of twice the length.

$\mathbb{F}$ with soundness error ϵ and knowledge error δ consists of two stateful PPT algorithms, the *prover* $\mathcal{P}$, and the *verifier* $\mathcal{V}$, that satisfy the following requirements:

Protocol syntax. For each ith round there is a prover state $\mathsf{st}_i^{\mathcal{P}}$ and a verifier state $\mathsf{st}_i^{\mathcal{V}}$. For any common input x and $\mathcal{R}$ witness w, at round 0 the states are $\mathsf{st}_0^{\mathcal{P}} = (x, w)$ and $\mathsf{st}_0^{\mathcal{V}} = x$. In the ith round (starting at $i = 1$) the prover outputs a single proof oracle $\mathcal{P}(\mathsf{st}_{i-1}^{\mathcal{P}}) \to \pi_i$, which is a polynomial $\pi_i(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$. The verifier deterministically computes the query matrix $\mathbf{\Sigma}_i \in \mathbb{F}^{\mu \times \ell}$ from its state and a string of public random bits $\mathit{coins}_i \xleftarrow{\$} \{0,1\}^*$, *i.e*, $\mathcal{V}(\mathsf{st}_{i-1}^{\mathcal{V}}, \mathit{coins}_i) \to \mathbf{\Sigma}_i$. This query matrix is interpreted as a list of ℓ points in $\mathbb{F}^\mu$ denoted $(\boldsymbol{\sigma}_{i,1}, \ldots, \boldsymbol{\sigma}_{i,\ell})$. The oracle π_i is queried on all points in this list, producing the response vector $(\pi_i(\boldsymbol{\sigma}_{i,1}), \ldots, \pi_\ell(\boldsymbol{\sigma}_{i,\ell})) = \mathbf{a}_i \in \mathbb{F}^{1 \times \ell}$. The updated prover state is $\mathsf{st}_i^{\mathcal{P}} \leftarrow (\mathsf{st}_{i-1}^{\mathcal{P}}, \mathbf{\Sigma}_i)$ and verifier state is $\mathsf{st}_i^{\mathcal{V}} \leftarrow (\mathsf{st}_{i-1}^{\mathcal{V}}, \mathbf{\Sigma}_i, \mathbf{a}_i)$. Finally, $\mathcal{V}(\mathsf{st}_t^{\mathcal{V}})$ returns 1 or 0.

(*Extensions: multiple and prior round oracles; various arity.* The syntax can be naturally extended such that multiple oracles are sent in the ith round; that the verifier may query oracles sent in the ith round or earlier; or that some of the oracles are polynomials in fewer variables than μ.)

Argument of Knowledge. As a proof system, $(\mathcal{P}, \mathcal{V})$ satisfies perfect completeness, soundness with respect to the relation $\mathcal{R}$ and with soundness error ϵ, and witness-extended emulation with respect $\mathcal{R}$ with knowledge error δ.

Furthermore, a Polynomial IOP is **stateless** if for each $i \in [t]$, $\mathcal{V}(\mathsf{st}_{i-1}^{\mathcal{V}}, \mathit{coins}_i) = \mathcal{V}(i, \mathit{coins}_i)$.

Polynomial IOPs as a subclass of Algebraic Linear IOPs In a Polynomial IOP, the two-step map $\mathit{coins}_i \xmapsto{\mathcal{V}(i,\cdot)} (\boldsymbol{\sigma}_{i,1}, \ldots, \boldsymbol{\sigma}_{i,\ell}) \xmapsto{\mathbf{M}} (\mathbf{q}_{i,1}, \ldots, \mathbf{q}_{i,\ell})$ is a special case of the two-step map $\mathit{coins}_i \xmapsto{\mathcal{Q}_0(i,\cdot)} \mathbf{\Sigma}_i \xmapsto{\mathcal{Q}_1(i,\cdot)} \mathbf{Q}_i$ in an algebraic linear IOP. Here $\mathbf{M} : \mathbb{F}^\mu \to \mathbb{F}^m$ represents the vector of monomials of degree at most d (in some canonical order) and the map associated with $\mathbf{M}$ is evaluation. Note that there are $m = \binom{\mu+d}{d}$ such monomials. Furthermore, for any $\mathbf{q}_{i,k}$, the inner product $\boldsymbol{\pi}_i^{\mathsf{T}} \mathbf{q}_{i,k}$ corresponds to the evaluation at $\boldsymbol{\sigma}_{i,k}$ of the polynomial $\pi_i(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$, whose coefficient vector (in the same canonical monomial order) is equal to $\boldsymbol{\pi}_i$.

5.2 Polynomial IOP reductions

In this section we show that one can construct any algebraic linear IOP from a (multivariate) Polynomial IOP. This construction rests on two tools for univariate Polynomial IOPs that we cover first:

- *Coefficient queries.* The verifier verifies that an indicated coefficient of a polynomial oracle has a given value.
- *Inner products.* The verifier verifies that the inner product of the coefficient vectors of two polynomial oracles equals a given value.

5.2.1 Coefficient queries

The following is a $(1, d)$-Polynomial IOP for the statement $f_i = a$ with respect to a polynomial $f(X) = \sum_{j=0}^{d} f_j X^j$.

- *Prover*: Split $f(X)$ about the term X^i into $f_L(X)$ (of degree at most $i-1$) and $f_R(X)$ (of degree at most $d-i-1$) such that $f(X) = f_L(X) + aX^i + X^{i+1} f_R(X)$. Send polynomials $f_L(X)$ and $f_R(X)$.
- *Verifier*: Sample uniform random $\beta \xleftarrow{\$} \mathbb{F}_p$ and query for $y_L \leftarrow f_L(\beta)$, $y_R \leftarrow f_R(\beta)$, and $y \leftarrow f(\beta)$. Check that $y = y_L + a\beta^i + \beta^{i+1} y_R \bmod p$ and return 0 (abort) if not. Otherwise output 1 (accept).

The verifier only accepts given proof oracles for polynomials f, f_L, and f_R in $\mathbb{F}_p[X]$ of degree at most d, $i-1$ and $d-i-1$ such that $f(\beta) = f_L(\beta) + a\beta^i + \beta^{i+1} f_R(\beta)$ for random $\beta \xleftarrow{\$} \mathbb{F}$. Via the Schwartz-Zippel lemma, if $f(X) \neq f_L(X) + aX^i + X^{i+1} f_R(X)$ then the

verifier would accept with probability at most $d/|\mathbb{F}|$, because the highest degree term in this equation is $X^{i+1}f_R(X)$ and its degree is at most d. This implies that a is the ith coefficient of f.

Note that this description assumes that the verifier is assured that the proof oracles for f_L and f_R have degrees $i-1$ and $d-i-1$, respectively. If no such assurance is given, then $f_L(X)$ should be shifted by $d-i+1$ digits. In particular, the proof oracle should $f_L^\star(X) = X^{d-i+1}f_L(X)$, in which case the verifier obtains the evaluation $y_L^\star = y_L\beta^{d-i+1}$ along with an assurance that $f_L^\star(X)$ has degree at most d. The verifier then tests $y = (\beta^{d-i+1})^{-1}y_L^\star + a\beta^i + \beta^{i+1}y_R$. This test admits false positives with probability at most $2d/|\mathbb{F}|$.

5.2.2 Inner product

The following is an IOP where the prover first sends two degree d univariate polynomial oracles f, g and proves to the verifier that $\langle \mathbf{f}, \mathbf{g}^r \rangle = a$ where $\mathbf{f}, \mathbf{g}$ denote the coefficient vectors of f, g respectively and $\mathbf{g}^r$ is the reverse of $\mathbf{g}$. This argument is sufficient for our application to transforming algebraic linear IOPs into Polynomial IOPs. It is also possible to prove the inner product $\langle \mathbf{f}, \mathbf{g} \rangle$ by combining this IOP together with another one that probes the relation $g(X) = X^d g^r(X^{-1})$ in a random point $z \xleftarrow{\$} \mathbb{F}\backslash\{0\}$, and thereby shows that $\mathbf{g}$ and $\mathbf{g}^r$ have the same coefficients only reversed. We omit this more elaborate construction as it is not needed for any of our applications.

- *Prover*: Sends proof oracles for $f(X)$, $g(X)$, and the degree $2d$ polynomial product $h(X) = f(X) \cdot g(X)$ to the verifier.
- *Verifier*: Chooses $\beta \xleftarrow{\$} \mathbb{F}$ and queries for $y_1 \leftarrow f(\beta)$, $y_2 \leftarrow g(\beta)$, and $y_3 \leftarrow h(\beta)$. Check that $y_1 y_2 = y_3$ and return 0 (abort) if not.
- Prover and verifier engage in the 1 round IOP (Section 5.2.1) for proving that the dth coefficient (*i.e.*, on term X^d) of $h(X)$ is equal to a. (Note that the proof oracles for this subprotocol can all be sent in the first round, so this does not add an additional round).

Via Schwartz-Zippel, if $h(X) \neq f(X) \cdot g(X)$ then the verifier's check $y_1 y_2 = y_3$ at the random point β fails with probability at least $(|\mathbb{F}| - 2d)/|\mathbb{F}|$. Observe that the middle coefficient of $h(X)$ is equal to $\sum_{i=0}^{d} f_i g_{d-i} = \sum_{i=0}^{d} f_i g_i^r = \langle \mathbf{f}, \mathbf{g}^r \rangle = a$.

Reducing algebraic linear IOPs to Polynomial IOPs

Theorem 7. Any public-coin t-round stateless algebraic linear IOP can be implemented with a $t+1$-round Polynomial IOP with preprocessing. Suppose the original ℓ-query IOP is (μ, d) algebraic with query length $(m_1, ..., m_t)$ then the resulting Polynomial IOP has for each $i \in [t]$: 2ℓ degree m_i univariate polynomial oracles, ℓ pre-processed multivariate oracles of degree d and $\mu + 1$ variables, ℓ degree $2m_i$ univariate polynomial oracles and 2ℓ degree $2m_i$ univariate polynomial oracles. There is exactly one query to each oracle on a random point in $\mathbb{F}$. The soundness loss of the transformation is $\mathsf{negl}(\lambda)$ for a sufficiently large field (*i.e.*, whose cardinality is exponential in λ).

Proof. By definition of a (μ, d) algebraic linear IOP, in each ith round of the IOP there are ℓ query generation functions $\vec{p}_{i,1}, \ldots, \vec{p}_{i,\ell} : \mathbb{F}^\mu \to \mathbb{F}^{m_i}$, where each $\vec{p}_{i,k}$ is a vector whose jth component is a μ-variate degree-d polynomial $p_{i,k,j}$. These polynomials are applied to a seed matrix $\boldsymbol{\sigma}_{i,k} \in \mathbb{F}^\mu$ (which is identifiable with or derived from the verifier's ith round public-coin randomness $coins_i$); this evaluation produces $\vec{p}_{i,k}(\boldsymbol{\sigma}_{i,k}) = \mathbf{q}_{i,k} \in \mathbb{F}^{m_i}$ for all $k \in [\ell]$. The vectors $\mathbf{q}_{i,k}$ are the columns of the query matrix $\mathbf{Q}_i \in \mathbb{F}^{m_i \times \ell}$.

Preprocessed oracles For each round i of the original algebraic linear IOP, the prover and verifier preprocess $(\mu+1)$-variate degree-d polynomial oracles. For each $k \in [\ell]$, the vector of polynomials $\vec{p}_{i,k} = (p_{i,k,1}, \ldots, p_{i,k,m_i}) \in (\mathbb{F}[\mathbf{X}])^{m_i}$ with $\mathbf{X} = (X_1, \ldots, X_\mu)$ is encoded as a single polynomial in $\mu + 1$ variables as follows. Introduce a new indeterminate Z, and then define $\tilde{P}_{i,k}(\mathbf{X}, Z) := \sum_{j=1}^{m_i} p_{i,k,j}(\mathbf{X}) Z^j \in \mathbb{F}[\mathbf{X}, Z]$. The prover and verifier establish the oracle $\tilde{P}_{i,k}$, meaning that the verifier queries this oracle on enough points to be reassured that it is correct everywhere.

The transformed IOP The original algebraic linear IOP is modified as follows.

- Wherever the original IOP prover sends an oracle $\boldsymbol{\pi}_i$ of length m_i, the new prover sends a degree $m_i - 1$ univariate polynomial oracle f_{π_i} whose coefficient vector is *the reverse* of $\boldsymbol{\pi}_i$.
- Wherever the original IOP verifier makes ℓ queries within a round to a particular proof oracle $\boldsymbol{\pi}_i$, where queries are defined by query matrix $\mathbf{Q}_i \in \mathbb{F}^{m_i \times \ell}$, consisting of column query vectors $(\mathbf{q}_{i,1}, ..., \mathbf{q}_{i,\ell})$, the new prover and verifier engage in the following interactive subprotocol for each $k \in [\ell]$ in order to replace the kth linear query $\langle \boldsymbol{\pi}_i, \mathbf{q}_{i,k} \rangle$:
  - Verifier: Run the original IOP verifier to get the public coin seed matrix $\boldsymbol{\Sigma}_i$ and send it to the prover.
  - Prover: Derive the query matrix $\mathbf{Q}_i$ from $\boldsymbol{\Sigma}_i$ using the polynomials $\vec{p}_{i,1}, \ldots, \vec{p}_{i,\ell}$. Send an oracle for the polynomial $F_{i,k}$ whose coefficient vector is $\mathbf{q}_{i,k}$.
  - Verifier: Sample uniform random $\beta \xleftarrow{\$} \mathbb{F}$ and query both $F_{i,k}$ and $\tilde{P}_{i,k}$ (the kth preprocessed oracle for round i) at β in order to check that $F_{i,k}(\beta) = \tilde{P}_{i,k}(\boldsymbol{\sigma}_{i,k}, \beta)$. If the check fails, abort and output 0.
  - Prover: Compute $a_{i,k} = \langle \boldsymbol{\pi}, \mathbf{q}_{i,k} \rangle$ and send $a_{i,k}$ to the verifier.
  - The prover and verifier run the inner product Polynomial IOP from Section 5.2.2 on the oracles $F_{i,k}$ and f_{π_i} to convince the verifier that $a_{i,k} = \langle \mathbf{q}_{i,k}, \boldsymbol{\pi}_i \rangle$. If the inner product subprotocol fails the verifier aborts and outputs 0.

If all substeps succeed, then the verifier obtains correct output of each oracle query; in other words, the responses are identical in the new and original IOP. These outputs are passed to the original verifier decision algorithm, which outputs 0 or 1.

Soundness and completeness If the prover is honest then the verifier receives the same exact query-response pairs $(\mathbf{q}_{i,k}, a_{i,k})$ as the original IOP verifier and runs the same decision algorithm, and therefore the protocol inherits the completeness of the original IOP. As for soundness, an adversary who sends a polynomial oracle $F^*_{i,k}$ whose coefficient vector is *not* $\mathbf{q}_{i,k}$, fails with overwhelming likelihood. To see this, note that since $\mathbf{q}_{i,k} = \vec{p}_{i,k}(\boldsymbol{\sigma}_k)$, the check that $F_{i,k}(\beta) = \tilde{P}_{i,k}(\boldsymbol{\sigma}_{i,k}, \beta)$ at a random β fails with overwhelming probability by the Schwartz-Zippel lemma. Similarly, an adversary who provides an incorrect $a^*_{i,k} \neq \langle \boldsymbol{\pi}_i, \mathbf{q}_{i,k} \rangle$ fails the inner-product IOP with overwhelming probability. Therefore, if the original IOP soundness error is ϵ then by a union bound the new soundness error is $\epsilon + \mathsf{negl}(\lambda)$. A similar composition argument follows for knowledge extraction.

Round complexity The prover and verifier can first simulate the t-round original IOP on the verifier's public-coin challenges, proceeding as if all queries were answered honestly. Wherever the original IOP prover would send an oracle for the vector $\boldsymbol{\pi}_i$ the prover sends f_{π_i}. Then, after the verifier has sent its final public coin challenge from the original IOP, there is one more round in which the prover sends all $F_{i,k}$ for the kth query vector in the ith round and all the purported answers $a_{i,k}$ to the kth query in the ith round. The prover and verifier engage in the protocol above to prove that these answers are correct. The inner product subprotocol for each $F_{i,k}$ with f_{π_i} can be done in parallel with the check that $F_{i,k}(\beta) = \tilde{P}_{i,k}(\boldsymbol{\sigma}_{i,k}, \beta)$. Therefore, there is only one extra round. □

5.3 Compiling Polynomial IOPs

Let $\Gamma = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$ be a multivariate polynomial commitment scheme. Given any t-round Polynomial IOP for $\mathcal{R}$ over $\mathbb{F}$, we construct an interactive protocol $\Pi = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ as follows. For clarity in our explanation, Π consists of t *outer rounds* corresponding to the original IOP rounds and *subrounds* where subprotocols may add additional rounds of interaction between outer rounds.

- Setup: Run $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
- In any round where the IOP prover sends a (μ, d) polynomial proof oracle $\boldsymbol{\pi} : \mathbb{F}^\mu \to \mathbb{F}$, in the corresponding *outer round* of Π, $\mathcal{P}$ sends the commitment $c_{\boldsymbol{\pi}} \leftarrow \mathsf{Commit}(\mathsf{pp}; \boldsymbol{\pi})$

- In any round where the IOP verifier makes an *evaluation* query $\mathbf{z}$ to a (μ, d) polynomial proof oracle $\boldsymbol{\pi}$, in the corresponding *outer round* of Π, insert an interactive execution of $\mathsf{Eval}(\mathsf{pp}, c_\pi, \mathbf{z}, y, \mu, d; \boldsymbol{\pi})$ between $\mathcal{P}$ and $\mathcal{V}$, where $\boldsymbol{\pi}(\mathbf{z}) = y$.

If $\mathcal{V}$ does not abort in any of these subprotocols, then it receives a simulated IOP transcript of oracle queries and responses. It runs the IOP verifier decision algorithm on this transcript and outputs the result.

Optimization: delayed evaluation As an optimization to reduce round-complexity and enable batching techniques, all invocations of Eval can be delayed until the final round, and heuristically could be run in parallel. Delaying the evaluations until the final round does not affect our analysis. However, our analysis does not consider parallel execution of the Eval subprotocols. We assume the protocol transcript contains an isolated copy of each Eval instance and does not interleave messages or re-use randomness.

Theorem 8. If the polynomial commitment scheme Γ has witness-extended emulation, and if the t-round Polynomial IOP for $\mathcal{R}$ has negligible knowledge error, then Π is a public-coin interactive argument for $\mathcal{R}$ that has witness-extended emulation. The compilation also preserves HVZK if Γ is hiding and Eval is HVZK.

The full proof is provided in Appendix B. HVZK is shown by a straightforward composition of the simulators for Eval and the original IOP simulator. The emulator E works as follows. Given the IP adversary P', E simulates an IOP adversary P'_O by using the Eval emulator E_{Eval} to extract proof oracles (*i.e.*, polynomials) from any commitment that P' sends and subsequently opens at an evaluation point. We argue that P'_O is successful whenever P' is successful, with negligible loss. (The only events that cause P'_O to fail when P' succeeds is if E_{Eval} fails to extract from a successful Eval or P' succesfully opens a commitment inconsistently with an extracted polynomial). E then runs the IOP knowledge extractor with P'_O to extract a witness for the input.

5.4 Concrete Instantiations

We consider examples of Polynomial IOPs to which this compiler can be applied: STARK [BBHR19]; Sonic [MBKM19] and its improvements PLONK [GWC19] and Marlin [CHM+19]; Spartan [Set19], and the popular QAP of Gennaro *et al.* [GGPR13]. For the purpose of the following discussion, we refer to the complexity of an NP relation $\mathcal{R}$ in various forms:

- $\mathcal{R}$ has *arithmetic complexity* n if the function computing $\mathcal{R}(x, w)$ can be expressed as 2-fan-in arithmetic circuit with a total of n gates.
- $\mathcal{R}$ has *multiplicative complexity* n if the function computing $\mathcal{R}(x, w)$ can be expressed an arithmetic circuit with a total of n multiplication gates, where each multiplication gate has 2 inputs.
- $\mathcal{R}$ has *R1CS complexity*[9] n if the function computing $\mathcal{R}(x, w)$ can be expressed as an R1CS instance (A, B, C, v, w) where $A, B, C \in \mathbb{F}^{m \times (\ell+1)}$, $(v, w) \in \mathbb{F}^\ell$, and n is the maximum number of non-zero entries in either A, B, or C.

Theorem 13 provides the main theoretical result of this work, tying together the new DARK polynomial commitment scheme (Theorem 1), the compilation of HVZK Polynomial IOPs into zk-SNARKs with preprocessing using polynomial commitments (Theorem 8), and a concrete univariate Polynomial IOP introduced in Sonic [MBKM19] (Theorem 10) or follow-up works.

5.4.1 Sonic

Sonic is a zk-SNARK system that has a universal trusted setup, which produces a Structured Reference String (SRS) of n group elements that can be used to prove any statement represented as an arithmetic circuit with at most n gates. The SRS can also be updated

[9]The arithmetic complexity and R1CS complexity are similar, but vary because the R1CS constraints correspond to the wiring of an arithmetic circuit with unrestricted fan-in.

without re-doing the initial setup, for instance, to enable proving larger circuits, or to increase the distribution of trust. The result in Sonic was not presented using the language of IOPs. Furthermore, the result also relied on a special construction of polynomial commitments (a modification of Kate *et al.* [KZG10]) that forces the prover to commit to a Laurent polynomial with no constant term. Given our generic reduction from coefficient queries to evaluation queries (Section 5.2.1), we re-characterize the main theorem of Sonic as follows:

Theorem 9 (Sonic Bivariate, [MBKM19])**.** There exists a 2-round HVZK Polynomial IOP with preprocessing for any NP relation $\mathcal{R}$ (with multiplicative complexity n) that makes 1 query to a bivariate polynomial oracle of degree n on each variable, and 6 queries to degree n univariate polynomial oracles. The preprocessing verifier does $O(n)$ work to check the single bivariate oracle.

The number of univariate queries increased from the original 3 in Sonic (with special commitments) to 6 with our generic coefficient query technique. If we were to compile the bivariate query directly using our multivariate commitment scheme this would result in $O(n^2)$ prover time (a bivariate polynomial with degree n on each variable is converted to a univariate polynomial of degree roughly n^2). However, Sonic also provides a way to replace the bivariate polynomial with several degree n univariate polynomials and more rounds of communication.

Theorem 10 (Sonic Univariate, [MBKM19])**.** There is a 5-round HVZK Polynomial IOP with preprocessing for any NP relation $\mathcal{R}$ (with multiplicative complexity n) that makes 39 queries overall to 27 univariate degree $2n$ polynomial oracles. The total number of distinct query points is 12. The preprocessing verifier does $O(n)$ work to check 12 of the univariate degree $2n$ polynomials.

The recent proof systems PLONK and Marlin improve on Sonic by constructing a different Polynomial IOP. They achieve the following:

Theorem 11 (PLONK, [GWC19])**.** There is a 3-round HVZK Polynomial IOP with preprocessing for any NP relation $\mathcal{R}$ (with arithmetic complexity n) that makes 12 queries overall to 12 univariate degree n polynomial oracles. The total number of distinct query points is 2. The preprocessing verifier does $O(n)$ work to check 7 of the univariate degree n polynomials.

Theorem 12 (Marlin, [CHM+19])**.** There exists a 4-round HVZK Polynomial IOP with preprocessing for any NP relation $\mathcal{R}$ (with R1CS complexity n) that makes 20 queries at 3 distinct query points to 19 univariate degree polynomial oracles of maximum degree $6n$. The preprocessing verifier does $O(n)$ work to check 9 univariate degree n polynomials.

Combining the Sonic Polynomial IOP with the new transparent polynomial compiler of Section 4 gives the following result. Similar results are obtained by using PLONK or Marlin instead.

Theorem 13 (**New Transparent zk-SNARK**)**.** There exists an $O(\log n)$-round public-coin interactive argument of knowledge for any NP relation of arithmetic complexity n that has $O(\log n)$ communication, $O(\log n)$ "online" verification, quasilinear prover time, and a preprocessing step that is verifiable in quasilinear time. The argument of knowledge has witness-extended emulation assuming it is instantiated with a group $\mathbb{G}$ for which the Strong RSA Assumption, and the Adaptive Root Assumption hold.

Proof. We apply the univariate polynomial commitment scheme from Section 4 to the 5-round Polynomial IOP from Theorem 10. Denote this commitment scheme by $\Gamma = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$

The preprocessing requires running Commit on 12 univariate degree n polynomials, which involves a quasilinear number of group operations in the group of unknown order $\mathbb{G}$ determined by Setup. The prover sends a constant number of proof oracles of degree $2n$ to the verifier, which also takes a quasilinear number of group operations. Finally, the 39 queries are replaced with at most 39 invocations of Eval, which adds $O(\log n)$ rounds and has $O(\log n)$ communication. By Theorem 1 (Γ has witness extended emulation) and Theorem 8, the compiled interactive argument has witness-extended emulation. □

5.4.2 STARK

The STARK proof system [BBHR19] builds an IOP for uniform computations, specified by a program P and timebound T on the running time of P. The IOP itself is then compiled into a concrete proof system using FRI [BBHR18] and Merkle trees. The STARK IOP can be cast as a univariate Polynomial IOP.

The IOP construction begins with an *algebraic intermediate representation* (AIR) of the program P. We present a simplified version of the original STARK AIR language for the purpose of illustrating how to recast the STARK IOP as a Polynomial IOP. The original AIR is more complex for efficiency reasons.

The AIR represents a computation as an algebraic execution trace of the program P for T timesteps. The AIR views the program as a system of n registers and a transition function. At every timestep each register holds an element of the finite field $\mathbb{F}$. Given a vector $\mathbf{w}_i \in \mathbb{F}^n$ representing the states of the registers at timestep i, the transition function determines the vector $\mathbf{w}_{i+1} \in \mathbb{F}^n$ representing the state of the registers at timestep $i+1$. The AIR represents the transition function as a system of constraints given by a vector of $2n$-variate polynomials $\boldsymbol{\mathcal{P}}$, and furthermore specifies a vector $\boldsymbol{\mathcal{B}}$ of tuples $([T], [n], \mathbb{F})$ representing "boundary conditions" of the form $\mathbf{w}_i[j] = \alpha$ for the value of the jth register at timestep i.

Definition 10. The relation $\mathcal{R}_{\mathsf{AIR}}$ is the set of all instance-witness pairs $((\mathbb{F}, T, n, \boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{B}}), W)$ satisfying the following description:
Instance. An instance is a tuple $(\mathbb{F}, T, n, \boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{B}})$ where

- $\mathbb{F}$ is a finite field.
- $T \in \mathbb{N}$ is the number of time steps.
- n is the number of registers.
- $\boldsymbol{\mathcal{P}} : \mathbb{F}^{2n} \to \mathbb{F}^k$ is a polynomial vector function whose k components $(\mathcal{P}_1, ..., \mathcal{P}_k)$ are each $2n$-variate polynomials of degree at most d called the "state transition constraints". On input $\mathbf{z} \in \mathbb{F}^{2n}$: $\boldsymbol{\mathcal{P}}(\mathbf{z}) = (\mathcal{P}_1(\mathbf{z}), ..., \mathcal{P}_k(\mathbf{z})) \in \mathbb{F}^k$.
- $\boldsymbol{\mathcal{B}} \in ([T] \times [n] \times \mathbb{F})^\ell$ are ℓ tuples, called the "boundary conditions".

Witness. A witness is a table $W \in \mathbb{F}^{T \times n}$ where each row $i \in [T]$ represents the full state of the system at time i, and each column $j \in [n]$ tracks the value of register j across time. A witness W is a valid witness for the instance $x = (\mathbb{F}, T, n, \boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{B}})$ if and only if the following conditions are satisfied:

- **State transition consistency**: $\boldsymbol{\mathcal{P}}(W[i,1], \ldots, W[i,n], W[i{+}1,1], \ldots, W[i{+}1,n]) = \mathbf{0}$ for all $i \in [T-1]$.
- **Boundary condition satisfaction**: $W[i,j] = \alpha$ for every tuple $(i,j,\alpha) \in \boldsymbol{\mathcal{B}}$.

The language $\mathcal{L}_{\mathsf{AIR}}$ is defined as $\mathcal{L}_{\mathsf{AIR}} = \{x = (\mathbb{F}, T, n, \boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{B}}) \mid \exists W\ (x, W) \in \mathcal{R}_{\mathsf{AIR}}\}$.

Theorem 14. There is a 2-round univariate Polynomial IOP for $\mathcal{R}_{\mathsf{AIR}}$ with preprocessing that makes $k + n + 2$ queries to $n + 2$ polynomials of degree at most T. The prover has complexity $\tilde{O}(nT)$ and the verifier has complexity $O(n \log T)$. The preprocessing verifier does $O(T)$ work.

STARK Polynomial IOP We sketch how this Polynomial IOP is constructed, omitting many details (see the STARK paper [BBHR19] for further details).

Let g be a generator of $\mathbb{F}^\times$. The preprocessing consists of computing the nonzero polynomial $z(X) = \prod_{i=1}^{T-1}(X - g^i)$ which satisfies $z(g^1) = z(g^2) = \cdots = z(g^{T-1}) = 0$. The online interaction is as follows:

1. The prover computing the n polynomials $w_j(X)$ of degree at most $T-1$ such that $w_j(g^i) = w[i,j]$, and sends n polynomial oracles to the verifier, one for each $w_j(X)$.
2. The verifier sends a random weight vector $\boldsymbol{\beta} \xleftarrow{\$} \mathbb{F}^k$.
3. The prover computes $f(X) = \boldsymbol{\beta}^{\mathsf{T}} \boldsymbol{\mathcal{P}}(w_1(X), \ldots, w_n(X), w_1(g \cdot X), \ldots, w_n(g \cdot X))$. The prover sends $q(X) = f(X)/z(X)$ to the verifier.

Note that for a valid witness, $f(g^1) = f(g^2) = \cdots = f(g^{T-1}) = 0$, so $z(X)$ divides $f(X)$. Note further that $q(X)$ has degree at most d. The verifier's queries to the proof oracles it received from the prover are as follows:

- For all boundary constraints of the form (i, j, α) the verifies queries for $w_j(g^i)$, and if $w_j(g^i) \neq \alpha$ then the verifier aborts and rejects.

- For a random point $h \xleftarrow{\$} \mathbb{F}$, the verifier queries for $u_j \leftarrow w_j(h)$ and $v_j \leftarrow w_j(gh)$ for all $j \in [n]$, as well as for $q(h)$ and $z(h)$. Finally it checks that $\boldsymbol{\beta}^{\mathsf{T}} \boldsymbol{\mathcal{P}}(u_1, \ldots, u_n, v_1, \ldots, v_n) = q(h) \cdot z(h)$, and if not it aborts and rejects.

If g instead can be chosen as an element of order $T - 1$ in $\mathbb{F}$, then the preprocessing phase can be omitted. In this case $z(X) = X^{T-1} - 1$ and it can be evaluated by the verifier locally in $O(\log T)$ time.

5.4.3 Spartan

Spartan [Set19] transforms an arbitrary circuit satisfaction problem into a Polynomial IOP based on an arithmetization technique developed by Blumberg *et al.* [BTVW14], which improved on the classical techniques of Babai, Fortnow, and Lund [BFL91]. Specifically, satisfiability of a 2-fan-in arithmetic circuit on n gates can be transformed into the expression:

$$\sum_{x,y,z \in \{0,1\}^{\log n}} G(x, y, z) = 0 \tag{1}$$

for a multilinear polynomial G on $3 \log n$ variables over $\mathbb{F}$. Furthermore, G decomposes into the form:

$$G(x, y, z) = A(x, y, z)F(x) + B(x, y, z)F(y) + C(x, y, z)F(y)F(z)$$

where A, B, C, and F are all multilinear polynomials. The polynomials A, B, C are derived from the arithmetic circuit defining the relation $\mathcal{R}$ and are input-independent. F is degree 1 with $\log n$ variables and is derived from a particular $(x, w) \in \mathcal{R}$. The classical LFKN sum-check protocol is applied in order to prove Expression 1 in a $3 \log n$ round Polynomial IOP, where the prover's oracle consist of Z and the low-degree polynomials sent in the sumcheck. Since the extra low-degree polynomials are constant size they can be read entirely by the verifier in constant time rather than via oracle access, and hence we ignore them in the total oracle count. The verifier must also evaluate A, B, C locally, which come from the multilinear extension of the circuit. This can be done in $O(\log n)$ time for certain circuits with a succinct representation. The main result in Spartan can be summarized in our framework as follows:

Theorem 15 ([Set19])**.** There exists a $3 \log n$ round Polynomial IOP for any NP relation $\mathcal{R}$ computed by any circuit with arithmetic complexity n, which makes three queries to a $\log n$-variate degree 1 polynomial oracle.

Applying our multivariate compiler to the Spartan Polynomial IOP we obtain an $O(\log n)$-round public-coin interactive argument of knowledge for circuits size n, where the verifier's work is dependent on the succinctness of the circuit representation (*i.e.*, the complexity of evaluating the multilinear extension of the circuit). In our multivariate scheme (Section 4.6), the $\log n$-variate degree 1 polynomial is tranformed into a univariate polynomial of degree n. With only three queries overall, the communication is just $6 \log n$ group elements and $6 \log n$ field elements.

5.4.4 Quadratic Arithmetic Programs

Quadratic Arithmetic Programs (QAPs) can be expressed as linear PCPs [BCI+13, BCG+13]. We review here how to express QAPs as a one round public-coin $(1, n)$ algebraic IOP. (This captures the satisfiability of any circuit with multiplicative complexity n, which is first translated to a system of quadratic equations over degree n polynomials.) Each linear query is computed by a vector of degree n univariate polynomials evaluated at a random point chosen by the public-coin verifier.

For illustration, we will use the description of the QAP language due to Ben-Sasson *et al.* [BCG+13, §E.1]. This language is defined by length $m+1$ polynomial vectors $A(X)$, $B(X)$, $C(X) \in (\mathbb{F}[X])^{m+1}$ such that the ith components $A_i(X)$, $B_i(X)$, $C_i(X)$ are all degree-$(n-1)$ polynomials over $\mathbb{F}_p[X]$ for $i \in [0, m-1]$, and $A_m = B_m = C_m$ is the degree-n polynomial $Z(X)$ that vanishes on a specified set of n distinct points in $\mathbb{F}_p$. There is a length-$(m-1)$ witness vector $\mathbf{w}$ whose first ℓ components are equal to the instance $\mathbf{x} \in \mathbb{F}^\ell$, and a degree-$n$ "quotient" polynomial $H(X)$, such that the following constraint equation is satisfied:

$$\begin{aligned}[(1, \mathbf{w}^\top, \delta_1)A(X)] \cdot [(1, \mathbf{w}^\top, \delta_2)B(X)] - (1, \mathbf{w}^\top, \delta_3)C(X) = H(X) \cdot Z(X) \\ \text{and } (1, \mathbf{w}^\top)(1, X, ..., X^\ell, \mathbf{0}^{m-\ell-1}) = (1, \mathbf{x}^\top)(1, X, ..., X^\ell)\end{aligned} \tag{2}$$

The deltas $\delta_1, \delta_2, \delta_3 \in \mathbb{F}$ are used as randomizers for zero-knowledge.

QAP algebraic linear PCP Equation 2 is turned into a set of linear queries by evaluating the polynomials at a random point in $\mathbb{F}$. Satisfaction in this random point implies satisfaction of the polynomial equation with error at most $2n/|\mathbb{F}|$ by the Schwartz-Zippel lemma. Translated to an algebraic IOP, the prover sends a proof oracle $\boldsymbol{\pi}_w$ containing the vector $(1, \mathbf{w}, \delta_1, \delta_2, \delta_3)$ as well as a proof oracle $\boldsymbol{\pi}_h$ containing the coefficient vector of $H(X)$. A common proof oracle $\boldsymbol{\pi}_z$ is jointly established containing the coefficient vector of $Z(X)$.

Let $\alpha \in \mathbb{F}$ be a random point. The verifier makes four queries to $\boldsymbol{\pi}_w$, computed by the polynomial vectors $A(X), B(X), C(X)$ and $D(X) = (1, X, ..., X^\ell, \mathbf{0}^{m-\ell-1})^\top$, evaluated in α. The verifier makes one query each to $\boldsymbol{\pi}_h$ and $\boldsymbol{\pi}_z$, which is the evaluation of $H(\alpha)$ and $Z(\alpha)$ respectively. The verifier obtains query responses $y_a, y_b, y_c, y_d, y_h, y_z$ and checks that $y_a \cdot y_b - y_c = y_h y_z$ and $y_d = \langle (1, \mathbf{w}^\top), D(\alpha) \rangle$.

QAP Polynomial IOP Following the compilation in Theorem 7 (Section 5.2.2), the QAP algebraic linear PCP can be transformed into a 2-round Polynomial IOP. For simplicity, assume $m + 3 < n$, where $m - 1$ is the length of the witness and n is the multiplicative complexity of the circuit. The preprocessing establishes three bivariate degree-n polynomials (*i.e.*, encoding $A(X), B(X), C(X)$) and two univariate degree-n polynomials (*i.e.*, encoding $Z(X)$ and $D(X)$). In the 2-round online phase the prover sends a degree-n univariate oracle for the witness vector $(1, \mathbf{w}, \delta_1, \delta_2, \delta_3)$, a degree-$n$ univariate oracle for $H(X)$, four degree-n univariate oracles encoding linear PCP queries, four degree-$2n$ univariate oracles encoding polynomial products, and eight degree-$2n$ univariate oracles for opening inner products. The total number of polynomial oracle evaluation queries is 3 bivariate degree-n, 8 univariate degree-$2n$, and 7 univariate degree-n.

Theorem 16 (QAP Polynomial IOP)**.** There exists a 2-round Polynomial IOP with preprocessing for any NP relation $\mathcal{R}$ (with multiplicative complexity n) that makes 7 queries to univariate degree-n oracles, 8 queries to univariate degree-$2n$ oracles, and 3 queries to bivariate degree-n oracles.

While theoretically intriguing, compiling the QAP-based IOP with our polynomial commitments of Section 4 is less practical than compiling the Sonic IOP. While the QAP Polynomial IOP has only 15 univariate queries (compared to Sonic's 39 queries to polynomials of twice the degree), the 3 bivariate polynomial oracles take quadratic time to preprocess and open. Unfortunately, our polynomial commitment scheme does not take advantage of the sparsity of these bivariate polynomials. Furthermore, ignoring prover time complexity, the size of the bivariate Eval proofs are twice as large as univariate Eval proofs.

6 Evaluation

We now evaluate Supersonic, the trustless-setup SNARK built on the Polynomial IOPs underlying Sonic [MBKM19], PLONK [GWC19], and Marlin [CHM+19], and compiled using our DARK polynomial commitment scheme. As explained in Section 4.5, the commitment scheme has several batching properties that can be put to good use here. It is possible to evaluate k polynomials of degree at most d using only 2 group elements and $(k+1)$ field elements. To take advantage of this we delay the evaluation until the last step of the protocol (see Section 5.3). We present the proof size for both the compilation of Sonic, PLONK and Marlin in Table 3. We use 1600 bits as the size of class group elements and $\lambda = 120$. The

security of 1600 bit class groups is believed to be equivalent to 3048bit RSA groups and have 120 bits of security [BH01, BJS10]. This leads to proof sizes of 16.5KB for Sonic, 10.1KB using PLONK and 12.3KB using Marlin for circuits with $n = 2^{20}$ (one million) gates. Using 3048-bit RSA groups the proof sizes becomes 18.4KB for the compilation of PLONK. If 100 bits of security suffice then a 1200 bit class group can be used and the compiled PLONK proofs are 7.8KB for the same setting. In a 2048-bit RSA group this becomes 12.7KB.

The comparison between the Polynomial IOPs is slightly misleading because for Sonic n is the number of multiplication gates whereas for PLONK it is the sum of multiplication and addition gates. For Marlin it is the number of non-zero entries in the R1CS description of the circuit. A more careful analysis is therefore necessary, but this shows that there are Polynomial IOPs that can be compiled using the DARK polynomial commitment scheme to SNARKs of roughly 10 kilobytes in size. These numbers stand in contrast to STARKs which achieve proofs of 600KB for computation of similar complexity [BBHR19]. We compare Supersonic to different other proof systems in Table 4. Supersonic is the only proof system with efficient verifier time, small proof sizes that does not require a trusted setup.

Polynomial IOP	Polynomials	Eval points	|SNARK|	concrete size
Sonic [MBKM19]	12 in pp + 15	12	$(15 + 2\log_2(n))\mathbb{G}$ $+(12 + 13\log_2(n))\mathbb{Z}_p$	15.3 KB
PLONK [GWC19]	7 in pp + 7	2	$(7 + 2\log_2(n))\mathbb{G}$ $+ (2 + 3\log_2(n))\mathbb{Z}_p$	10.1 KB
Marlin [CHM+19]	9 in pp + 10	3	$(10 + 2\log_2(6n))\mathbb{G}$ $+ (3 + 4\log_2(6n))\mathbb{Z}_p$	12.3 KB

Table 3: Proof size for Supersonic. Column 2 says how many polynomials are committed to in the SRS (offline oracles) and how many are sent by the prover (online oracles). Column 3 states the number of distinct evaluation points. The proof size calculation uses $|\mathbb{Z}_p| = 120$ and $|\mathbb{G}| = 1600$ for $n = 2^{20}$ gates.

Prover and Verifier cost We use the notation $O_\lambda(\cdot)$ to denote asymptotic complexity for a fixed security parameter λ, *i.e.* how the prover and verifier costs scale as a function of variables other than λ. The main cost for the Supersonic prover consist of computing the commitments to the polynomial oracles and producing the single combined Eval proof. This proof requires calculating the commitments to the polynomials $f_L(q)$ and $f_R(q)$ in each round and performing the $\mathsf{PoE}(\mathsf{C}_R, \mathsf{C}/\mathsf{C}_L, q^{d'+1})$. Using precomputation, *i.e.*, computing g^{q^i} for all i and using multi-exponentiation, the commitments can be computed in $O_\lambda(\frac{d}{\log(d)})$ group operations. The same techniques can be used to reduce the number of group operations for the PoEs to $O_\lambda(d)$. The total number of group operations is therefore linear in the maximum degree of the polynomial oracles and the number of online oracles. Interestingly, the number of offline oracles hardly impacts the prover time and proof size.

The verifier time is dominated by the group operations for exponentiation in various places in the single combined Eval protocol. It consists of 3 λ-bit exponentiations in each round: 1 for combining C_L and C_R and two for verifying the PoE. In the final round the verifier does another $\lambda\log_2(d+1)$-bit exponentiation to open the commitment but this could also be outsourced to the prover using yet another PoE. The total verifier time therefore consists of roughly an exponentiation of $3\lambda\log_2(d+1)$ group operations. Using 10μs per group operation[10], this gives us for $\lambda = 120$ and $n = 2^{20}$ a verification time of around 72ms.

[10]The estimate comes from the recent Chia Inc. class group implementation competition. The competition used a larger 2048bit discriminant but only performed repeated squaring. `https://github.com/Chia-Network/vdfcontest2results`

Scheme	Transp.	$\|\mathsf{pp}\|$	Prover	Verifier	$\|\pi\|$	$n = 2^{20}$
Supersonic	yes	$O(1)$	$O(n \log(n))$ EXP	$3\log(n)$ EXP	$2\log(n)$ $\mathbb{G}_U$	10.1KB
PLONK [GWC19]	no	$2n$ $\mathbb{G}_B$	$O(n)$ EXP	1 Pairing	$O(1)$ $\mathbb{G}_B$	720b
Groth16 [Gro16]	no	$2n$ $\mathbb{G}_B$	$O(n)$ EXP	1 Pairing	$O(1)$ $\mathbb{G}_B$	192b
BP [BBB+18]	yes	$2n$ $\mathbb{G}_P$	$O(n)$ EXP	$O(n)$ EXP	$2\log(n)$ $\mathbb{G}_P$	1.7KB
STARK	yes	$O(1)$	$O(\lambda T)$ H	$O(\lambda \log^2(T))$ H	$O(\lambda \log^2(T))$ H	600 KB
Virgo[ZXZS19]	yes	$O(1)$	$O(\lambda n)$ H	$O(\lambda \log^2(n))$ H	$O(\lambda \log^2(n))$ H	271 KB

Table 4: Comparison table between different succinct arguments. In column order we compare on transparent setup, CRS size, prover and verifier time, asymptotic proof size and concrete proof for an NP relation with arithmetic complexity 2^{20}. Even when precise factors are given the numbers should be seen as estimates. For example, we chose to not display smaller order terms. The symbol $\mathbb{G}_U$ denotes an element in group of unknown order, $\mathbb{G}_B$ one in a group with a bilinear map (pairing), $\mathbb{G}_P$ one in a prime order group with known order. Furthermore, EXP refers to exponentiation of λ-bit numbers in these groups, and H is either the size of a hash output or the time it takes to compute a hash. The prover time for the group based schemes can be brought down by a log factor when using multi-exponentiation techniques.

7 Conclusion

In this work we presented the DARK compiler: a polynomial commitment scheme from falsifiable assumptions in groups of unknown order with evaluation proofs that can be verified in logarithmic time. We also presented Polynomial IOPs, a unifying information-theoretical framework underlying the information theoretic foundation of several recent SNARK constructions. Polynomial IOPs can be compiled into a concrete SNARK using a polynomial commitment scheme and the Fiat-Shamir transform. We showed that applying the DARK compiler to recent Polynomial IOPs yields **the first trustless SNARKs (*i.e.*, with a transparent untrusted setup) that have practical proof sizes and verification times**. In particular, this is the first trustless/transparent SNARK construction that has asymptotically logarithmic verification time (ignoring the λ-dependent factors, which are comparable to λ-dependent factors in prior works). Finally, unlike all known SNARKs in bilinear groups, the construction does not require knowledge of exponent assumptions. Several important open questions remain:

- Our polynomial commitment scheme has prover time linear in the total number of coefficients, even for zero coefficients. Consequently for a sparse bivariate polynomial of degree d in each variable the prover time is quadratic in d. A sparse polynomial commitment scheme would directly enable an efficient compilation of simple information theoretic protocols such as QAPs.

- Assymptotically, Supersonic's prover time is on par with pairing-based SNARK constructions, however, a concrete implementation and performance comparison remains open.

- This work further motivates the study of class groups and groups of unknown order. In particular we rely on a recently introduced Adaptive Root Assumption.

- Our polynomial commitment scheme uses a simple underlying information theoretic protocol that could be compiled using a (partially) homomorphic commitment scheme over polynomials, or even another type of integer homomorphic commitment scheme. This leaves open whether there are different ways of instantiating our DARK compiler under different cryptographic assumptions.

Acknowledgements

We thank Dan Boneh for helpful discussions and comments. This work was partially supported by NSF, SGF, ONR, the Simons Foundation, the Nervos Foundation and the Findora Foundation.

References

[ALM+92] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. In *33rd FOCS*, pages 14–23. IEEE Computer Society Press, October 1992.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.

[BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 757–788. Springer, Heidelberg, August 2018.

[BBC+19] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear PCPs. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 67–97. Springer, Heidelberg, August 2019.

[BBF19] Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 561–586. Springer, Heidelberg, August 2019.

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl, July 2018.

[BBHR19] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 701–732. Springer, Heidelberg, August 2019.

[BCC+16a] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. Cryptology ePrint Archive, Report 2016/263, 2016. `http://eprint.iacr.org/2016/263`.

[BCC+16b] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357. Springer, Heidelberg, May 2016.

[BCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108. Springer, Heidelberg, August 2013.

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014.

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333. Springer, Heidelberg, March 2013.

[BCR+19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Heidelberg, May 2019.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Heidelberg, October / November 2016.

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *23rd ACM STOC*, pages 21–31. ACM Press, May 1991.

[BG10] Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 1–20. Springer, Heidelberg, August 2010.

[BGG+88] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 37–56. Springer, Heidelberg, August 1988.

[BGKS19] Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. DEEP-FRI: sampling outside the box improves soundness. *IACR Cryptology ePrint Archive*, 2019:336, 2019.

[BH01] Johannes Buchmann and Safuat Hamdy. A survey on iq cryptography. In *Public-Key Cryptography and Computational Number Theory*, pages 1–15, 2001.

[BJS10] Jean-François Biasse, Michael J. Jacobson Jr., and Alan K. Silvester. Security estimates for quadratic field based cryptosystems. *CoRR*, abs/1004.5512, 2010.

[Bow16] Sean Bowe. Bellman zk-snarks library, 2016. `https://github.com/zkcrypto/bellman`.

[BP97] Niko Bari and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 480–494. Springer, Heidelberg, May 1997.

[BS96] Wieb Bosma and Peter Stevenhagen. On the computation of quadratic 2-class groups. In *Journal de Theorie des Nombres*, 1996.

[BSGKS19] Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. Deep-fri: Sampling outside the box improves soundness. 26:44, 2019.

[BTVW14] Andrew J. Blumberg, Justin Thaler, Victor Vu, and Michael Walfish. Verifiable computation using multiple provers. Cryptology ePrint Archive, Report 2014/846, 2014. `http://eprint.iacr.org/2014/846`.

[But16] Vitalik Buterin. Zk rollup, 2016. `https://ethresear.ch/t/on-chain-scaling-to-potentially-500-tx-sec-through-mass-tx-validation/3477`.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *51st ACM STOC*, pages 1082–1090. ACM Press, June 2019.

[CCRR18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. Fiat-Shamir and correlation intractability from strong KDM-secure encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 91–122. Springer, Heidelberg, April / May 2018.

[CFS17] Alessandro Chiesa, Michael A. Forbes, and Nicholas Spooner. A zero knowledge sumcheck and its applications. Cryptology ePrint Archive, Report 2017/305, 2017. `http://eprint.iacr.org/2017/305`.

[CHM+19] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zksnarks with universal and updatable srs. Cryptology ePrint Archive, Report 2019/1047, 2019. `https://eprint.iacr.org/2019/1047`.

[COS19] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography, 2019.

[CPP17] Geoffroy Couteau, Thomas Peters, and David Pointcheval. Removing the strong RSA assumption from arguments over the integers. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 321–350. Springer, Heidelberg, April / May 2017.

[CS99] Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. In Juzar Motiwalla and Gene Tsudik, editors, *ACM CCS 99*, pages 46–51. ACM Press, November 1999.

[DF02] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 125–142. Springer, Heidelberg, December 2002.

[DK02] Ivan Damgård and Maciej Koprowski. Generic lower bounds for root extraction and signature schemes in general groups. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 256–271. Springer, Heidelberg, April / May 2002.

[Ebe] Jacob Eberhardt. Zokrates. `https://zokrates.github.io/`.

[FO97] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 16–30. Springer, Heidelberg, August 1997.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013.

[GI08] Jens Groth and Yuval Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 379–396. Springer, Heidelberg, April 2008.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 113–122. ACM Press, May 2008.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity. *Journal of the ACM*, 38(3), 1991.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

[GVW02] Oded Goldreich, Salil Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. volume 11(1/2), pages 1–53, 2002.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. `https://eprint.iacr.org/2019/953`.

[HBHW19] Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash Protocol Specification, 2019.

[IKO07] Yuval Ishai, Eyal Kushilevitz, and Rafael Ostrovsky. Efficeint arguments without short pcps. 2007.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992.

[KPV19] Assimakis Kattis, Konstantin Panarin, and Alexander Vlasov. RedShift: Transparent SNARKs from list polynomial commitment IOPs. Cryptology ePrint Archive, Report 2019/1400, 2019. `https://eprint.iacr.org/2019/1400`.

[KRR17] Yael Tauman Kalai, Guy N. Rothblum, and Ron D. Rothblum. From obfuscation to the security of Fiat-Shamir for proofs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 224–251. Springer, Heidelberg, August 2017.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Heidelberg, December 2010.

[Lab18] O(1) Labs. Coda protocol, 2018. `https://codaprotocol.com/`.

[Lin01] Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 171–189. Springer, Heidelberg, August 2001.

[Lip03] Helger Lipmaa. On diophantine complexity and statistical zero-knowledge arguments. In Chi-Sung Laih, editor, *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 398–415. Springer, Heidelberg, November / December 2003.

[LM19] Russell W. F. Lai and Giulio Malavolta. Subvector commitments with application to succinct arguments. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 530–560. Springer, Heidelberg, August 2019.

[LRY16] Benoît Libert, Somindu C. Ramanna, and Moti Yung. Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *ICALP 2016*, volume 55 of *LIPIcs*, pages 30:1–30:14. Schloss Dagstuhl, July 2016.

[MBKM19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings. Cryptology ePrint Archive, Report 2019/099, 2019. `https://eprint.iacr.org/2019/099`.

[Pie19] Krzysztof Pietrzak. Simple verifiable delay functions. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 60:1–60:15, 2019.

[Pip80] Nicholas Pippenger. On the evaluation of powers and monomials. *SIAM Journal on Computing*, 9:230–250, 1980.

[RRR16] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 49–62. ACM Press, June 2016.

[RSW96] Ron Rivest, Adi Shamir, and David Wagner. Time-lock puzzles and timed-release crypto. In *MIT Technical report*, 1996.

[SBV+13] S. Setty, B. Braun, V. Vu, A. J. Blumberg, B. Parno, and M. Walfish. Resolving the conflict between generality and plausibility in verified computation. 2013.

[Set19] Srinath Setty. Spartan: Efficient and general-purpose zksnarks without trusted setup. Cryptology ePrint Archive, Report 2019/550, 2019. `https://eprint.iacr.org/2019/550`.

[Str19] Michael Straka. Class groups for cryptographic accumulators. 2019.

[WB15] Michael Walfish and Andrew J. Blumberg. Verifying computations without reexecuting them: From theoretical possibility to near practicality. *Communications of the ACM*, 58(2), 2015.

[Wee05] Hoeteck Wee. On round-efficient argument systems. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP 2005*, volume 3580 of *LNCS*, pages 140–152. Springer, Heidelberg, July 2005.

[Wes19] Benjamin Wesolowski. Efficient verifiable delay functions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 379–407. Springer, Heidelberg, May 2019.

[Wil16] Z. Wilcox. The design of the ceremony, 2016. `https://z.cash/blog/the-design-of-the-ceremony.html`.

[WTs+18] Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *2018 IEEE Symposium on Security and Privacy*, pages 926–943. IEEE Computer Society Press, May 2018.

[XZZ+19] Tiacheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. Cryptology ePrint Archive, Report 2019/317, 2019. `https://eprint.iacr.org/2019/317`.

[Zca] Zcash. `https://z.cash`.

[ZGK+17] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. vSQL: Verifying arbitrary SQL queries over dynamic outsourced databases. In *2017 IEEE Symposium on Security and Privacy*, pages 863–880. IEEE Computer Society Press, May 2017.

[ZXZS19] Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. Transparent polynomial delegation and its applications to zero knowledge proof. Cryptology ePrint Archive, Report 2019/1482, 2019. `https://eprint.iacr.org/2019/1482`.

Appendix

A Security Proofs

In the preliminaries we already stated the two main hardness assumptions, the r-Strong RSA Assumption and the Adaptive Root Assumption. We additionally use two more assumptions, the *Order Assumption*, and the *r-Fractional Root Assumption*. However, both of them reduce to the Strong RSA and the Adaptive Root Assumptions.

The first assumption states that computing the order for *any* element is hard. It reduces to the Adaptive Root Assumption. Interestingly, it doesn't necessarily hold for all candidate groups of unknown order as we explain below. In particular it is important to exclude elements of known order such as -1 from the candidate unknown order group $\mathbb{Z}_n$.

Assumption 3 (Order Assumption)**.** The Order Assumption holds for *GGen* if for any efficient adversary $\mathcal{A}$:

$$\Pr\left[\mathsf{w} \neq 1 \wedge \mathsf{w}^{\alpha} = 1 : \begin{array}{l} \mathbb{G} \xleftarrow{\$} GGen(\lambda) \\ (\mathsf{w}, \alpha) \xleftarrow{\$} \mathcal{A}(\mathbb{G}) \\ \text{where } |\alpha| < 2^{\mathsf{poly}(\lambda)} \in \mathbb{Z} \\ \text{and } \mathsf{w} \in \mathbb{G} \end{array}\right] \leq \mathsf{negl}(\lambda) \quad .$$

Lemma 5. The Adaptive Root Assumption implies the Order Assumption.

Proof. We show that given an adversary $\mathcal{A}_{\mathsf{Ord}}$ that breaks the Order Assumption we can construct with overwhelming probability $\mathcal{A}_{\mathsf{AR}}$ that breaks the Adaptive Root Assumption. We run $\mathcal{A}_{\mathsf{Ord}}$ to get a $\mathsf{w} \neq 1 \in \mathbb{G}$ and $\alpha \in \mathbb{Z}$ such that $\mathsf{w}^{\alpha} = 1$. To construct $\mathcal{A}_{\mathsf{AR}}$, $\mathcal{A}_{\mathsf{AR},0}$ outputs (w, α). The challenger generates a random challenge ℓ. If $\gcd(\ell, \alpha) = 1$ then $\mathcal{A}_{\mathsf{AR},1}$ can compute $\beta \leftarrow \ell^{-1} \bmod \alpha$ and output $\mathsf{u} \leftarrow \mathsf{w}^{\beta}$. By construction $\mathsf{u}^{\ell} = \mathsf{w}$. The probability that $\gcd(\ell, \alpha) = 1$ is overwhelming because $\gcd(\ell, \alpha) \neq 1 \implies \ell | \alpha$. This happens with negligible probability as ℓ is picked from a set of 2^{λ} primes and at most $\mathsf{poly}(\lambda)$ distinct primes can divide α. □

We also define the Fractional Root Assumption, which states that for random group elements g it is hard to find a tuple $(\mathsf{u} \in \mathbb{G}, \alpha \in \mathbb{Z}, \beta \in \mathbb{Z})$ such that $\mathsf{u}^{\beta} = \mathsf{g}^{\alpha}$. We say that $(\mathsf{u}, \alpha, \beta)$ is a *fractional root* of g. Shoup[CS99] showed that for the unknown order group of quadratic residues in $\mathbb{Z}_n$, where n is the composite of two strong primes, that the Fractional Root Assumption reduces to just the Strong RSA Assumption.

Assumption 4 (r-Fractional Root Assumption)**.** The **r-Fractional Root Assumption** holds for *GGen* for any efficient adversary $\mathcal{A}$:

$$\Pr\left[\mathsf{u}^{\beta} = \mathsf{g}^{\alpha} \wedge \frac{\beta}{\gcd(\alpha, \beta)} \neq r^{k}, k \in \mathbb{N} : \begin{array}{l} \mathbb{G} \xleftarrow{\$} GGen(\lambda) \\ \mathsf{g} \xleftarrow{\$} \mathbb{G} \\ (\alpha, \beta, \mathsf{u}) \xleftarrow{\$} \mathcal{A}(\mathbb{G}, \mathsf{g}) \\ \quad \text{where } |\alpha| < 2^{\mathsf{poly}(\lambda)}, \\ \quad |\beta| < 2^{\mathsf{poly}(\lambda)} \in \mathbb{Z}, \\ \quad \text{and } \mathsf{u} \in \mathbb{G} \end{array}\right] \leq \mathsf{negl}(\lambda) \quad .$$

We say $(\alpha, \beta, \mathsf{u})$ is a non power of r fractional root of g.

The Fractional Root Assumption reduces to the Order Assumption (and therefore to the Adaptive Root Assumption) and the Strong RSA Assumption.

Lemma 6. The Adaptive Root Assumption and the r-Strong RSA Assumption imply the r-Fractional Root Assumption

Proof. Given an adversary $\mathcal{A}_{\mathsf{FR}}$ that succeeds in breaking the Fractional Root Assumption for *GGen* we can construct either an adversary $\mathcal{A}_{RSA}$ for the Strong RSA Assumption or an adversary $\mathcal{A}_{\mathsf{Ord}}$ that breaks the Order Assumption for *GGen*. As shown in Lemma 5 the Order Assumption reduces to the Adaptive Root Assumption with overwhelming probability. We first generate a group of unknown order $\mathbb{G} \xleftarrow{\$} GGen(\lambda)$. Then we sample $\mathsf{g} \xleftarrow{\$} \mathbb{G}$ as done in the strong RSA security definition.

We now run the $\mathcal{A}_{\mathsf{FR}}$ on input $\mathbb{G}$ and g to generate a tuple $(\alpha, \beta, \mathsf{u})$ such that $\mathsf{u}^\beta = \mathsf{g}^\alpha$. Let $\gamma = \gcd(\alpha, \beta)$ and $\alpha' = \frac{\alpha}{\gamma} \in \mathbb{Z}$ and $\beta' = \frac{\beta}{\gamma} \in \mathbb{Z}$. Now either $\mathsf{g}^{\alpha'} = \mathsf{u}^{\beta'}$ or $\mathsf{g}^{\alpha'}/\mathsf{u}^{\beta'}$ is a non trivial element of order γ which would directly break the Order Assumption. In that case we constructed $\mathcal{A}_{\mathsf{Ord}}$ that outputs $(\mathsf{g}^{\alpha'}/\mathsf{u}^{\beta'}, \gamma)$.

Now assume otherwise, i.e. $\mathsf{g}^{\alpha'} = \mathsf{u}^{\beta'}$. By construction $\gcd(\alpha', \beta') = 1$ and we can efficiently compute integers a, b such that $a\alpha' + b\beta' = 1$. By assumption on $\mathcal{A}_{\mathsf{FR}}$ β' is not r^k. Now let $\mathsf{w} \leftarrow \mathsf{u}^a\mathsf{g}^b$. Note that $\mathsf{w}^{\alpha'\beta'} = \mathsf{g}^{\alpha'}$. So either $\mathsf{w}^{\beta'} = \mathsf{g}$ or $\mathsf{w}^{\beta'}/\mathsf{g}$ is a non-trivial element of order α'. The first case breaks the Strong RSA Assumption, as we can construct $\mathcal{A}_{\mathsf{RSA}}$ that outputs (w, β'), and the second breaks the Order Assumption. □

A.1 Binding

Lemma 3. The polynomial commitment scheme is binding for polynomials in $\mathbb{Z}(b)[X]$ for $b < q/2$ if either the Adaptive Root Assumption or the Strong RSA Assumption hold.

Proof. Assume that there is an adversary that breaks the binding property of the scheme. Specifically, assume that some probabilistic polynomial time algorithm $\mathcal{A}$ takes as input pp and outputs $\mathsf{C} \in \mathbb{G}, f(X) \in \mathbb{Z}(b)[X], f'(X) \in \mathbb{Z}(b)[X]$ such that with non-negligble probability $\mathsf{Open}(\mathsf{pp}, \mathsf{C}, \tilde{f}(X), f(X)) = \mathsf{Open}(\mathsf{pp}, \mathsf{C}, \tilde{f}'(X), f'(X)) = 1$ and $\tilde{f}(X) \neq \tilde{f}'(X)$. We proceed to show that this implies a violation of the Order Assumption (Assumption 3) and the Strong RSA Assumption (Assumption 1). The assumptions are incomparable so we show that either suffices to achieve the binding property of the commitment scheme.

If $f(X) \neq f'(X)$ and $q/2 > b$ then $f(q) \neq f'(q) \in \mathbb{Z}$. Since $\mathsf{g}^{f(q)} = \mathsf{g}^{f'(q)} = \mathsf{C}$ we have that $\mathsf{g}^{f(q)-f'(q)} = 1$. This directly breaks the Order Assumption and we can also create an adversary $\mathcal{A}_{RSA}$ that breaks the Strong RSA Assumption. To do so the $\mathcal{A}_{RSA}$ picks an odd prime ℓ that is co-prime with $f(q) - f'(q)$ and computes $\mathsf{u} \leftarrow \mathsf{g}^{\ell^{-1} \bmod (f(q)-f'(q))}$ as the ℓth root of g. □

A.2 Correctness

Lemma 4. The polynomial commitment scheme is correct for polynomials in $\mathbb{Z}_p[X]$ of degree at most d if $q > p^{\lceil \log_2(d+1) \rceil + 1}$.

Proof. In order to ensure correctness we must ensure that $b < q/2$ and that $|f| \leq b$. To show this we show that in each recursion step the honest prover's witness polynomial has coefficients bounded by b and has degree d. We argue inductively that for each recursive call of $\mathsf{EvalBounded}$ the following constraints on the inputs are satisfied: The degree of $f(X)$ is bounded by d. C encodes the polynomial, *i.e.*, $\mathsf{C} = \mathsf{g}^{f(q)}$ and $f(X) \in \mathbb{Z}(b)$. Also $f(z) = y \bmod p$.

Initially, during the execution of Eval, the prover maps the coefficients of a polynomial $\tilde{f}(X) \in \mathbb{Z}_p$ to an integer polynomial $f(X)$ with coefficients in $\mathbb{Z}(\frac{p-1}{2})$ and degree at most d such that $\mathsf{C} = \mathsf{g}^{f(q)}$. Additionally $f(z) \bmod p = \tilde{f}(z) = y$.

In a recursion steps where $d+1$ is odd, $f'(X) = X \cdot f(X)$ is a polynomial of degree $d+1$ such that $\mathsf{C}' = \mathsf{C}^q = \mathsf{g}^{q \cdot f(q)} = \mathsf{g}^{f'(X)}$ and the bound b is unchanged as are the coefficients. Also, $f'(z) \bmod p = z \cdot f(z) \bmod p = z \cdot y \bmod p = y' \bmod p$. If $d+1$ is odd, then in the next step $d+1$ must be even.

If $d+1$ is even then, $\mathcal{P}$ computes $f_L(X)$ and $f_R(X)$ such that $f_L(X) + X^{\frac{d+1}{2}} f_R(X) = f(X)$. Consequently $f(z) \bmod p = f_L(z) + z^{\frac{d+1}{2}} f_R(z) \bmod p = y_L + z^{\frac{d+1}{2}} y_R \bmod p = y$. The PoE protocol has perfect correctness so $\mathsf{g}^{f_L(q) + q^{\frac{d+1}{2}} f_R(X)} = \mathsf{C}$. Finally $f'(X) = \alpha f_L(X) + f_R(X) \in \mathbb{Z}(\frac{p+1}{2} \cdot b)$ is a degree d polynomial with coefficients bounded in absolute value by $(\frac{p+1}{2}) \cdot b$. This is precisely the value of b' the input to the next call of $\mathsf{EvalBounded}$. The value y' is also correct: $f'(z) \bmod p = \alpha f_L(z) + f_R(z) \bmod p = \alpha y_L + y_R \bmod p = y'$

There are exactly $\lceil \log_2(d+1) \rceil$ recursion steps with even $d+1$. In the final recursion step we therefore have $b = \frac{p-1}{2}(\frac{p+1}{2})^{\lceil \log_2(d+1) \rceil}$ and as such the requirement that $q/2 > \frac{p-1}{2}(\frac{p+1}{2})^{\lceil \log_2(d+1) \rceil}$. So if $q > p^{\lceil \log_2(d+1) \rceil + 1} \geq (p-1)(\frac{p+1}{2})^{\lceil \log_2(d+1) \rceil}$ then all verifier checks pass and the verifier outputs 1. □

A.3 Proof of Theorem 1

Security of PoE substitutions We first begin by showing that we can safely replace all of the PoE evaluations with direct verification checks. Concretely, under the Adaptive Root

Assumption, the Eval protocol is as secure as the protocol Eval' in which all PoEs are replaced by direct checks. We show that the witness-extended emulation for Eval' implies the same property for Eval. This is useful because we will later show how to can build an extractor for Eval', thereby showing that the same witness-extended emulation property extends to Eval.

Lemma 7. Let Eval' be the protocol that is identical to Eval but in line 18 of $\mathsf{EvalBounded}\ \mathcal{V}$ directly checks $\mathsf{C}_L\mathsf{C}_R^{q^{d'+1}} = \mathsf{C}$ instead of using a PoE. If the Adaptive Root Assumption holds for $GGen$, and Eval' has witness-extended emulation for polynomials of degree $d = \mathsf{poly}(\lambda)$, then so does Eval.

Proof. We show that if an extractor E', as defined in Definition 2, exists for the protocol Eval' then we can construct an extractor E for the protocol Eval. Specifically, E simulates E' and presents it with a $\mathsf{Record}'(\cdots)$ oracle, while extracting the witness from its own $\mathsf{Record}(\cdots)$ oracle.

Whenever E' queries the Record' oracle, E queries its Record oracle and relays the response after dropping those portions of the transcript that correspond to the PoE proofs. Whenever E' rewinds its prover, so does E rewind its prover. When E' terminates by outputting a transcript-and-witness pair $(\mathsf{tr}', f(X))$, E adds PoEs into this transcript to obtain tr and outputs $(\mathsf{tr}, f(X))$.

For each PPT adversary $(\mathcal{A}, P^*)$, E will receive a polynomial number of transcripts from its Record oracle. Any transcript tr of Eval such that $\mathcal{A}(\mathsf{tr}) = 1$ and tr is accepting contains exactly $\lceil \log(d+1) \rceil$ $\mathsf{PoE}s$ transcripts. So in total E sees only a polynomial number of PoE transcripts generated by a probabilistic polynomial-time prover and verifier. By Lemma 2 under the Adaptive Root Assumption, the probability that a polynomial time adversary can break the soundness of PoE, *i.e.*, convince a verifier on an instance $(\mathsf{C}_R, \mathsf{C}/\mathsf{C}_L, q^{d'+1}) \notin \mathcal{R}_{\mathsf{PoE}}$, is negligible. Consequently, the probability that the adversary can break PoE on *any* of the polynomial number of executions of PoE is still negligible.

This means that with overwhelming probability all transcripts are equivalent to having the verifier directly check $(\mathsf{C}_R, \mathsf{C}/\mathsf{C}_L, q^{d'+1}) \in \mathcal{R}_{\mathsf{PoE}}$. By assumption, the witness-candidate $f(X)$ that E' outputs is a valid witness if the transcript tr' that E' also outputs is accepting. The addition of honest PoE transcripts to tr' preserves the transcript's validity. So tr is an accepting transcript for Eval if and only if tr' is an accepting transcript for Eval'. Therefore, E' outputs a valid witness $f(X)$ whenever E outputs a valid witness. This suffices to show that Eval has witness-extended emulation if Eval' has, and if the Adaptive Root Assumption holds for $GGen$. □

Combining statements. The Eval protocol combines two statements into one by using a random linear combination of group elements, *i.e.*, $\mathsf{C}' \leftarrow \mathsf{C}_L^{\alpha}\mathsf{C}_R$. We now show that this step is sound and that given the discrete logarithm for C' the extractor can extract the discrete logarithm for C_L and C_R we also show that the we can bound the size of the discrete logarithm. We show that this statement holds in two settings. First we consider a group $\mathbb{G}$ were the standard Strong RSA Assumption holds and group elements are encodings of integers. Then we will show that in groups in which taking square roots is easy we can extract dyadic rationals using the 2-Strong RSA Assumption.

Lemma 8 (Combining for integer witnesses)**.** For $\mathbb{G} \leftarrow GGen(\lambda)$, and $\mathsf{g} \xleftarrow{\$} \mathbb{G}$. Let $(z, \mathsf{C}_L, \mathsf{C}_R, y_L, y_R, \alpha, f, y)$ and $(z, \mathsf{C}_L, \mathsf{C}_R, y_L, y_R, \alpha', f', y')$ be two transcripts such that $\mathsf{g}^f = \mathsf{C}_L^{\alpha}\mathsf{C}_R$ and $\mathsf{g}^{f'} = \mathsf{C}_L^{\alpha'}\mathsf{C}_R$ for group elements $\mathsf{C}_L, \mathsf{C}_R \in \mathbb{G}$, and integers $\alpha, \alpha' \in [-\frac{p-1}{2}, \frac{p-1}{2}]$, $\alpha \neq \alpha'$. Further let $f, f' \in \mathbb{Z}$ be such that $f(X) \leftarrow \mathsf{Dec}(f)$ and $f'(X) \leftarrow \mathsf{Dec}(f')$ are degree d bounded polynomials with coefficients bounded by b, *i.e.*, $f(X), f'(X) \in \mathbb{Z}(b)[X] \subset \mathbb{Z}[X]$. And finally let $y = f(z) \bmod p$ and $y' = f'(z) \bmod p$. Then there exists a PPT algorithm $\mathcal{X}$ that given these transcripts computes either (1) $y_L, y_R \in \mathbb{Z}_p$, $f_L(X), f_R(X) \in \mathbb{Z}((p-1)\cdot b)[X]$ such that $f_L(z) = y_L \bmod p$ and $f_R(z) = y_R \bmod p$ or (2) an element in $\mathbb{G}$ of known order or (3) a fractional root of g.

Proof. Using the transcripts $\mathcal{X}$ computes $\Delta_\alpha \leftarrow \alpha - \alpha'$ and $\Delta_f \leftarrow f - f'$ such that $\mathsf{C}_L^{\Delta_\alpha} = \mathsf{g}^{\Delta_f}$. If $\frac{\Delta_f}{\Delta_\alpha}$ is not an integer then $\mathcal{X}$ outputs a fractional root of g, that is the tuple $(\Delta_f, \Delta_\alpha, \mathsf{C}_L)$. If $\frac{\Delta_f}{\Delta_\alpha}$ on the other hand is an integer then $\mathcal{X}$ can compute $\mathsf{D} \leftarrow \mathsf{g}^{\frac{\Delta_f}{\Delta_\alpha}}$. Either $\mathsf{D} = \mathsf{C}_L$ or $(\mathsf{D}/\mathsf{C}_L)^{\Delta_\alpha} = 1$. In the second case, D/C_L is an element of known order.

Otherwise $\mathsf{D} = \mathsf{C}_L$ and we have $\mathsf{C}_L = \mathsf{g}^{f_L}$ where $f_L = \frac{\Delta_f}{\Delta_\alpha}$ is an integer. Additionally $\mathsf{C}_R = \mathsf{g}^{f_R}$ for $f_R \leftarrow f - \alpha \cdot f_L$.

$\mathcal{X}$ now computes the corresponding polynomials $f_L(X) \leftarrow \mathsf{Dec}(f_L)$ and $f_R(X) \leftarrow \mathsf{Dec}(f_R)$. Now if for all i, the coefficients f_i and $f'_i \in [-b, b]$ and $\alpha, \alpha' \in [-\frac{p-1}{2}, \frac{p-1}{2}]$ then by the triangle inequality we have that for the ith coefficient of $f_L(X)$, $f_{L,i} \in [-2b, 2b]$. Additionally we have $f_{R,i} = \frac{f'_i\alpha - f_i\alpha'}{\Delta_\alpha}$. Using the triangle inequality again we have that $f_{R,i} \in [-(p-1)\cdot b, (p-1)\cdot b]$. For an odd prime p, $(p-1)\cdot p \geq 2$. The bound on $f_{R,i}$ is, therefore, greater than the bound on $f_{L,i}$. This gives us $f_L(X), f_R(X) \in \mathbb{Z}((p-1)\cdot b)[X]$

Let $y_L = \frac{y-y'}{\Delta_\alpha} \bmod p = \frac{f(z)-f'(z)}{\Delta\alpha} \bmod p$ and $y_R = y - \alpha\frac{y-y'}{\Delta_\alpha} \bmod p$. Since $f_L(X) = \frac{f(X)-f'(X)}{\Delta\alpha}$ this shows that $y_L = f_L(z) \bmod p$ and $y_R = f_R(z) \bmod p$. □

Theorem 1. The polynomial commitment scheme for polynomials in $\mathbb{Z}_p[X]$ of degree at most $d = \mathsf{poly}(\lambda)$, instantiated using $q > p^{2\lceil \log_2(d+1)\rceil + 1}$ and *GGen*, has witness extended emulation (Definition 2) if the Adaptive Root Assumption and the Strong RSA Assumption hold for *GGen*.

Proof. We will prove security by showing that given a polynomial time adversary $\mathcal{A}_{\mathsf{Eval}}$ that succeeds in convincing an honest verifier in the Eval protocol on any public input with non-negligible probability we can either (1) construct an adaptive root adversary $\mathcal{A}_{\mathsf{AR}}$, (2) extract an element of known order, and hence break the Order Assumption, (3) extract a fractional root of $\mathsf{g} \in \mathbb{G}$ or (4) extract the polynomial $f(X) \in \mathbb{Z}[X]$ such that $f(X)$ has degree at most d and the coefficients of $f(X)$ are integers bounded by $q/2$, such that $f(q)$ is a unique encoding of $f(X)$, $\mathsf{g}^{f(q)} = \mathsf{C}$, and $f(z) \bmod p = y$. The proof will use the general forking lemma (Lemma 1) to show that the polynomial commitment scheme has witness-extended emulation.

In particular we construct an extractor $\mathcal{X}$ that given transcripts with 2 distinct challenges per round, *i.e.*, $2^{\lceil \log_2(d+1)\rceil} < 2(d+1)$ transcripts in total, can compute either an opening to the commitment scheme, an element of known order, or a fractional root of $\mathsf{g} \in \mathbb{G}$ as encoded in the public parameters pp.

Using Lemma 7 and under the Adaptive Root Assumption, it suffices to consider an extractor $\mathcal{X}$ that works on transcripts of Eval' were all PoEs prove true statements. That is $\mathsf{C}_L\mathsf{C}_R^{q^{d'+1}} = \mathsf{C}$ on all transcripts.

Given a tree of Eval' transcripts, the extractor $\mathcal{X}$ recursively either extracts the encoding of an integer polynomial $f(X) \in \mathbb{Z}(b)[X] \subset \mathbb{Z}[X]$ with bounded coefficients or a break of the Order Assumption or the Fractional Root Assumption. In order to break the Order Assumption we instantiate the adversary $\mathcal{A}_{\mathsf{Ord}}$ with the description of the group $\mathbb{G}$. We also instantiate the fractional root adversary $\mathcal{A}_{\mathsf{FR}}$ with $\mathbb{G}$ and g as encoded in pp.

Given the tree of transcripts as specified in the general forking lemma (Lemma 1) with branching factor 2 at each level, *i.e.*, 2 different challenges, we will extract a witness at each node of the tree given witnesses for both nodes' children. Each level corresponds to a separate invocation to $\mathsf{EvalBounded}'$. We denote the input to Eval' without subscripts, *i.e.*, $\mathsf{C}, z, y, d; f(X)$, and the input to $\mathsf{EvalBounded}'$ with a subscript indicating the round, *e.g.*, $d_0 = d$, $\mathsf{C}_0 = \mathsf{C}$ and $d_{\lceil \log_2(d)\rceil} = 0, \mathsf{C}_{\lceil \log_2(d+1)\rceil} = \mathsf{g}^f$, *etc.* For the witness polynomials we use superscripts and parentheses, *i.e.*, $f^{(i)}(X)$ to avoid confusion with the notation for coefficients.

In each round the extracted witness is an integer polynomial $f^{(i)}(X) \in \mathbb{Z}[X]$ such that $\mathsf{g}^{f^{(i)}(q)} = \mathsf{C}_i$ and such that the coefficients are bounded, *i.e.*, all $f^{(i)}(X)$ are in $\mathbb{Z}(b_i)[X]$. The degree of $f^{(i)}(X)$ is at most d_i and $f(z) \equiv y \bmod p$. Note that for odd primes p and integer z, $f(z) \bmod p$ is always defined.

We extract starting from the leafs of the tree, *i.e.*, $d_{\lceil \log_2(d)\rceil} = 0$. From the transcript we can directly extract the constant integer polynomial $f(X) = f \in \mathbb{Z}$ such that $|f| \leq (p-1)(\frac{p+1}{2})^{\lceil \log_2(d+1)\rceil}$, $y = f \mod p$, $f(X) = y \in \mathbb{Z}_p[X]$ and $\mathsf{g}^f = \mathsf{C}$ as the witness.

We now show how to compute the witness for $i-1$ given a witnesses for i. If $d_i + 1$ is odd then we have $\mathsf{C}_{i-1}^q = \mathsf{C}_i$. Since $\mathsf{C}_{i-1} = \mathsf{g}^{f^{(i-1)}(q)}$ we either have that q divides $f^{(i-1)}(q)$ or since q is odd we have a fractional root of g. If this is not the case then $f^{(i-1)}(q) = f^{(i)}(q) \cdot q^{-1}$ and $f^{(i)}(X) = \mathsf{Dec}(f^{(i)}(q))$ has a zero constant term. Additionally since $y_i = y_{i-1} \cdot z$ and $f^{(i)}(z) \equiv y_i \bmod p$ we have $f^{(i-1)}(z) \equiv y_{i-1} \bmod p$, *i.e.*, $f^{(i-1)}(q)$ is a valid witness and the degree of $f^{(i-1)}(X) = \mathsf{Dec}(f^{(i-1)}(q))$ is at most $d_{i-1} = d_i - 1$.

Now if $d_i + 1$ is even then we can use Lemma 8 to either extract a fractional root of g, an element of known order in $\mathbb{G}$ or the two bounded polynomials $f_L^{(i)}(X), f_R^{(i)}(X)$ of degree

$\frac{d_i+1}{2} - 1$ and $y_L = f_L^{(i)}(z) \bmod p$ as well as $y_R = f_R^{(i)}(z) \bmod p$. This yields $f^{(i)}(X) = f_L^{(i)}(X) + X^{\frac{d_i+1}{2}} f_R^{(i)}(X)$ a polynomial of degree at most d_i such that $\mathsf{C}_i = \mathsf{g}^{f^{(i)}(q)}$ and such that $f^{(i)}(z) \bmod p = y_L + y_R \cdot z^{\frac{d_i+1}{2}} \bmod p = y_i$.

Note that the application of Lemma 8 requires that $q/2$ is greater than the magnitude of each of $f^{(i)}$'s coefficient. We will show now that this is the case.

The check on f ensures that $|f| \leq b = \frac{(p-1)}{2}(\frac{p+1}{2})^{\lceil \log_2(d+1) \rceil}$.

Lemma 8 in each invocation guarantees that the extracted parent polynomial has coefficients at most $(p-1)$ times larger than the coefficients of the children's polynomials. Given that the transcript tree has depth $\lceil \log_2(d+1) \rceil$ we get that the final extracted polynomial $f_0(X) \in \mathbb{Z}(b)[X]$ has coefficients bounded by $b = \frac{p-1}{2}(\frac{p^2-1}{2})^{\lceil \log_2(d+1) \rceil}$.

Therefore, q needs to be large enough such that $f_0(X)$ is uniquely decodable, *i.e.*, $q > 2 \cdot b = (p-1)(\frac{p^2-1}{2})^{\lceil \log_2(d+1) \rceil}$. This shows that $q > p^{2\lceil \log_2(d+1) \rceil + 1} > 2b$ suffices.

We can successfully extract either a witness or a fractional root or an element of known order from any tree of valid transcripts of Eval'. Under the Fractional Root Assumption and the Order Assumption, the probability that a polynomial time adversary along with a polynomial time extractor $\mathcal{X}$ can produce such a fractional root or an element of known order is negligible. Eval', therefore, has witness extended emulation and under the Adaptive Root Assumption by Lemma 7 so does Eval. Lemma 5 and Lemma 6 show that we can reduce the hardness assumptions to just the Adaptive Root Assumption and the Strong RSA Assumption.

□

A.4 Proof of Theorem 2

We begin by stating and proving the combining lemma, (Lemma 8) for dyadic rational witnesses.

Lemma 9 (Combining for Dyadic Rational Witnesses)**.** Let tr and tr' be two transcripts as specified in Lemma 8 with the difference that $f, f' \in \mathbb{D}$ are dyadic rationals such that $f(X) \leftarrow \mathsf{Dec}(f)$ and $f'(X) \leftarrow \mathsf{Dec}(f')$ are degree d bounded dyadic rational polynomials with coefficients' numerators bounded by N and denominators bounded by D, i.e. $f(X), f'(X) \in \mathbb{D}(N, D)$. Assume that there exists a PPT algorithm for taking square roots of any element in $\mathbb{G}$ and that the order of $\mathbb{G}$ is odd, then there exists a PPT algorithm $\mathcal{X}$ that given these transcripts computes either (1) $y_L, y_R \in \mathbb{Z}_p, f_L(X), f_R(X) \in \mathbb{D}(N \cdot (p-1), D \cdot (p-1))[X]$ such that $f_L(z) = y_L \bmod p$ and $f_R(z) = y_R \bmod p$ or (2) an element in $\mathbb{G}$ of known order or (3) a non-power of 2 fractional root of g.

Proof. The proof follows a similar structure to the proof of Lemma 8.

Using the transcripts we get $\Delta_\alpha \leftarrow \alpha - \alpha'$ and $\Delta_f \leftarrow f - f'$ such that $\mathsf{C}_L^{\Delta_\alpha} = \mathsf{g}^{\Delta_f}$. If $\frac{\Delta_f}{\Delta_\alpha}$ is not a dyadic rational then this gives us a non-power of 2 fractional root of g root of g, that is the tuple $(\Delta_f, \Delta_\alpha, \mathsf{C}_L)$. If $\frac{\Delta_f}{\Delta_\alpha}$ on the other hand is a dyadic rational then we can compute $\mathsf{D} \leftarrow \mathsf{g}^{\frac{\Delta_f}{\Delta_\alpha}}$. This may requires taking a power of 2 root. By assumption the group order is odd, so every element has a square root and there exists an efficient algorithm for taking square roots. This implies that taking higher power of 2 roots is also efficient.

Now either $\mathsf{D} = \mathsf{C}_L = \mathsf{g}^{f_L}$ or we can extract an element of known order. Additionally $\mathsf{C}_R = \mathsf{g}^{f_R}$ for $f_R \leftarrow f - \alpha \cdot f_L$.

We now compute the corresponding polynomials $f_L(X) \leftarrow \mathsf{Dec}(f_L)$ and $f_R(X) \leftarrow \mathsf{Dec}(f_R)$. Now if the coefficients f_i and $f_i' \in \mathbb{D}(N, D)$ and $\alpha, \alpha' \in [-\frac{p-1}{2}, \frac{p-1}{2}]$ then by the triangle inequality we have that for the numerator of the ith coefficient of $f_L(X)$ is between $[-2N, 2N]$. The denominator grows by at most $p-1$. The bound on the denominators is therefore $D \cdot (p-1)$. Additionally we have $f_{R,i} = \frac{f_i'\alpha - f_i\alpha'}{\Delta_\alpha}$. Using the triangle inequality again we have that the numerator of $f_{R,i} \in [-(p-1) \cdot N, (p-1) \cdot N]$. The denominator is bounded by $D \cdot (p-1)$. This gives us $f_L(X), f_R(X) \in \mathbb{D}((p-1) \cdot N, D \cdot (p-1))[X]$

Finally $y_L = f_L(z)$ and $y_R = f_R(z)$ as in Lemma 8. It is important that 2 is co-prime with the odd prime p such that each dyadic rational can be mapped to a field element. □

We now restate the theorem for the security of the protocol with dyadic rational witnesses in groups where taking square roots is easy.

Theorem 2. Let *GGen* generate groups $\mathbb{G}$ of unknown order such that the order of $\mathbb{G}$ is odd, and such there exists a PPT algorithm for taking square roots in $\mathbb{G}$. The polynomial commitment scheme for polynomials in $\mathbb{Z}_p[X]$ of degree at most $d = \mathsf{poly}(\lambda)$, instantiated using $q > p^{3\lceil \log_2(d+1)\rceil+1}$ and *GGen*, has witness extended emulation (Definition 2) if the Adaptive Root Assumption and the 2-Strong RSA Assumption hold for *GGen*.

Proof. The proof largely follows the same structure of the proof of Theorem 1.

We will prove security by showing that we can extract a dyadic rational polynomial $f(X) \in \mathbb{Z}[X]$ such that $f(X)$ has degree at most d and the coefficients of $f(X)$ are dyadic rationals such that the product of the numerator and denominator is bounded by $q/2$. This ensures that $f(q)$ is a unique encoding of $f(X)$. Additionally $\mathsf{g}^{f(q)} = \mathsf{C}$ and $f(z) \bmod p = y$. The proof will use the general forking lemma (Lemma 1) to show that the polynomial commitment scheme has witness-extended emulation.

We use the same extractor $\mathcal{X}$ as in Theorem 1 with one key distinction. For $d+1$ odd we invoke the extractor described by Lemma 9 instead of Lemma 8. This means that at every tree level either bounded dyadic rational witness polynomials are extracted or an element of known order or a non-power of 2 fractional root of g. By assumption the ladder two cases happen only with negligible probability.

We, therefore, now need to compute a bound on the size of the extracted polynomial. The check on f ensures that $f \in \mathbb{Z}$ and that $|f| \leq b = \frac{p-1}{2}(\frac{p+1}{2})^{\lceil \log_2(d+1)\rceil}$. We can write $f \in \mathbb{D}(\frac{p-1}{2}(\frac{p+1}{2})^{\lceil \log_2(d+1)\rceil}, 1)$. By 8 both the numerator and the denominator grow by at most a factor $p-1$ in every round. Given that the transcript tree has depth $\lceil \log_2(d+1)\rceil$ we get that the final extracted polynomial $f_0(X) \in \mathbb{D}(N, D)[X]$ has coefficients with numerators bounded by $N = \frac{p-1}{2}(\frac{p^2-1}{2})^{\lceil \log_2(d+1)\rceil}$ and denominators bounded by $D = (p-1)^{\lceil \log_2(d+1)\rceil}$

q needs to be large enough such that $f_0(X)$ is uniquely decodable, *i.e.*, $q > 2 \cdot N \cdot b = (p-1)^{\lceil \log_2(d+1)\rceil+1}(\frac{p^2-1}{2})^{\lceil \log_2(d+1)\rceil}$. In a simpler form $q > p^{3\lceil \log_2(d+1)\rceil+1}$ suffices.

This shows that we can successfully extract either a witness or a non-power of 2 fractional root or an element of known order from any tree of valid transcripts. Under the 2-Fractional Root Assumption and the Order Assumption, the probability that a polynomial time adversary along with a polynomial time extractor $\mathcal{X}$ can produce such a a non-power of 2 fractional root or an element of known order is negligible. Eval', therefore, has witness extended emulation and under the Adaptive Root Assumption by Lemma 7 so does Eval. Lemma 5 and Lemma 6 show that we can reduce the hardness assumptions to just the Adaptive Root Assumption and the 2-Strong RSA Assumption. □

A.5 Proof of Theorem 3

Theorem 3. The protocol $\mathsf{JoinedEval}$ is an interactive argument for the relation $\mathcal{R}_{\mathsf{JE}}$ and has perfect completeness and witness extended emulation if the Strong RSA and Order Assumption hold for *GGen* and if $q > (p-1)(\frac{p^2-1}{2})^{\lceil \log_2(d+1)\rceil+1}$ (e.g., $q > p^{2\log_2(d+1)+3}$).

Proof. Correctness is immediate and witness extended emulation requires a single application of Lemma 8 which leads to an updated bound on q. In general q needs to be $\frac{p^2-1}{2}$ larger for any random linear combination that is taken with $\alpha \in [-\frac{p-1}{2}, \frac{p-1}{2}]$. For class groups, Lemma 9 is used and the lower bound on q grows by a factor of $(p-1)^2\frac{p+1}{2} \leq p^3$. □

B Proof of Theorem 8 (Polynomial IOP Compilation)

Theorem 8. If the polynomial commitment scheme Γ has witness-extended emulation, and if the t-round Polynomial IOP for $\mathcal{R}$ has negligible knowledge error, then Π is a public-coin interactive argument for $\mathcal{R}$ that has witness-extended emulation. The compilation also preserves HVZK if Γ is hiding and Eval is HVZK.

The fact that the compilation preserves HVZK is straightforward. We prove this part first and then move on to proving witness-extended emulation.

HVZK

Proof. Let S_{Eval} denote the HVZK simulator for Eval and S_{IOP} denote the HVZK simulator for the original polynomial IOP. We construct an HVZK simulator S for the compiled interactive

argument as follows. S begins by running S_{IOP} on the input x, which produces a series of query/response pairs to arbitrarily labeled oracles that are "sent" from the IOP prover to the verifier. S simulates the view of the honest verifier in the compiled interactive proof by replacing each distinctly labeled oracle with a fresh Γ commitment to 0, *i.e.*, the zero polynomial over $\mathbb{F}_p$. By the hiding property of Γ this has negligible distance δ_0 from the commitment sent in the real protocol. (It places this commitment at the location in the transcript where the commitment to this oracle would be sent in the compiled protocol). For each query/response pair (z, y) to an oracle, S runs S_{Eval} to simulate the view of an honest-verifier in the Eval protocol opening a hiding polynomial commitment to the value y at the point z. Let P denote an upper bound on the total number of oracles sent and Q denote an upper bound on the total number of queries to IOP oracles. If the simulation of S_{IOP} has statistical distance δ_1 from the real IOP verifier's view, and each simulated Eval subprotocol has statistical distance δ_2 to the real Eval verifier's view, then the output of S has statistical distance at most $P\delta_0 + \delta_1 + Q\delta_2$ from $\mathsf{View}_{\langle P(x,w),V(x)\rangle}$. For $P, Q < \mathsf{poly}(\lambda)$ and $\delta_0, \delta_1, \delta_2 < \mathsf{negl}(\lambda)$ this statistical distance is negligible in λ. □

Witness-extended emulation (knowledge)

Proof. Without loss of generality, assume the original IOP makes at least one query to each oracle sent. An oracle which is never queried can be omitted from the IOP.

We denote by $\mathcal{V}$ the IP verifier for the compiled IP, and $\mathcal{V}_O$ the verifier for the original IOP. Given a record oracle $\mathsf{Record}(P^*, \mathsf{pp}, x, \mathsf{st})$ for an IP prover P^* that produces accepting transcripts with non-negligible probability, we build an emulator E for the compiled IP. E begins by constructing an IOP adversary P'_O, which succeeds also with non-negligible probability on input x. Every successful interaction of P'_O with $\mathcal{V}_O$ on input x corresponds to a successful transcript of P^* with V on x. In showing how E builds P'_O we also show how E can obtain this corresponding transcript. E will make use of the emulator E_{Eval} for the commitment scheme Γ.

Finally, E can use the IOP knowledge extractor $E_{\mathsf{IOP}}^{P'_O}(x)$ in order to output a witness for x along with the corresponding transcript.

Constructing P'_O (IOP adversary) P'_O runs as follows on initial state st_0 and input x. It internally simulates the interaction of P^* and V, using the record oracle $\mathsf{Record}(P^*, \mathsf{pp}, x, \mathsf{st})$. It begins by running this for the first round on state st_0. For every message that P^* sends in this first round, P'_O continues simulation until there is an Eval on this commitment. (There is guaranteed to be at least one Eval on each commitment, independent of the randomness). Therefore, denoting by E_{Eval} the extractor for the Eval subprotocol between P^* and $\mathcal{V}$ on a given commitment and evaluation point, the record oracle can be used to simulate E_{Eval}'s record oracle.

For each message m that P^* sends to V at the beginning of the first round, P'_O interprets m as a commitment, and attempts to extract from it a polynomial by running the PPT emulator E_{Eval}, simulating its record oracle as just described. **If it fails in any extraction attempt it aborts.**

If P'_O succeeds in all these extractions, then it uses these extracted polynomials as its first round proof oracles that it gives to $\mathcal{V}_O$. Upon receiving the first public-coin challenge from the IOP verifier, P' uses the query function to derive the corresponding queries to each of these proof oracles. Before answering, it rewinds P^* and $\mathcal{V}$ back to the point immediately after P sent its first messages, and now substitutes random challenge from $\mathcal{V}_O$ in order to simulate P^* and V on these same queries. It checks that P^*'s answers are consistent with the answers it can compute on its own from the extracted polynomials. **If any answers are inconsistent, P'_O aborts**. Otherwise, it sends the answers to $\mathcal{V}_O$.

At the end of this first round (assuming P' has not yet aborted), P'_O has stored an updated state st' for P^* based on this simulation. It proceeds to the next round and repeats the same process, using the record oracle $\mathsf{Record}(P^*, \mathsf{pp}, x, \mathsf{st}')$. Finally, if P' makes it through all rounds without aborting, then it has a final state st_V for $\mathcal{V}_O$ based on its internal simulation of P^* and V up through the end of the last round. Finally, $\mathcal{V}_O(\mathsf{st}_V)$ outputs Accept or Reject.

Analysis of P'_O success probability We claim that if $\mathsf{Record}(P^*, \mathsf{pp}, x, \mathsf{st}_0)$ outputs an accepting transcript tr with non-negligible probability, then P'_O succeeds with non-negligible probability.

Observe that for any accepting tr between P^* and V, if P'_O happens to follow the same exact sequence of query/responses without ever aborting then it succeeds because $\mathcal{V}_O$ and $\mathcal{V}$ run the same decision algorithm on the final state of query/response pairs. Thus, it remains only to take a closer look at what events cause P'_O to abort, and bound the fraction of accepting tr for which this occurs.

As indicated in bold above, there are two kinds of events that cause P'_O to abort:

- It fails to extract from a "commitment" message m sent by P^*
- After successfully extracting a polynomial f from a commitment, P^* answer queries to f in a way that is inconsistent with f.

The second type of event contradicts the evaluation binding property of Γ, therefore it occurs with negligible probability.

To analyze the first type of event, let us define "bad commitments" for a parameter D. We define this as a property of a message m (purportedly a commitment) sent in a transcript state st.

Bounding probability of commitment extraction failure The pair (m, st) is a "bad commitment" if there is less than a $1/D$ probability that extending the transcript between P^* and $\mathcal{V}$, starting from state st, will contain a successful execution of Eval on m. This probability is over the randomness of the public-coins of $\mathcal{V}$ in the extended transcript.

Let $A(\mathsf{tr})$ denote the event that a transcript tr sampled from $\mathsf{Record}(P^*, \mathsf{pp}, x, \mathsf{st}_0)$ is accepting. Let $B(\mathsf{tr})$ denote the event that tr contains a "bad commitment" (i.e. some message m sent in state st such that $\mathsf{Bad}(m, \mathsf{st}) = 1$). The conditional probability of event $A(\mathsf{tr})$ conditioned on event $B(\mathsf{tr})$ is less than $1/D$. To see this, fix (m, st) with $\mathsf{Bad}(m, \mathsf{st}) = 1$ and consider "sampling" a random tr that contains m at state st. This is done by first choosing randomly from all partial transcripts that result in (m, st) via brute force, and then running the transcript normally from state st on random public-coins. No matter how (m, st) is chosen, the probability that this process produces an accepting transcript is by definition less than $1/D$. (The second part of the transcript following (m, st) contains at least one execution of Eval on m by hypothesis, and by the definition of $B(m, \mathsf{st}) = 1$ this execution is accepting with probability less than $1/D$).

Assume that $P(A(\mathsf{tr})) \geq 1/\mathsf{poly}(\lambda)$. Applying Bayes' law,

$$P[B(\mathsf{tr})|A(\mathsf{tr})) \leq \frac{P[A(\mathsf{tr})|B(\mathsf{tr})]}{P(A(\mathsf{tr}))} \leq \mathsf{poly}(\lambda)/D \ .$$

In other words, at least a $1 - \mathsf{poly}(\lambda)/D$ fraction of accepting transcripts do not contain "bad commitments". Furthermore, so long as a commitment m is not "bad", we can invoke the witness-emulation property of Eval to say that the PPT E_Γ emulator extracts a witness polynomial from each m with overwhelming probability.

Setting $D = 2\mathsf{poly}(\lambda)$ we get that on at least a $1/2$ fraction of accepting transcripts, P'_Os simulation also succeeds (i.e. successfully extracts from each prover commitment message) with probability at least $1/2$. This means that P'_O has a non-negligible success probability conditioned on the event that tr is an accepting transcript.

In conclusion, if tr is accepting with non-negligible probability, then there is a non-negligible probability that P'_O succeeds. □