Transparent SNARKs from DARK Compilers

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. This submission is an extended abstract omitting preliminaries, optimizations and formal security proof. The full version of the paper is available online [Ano19].

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 statementsare 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, Set19, 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 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 variants of the Kate *et al.* commitment scheme for univariate polynomials [KZG10], which requires a trusted setup.

[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.

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 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.

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

[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 $O(n)$.

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] and PLONK [GWC19] 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.

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 $f(10) = 2341$ 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.

[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].

[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.*, $[-\frac{p-1}{2}; \frac{p-1}{2}]$.

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 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 Polynomial Commitments from Groups of Unknown Order

3.1 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 $[\![*]\!]$:

[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

- 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.

3.2 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 3.1. 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 $f(X) = f_L + Xf_R$, 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 $f_L + qf_R > 0$, 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 $f_L + qf_R = q^2/2 - 1$, 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 $g(X) = f^*_L + Xf^*_R$ 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 poylnomials 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. □

3.3 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. $[\![f(X)]\!]$, therefore, 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 $\mathsf{Setup}, \mathsf{Commit}$ 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 **??**)
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 $\mathsf{EvalBounded}(\mathsf{pp}, \mathsf{C}', z, y', d', bd; f'(X))$
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 **??** 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 $\mathsf{EvalBounded}(\mathsf{pp}, \mathsf{C}', z, y', d', b'; f'(X))$

3.4 Security Analysis

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 **??**) if the Adaptive Root Assumption and the Strong RSA Assumption hold for *GGen*.

The proof of this theorem is in the full version of the paper[Ano19]

3.5 Multivariate Polynomials, Zero-Knowledge and Optimizations

In the full version of the paper[Ano19] we show how to extend the scheme to multivariate polynomials. Additionally we give a hiding polynomial commitment with zero-knowledge evaluation proofs. We also give several optimizations, in particular showing how multiple commitments can be efficiently batch opened at multiple evaluation points.

3.6 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 **??** of the Appendix). 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.

3.7 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. $\mathbb{G}_U$ is a group of unknown order $\mathbb{G}_B$ is a group with a bilinear map (pairing), $\mathbb{G}_P$ is a prime order group with known order. EXP refers to exponentiation of a λ bit number in these groups. 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.	$\|\mathsf{pp}\|$	Prover	Verifier	$\|\pi\|$
DARK *(this work)*	yes	$O(1)$	$O(d^\mu\mu\log(d))$ EXP	$2\mu\log(d)$ 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 ($\mu=1$)	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.

4 Transparent SNARKs via Polynomial IOPs

In the full version of the paper we introduce the information theoretic concept of Polynomial IOPs. These are multi-round interactive proofs in which the prover sends oracles to polynomials in each round. The verifier can query these oracles in order to evaluate polynomials at query points. Polynomial IOPs additionally have a preprocessing phase where a verifier can preprocess oracles to polynomials that depend only on the language (e.g. the circuit) but not on the specific instance or witness. The main theorem is that we can compile an efficient Polynomial IOP to a SNARK using efficient polynomial commitments and the Fiat-Shamir transform.

5 Evaluation

We now evaluate Supersonic, the trustless-setup SNARK built on the Polynomial IOPs underlying Sonic [MBKM19] and PLONK [GWC19] and compiled using our DARK polynomial commitment scheme. As explained in Section **??**, 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 **??**). 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]. 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[7], this gives us for $\lambda = 120$ and $n = 2^{20}$ a verification time of around 72ms.

Scheme	Transp.	|pp|	Prover	Verifier	$\lvert\pi\rvert$	$n = 2^{20}$
Supersonic	yes	$O(1)$	$O(n \log(n))$ EXP	$O(\log(n))$ EXP	$O(\log(n)\ \mathbb{G}_U)$	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

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. $\mathbb{G}_U$ is a group of unknown order $\mathbb{G}_B$ is a group with a bilinear map (pairing), $\mathbb{G}_P$ is a prime order group with known order. EXP refers to exponentiation of λ-bit numbers in these groups. 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]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`

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.

[Ano19] Anonymized. Transparent snarks from dark compilers. Cryptology ePrint Archive, Report 2019/1229, 2019. `https://eprint.iacr.org/2019/1229`.

[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.

[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.

[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.

[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.

[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.

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

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

[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.

[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.

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

[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.

[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.

[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/.

[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.

[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.

[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.

[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`.

[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`.