Shorter and Faster Post-Quantum Designated-Verifier zkSNARKs from Lattices[*]

Yuval Ishai
Technion
yuvali@cs.technion.ac.il

Hang Su
University of Virginia
hs2nu@virginia.edu

David J. Wu
UT Austin
dwu4@cs.utexas.edu

Abstract

Zero-knowledge succinct arguments of knowledge (zkSNARKs) enable efficient privacy-preserving proofs of membership for general NP languages. Our focus in this work is on *post-quantum* zkSNARKs, with a focus on minimizing proof size. Currently, there is a 1000× gap in the proof size between the best pre-quantum constructions and the best post-quantum ones. Here, we develop and implement new *lattice-based* zkSNARKs in the designated-verifier preprocessing model. With our construction, after an initial preprocessing step, a proof for an NP relation of size 2^{20} is just over 16 KB. Our proofs are 10.3× shorter than previous post-quantum candidates. Compared to previous lattice-based zkSNARKs (also in the designated-verifier preprocessing model), we obtain a 42× reduction in proof size and a 60× reduction in the prover's running time, all while achieving a much higher level of soundness. Finally, compared to the shortest pre-quantum zkSNARKs by Groth (Eurocrypt 2016), the proof size in our lattice-based construction is 131× longer, but the prover's running time is 1.2× *faster*.

Our construction follows the general blueprint of Bitansky et al. (TCC 2013) and Boneh et al. (Eurocrypt 2017) of combining a linear probabilistically checkable proof (linear PCP) together with a linear-only vector encryption scheme. We develop a concretely-efficient lattice-based instantiation of this compiler by considering quadratic extension fields of moderate characteristic and using linear-only vector encryption over rank-2 module lattices.

1 Introduction

A zero-knowledge proof of knowledge [GMR85] for an NP relation $\mathcal{R}$ enables a prover to convince a verifier that a statement is true without revealing anything more about the statement. In a zero-knowledge succinct argument of knowledge (zkSNARK) [Kil92, Mic00, GW11], we additionally require that the proof consist of a single message π from the prover to the verifier, and moreover, that the length of the proof π and the verification complexity be sublinear (ideally, polylogarithmic) in the size of the circuit computing $\mathcal{R}$. Zero-knowledge SNARKs have applications to delegating and verifying computations [WB15] and for constructing privacy-preserving cryptocurrencies [BCG$^+$14]. In the last few years, there have been numerous works studying constructions from different assumptions and on optimizing the asymptotic and concrete efficiency of zkSNARKs (e.g., [PHGR13, BCI$^+$13, BCC$^+$16, Gro16, ZGK$^+$17, AHIV17, BBHR18b, WTS$^+$18, GMNO18, BBB$^+$18, BCR$^+$19, CHM$^+$20, BFS20, SL20, COS20, LSTW21, CY21, GNS21]).

Post-quantum zkSNARKs. Many existing constructions of practical zkSNARKs for NP rely on group-based and pairing-based assumptions [Gro10, PHGR13, GGPR13, BCI$^+$13, Gro16, BCC$^+$16, BBB$^+$18, MBKM19, CHM$^+$20, Set20, SL20] and are insecure against quantum adversaries. Several recent works have introduced new concretely-efficient post-quantum zkSNARKs based on cryptographic hash functions [AHIV17, BBHR18b, BCR$^+$19, COS20, BFH$^+$20] or lattice-based assumptions [GMNO18]. However, compared to their

[*] This is the extended version of a paper by the same title that appeared at ACM CCS 2021 [ISW21].

pre-quantum analogs, current post-quantum constructions have substantially longer proofs. As a point of comparison, proofs in the most succinct pre-quantum construction by Groth [Gro16] are just 128 bytes while those in the most succinct post-quantum constructions [BCR$^+$19, COS20, BFH$^+$20] are over 1000× longer (see Table 1). The increase in parameter sizes is not entirely surprising since a similar, although smaller, gap exists between the sizes of group-based pre-quantum signatures [Sch80, BLS01] and hash-based [BHH$^+$15, CDG$^+$17] or lattice-based post-quantum signatures [DKL$^+$18, FHK$^+$20].

This work: lattice-based designated-verifier zkSNARKs. Our focus in this work is on new approaches for constructing shorter (and faster) post-quantum zkSNARKs from *lattice-based assumptions*. Like recent works [GGPR13, SBV$^+$13, BCR$^+$19, Set20, COS20, SL20], we focus on the NP-complete language of rank-1 constraint satisfiability (R1CS), which generalizes Boolean and arithmetic circuit satisfiability and enjoys efficient compiler support from other program representations [SVP$^+$12, BCGT13, BCG$^+$13, PHGR13, BCTV14b, BFR$^+$13]. While recent works have introduced post-quantum zkSNARKs from lattice-based assumptions [BISW17, BISW18, GMNO18, Nit19], to our knowledge, only the construction of Gennaro et al. [GMNO18] has been implemented. Its proof sizes are significantly worse compared to alternative post-quantum constructions based on interactive oracle proofs (IOPs) and the Fiat-Shamir heuristic (e.g., 640 KB for the lattice-based approach [GMNO18] vs. 169 KB for the IOP-based approach [BCR$^+$19]). The prover time for current lattice-based instantiations is also over 10× worse than the alternative constructions.

Similar to the previous lattice-based constructions, we design our zkSNARKs in the designated-verifier preprocessing model where there is an (expensive but practically feasible) setup algorithm that samples public parameters and a *secret* verification key (needed to verify proofs). While the designated-verifier model is a relaxation of the conventional setting of zkSNARKs, it nonetheless suffices for applications to verifiable computation and other privacy-preserving protocols.

Our results. Our main result is a new designated-verifier zkSNARK from lattice-based assumptions where the proof size (for verifying an R1CS instance of size 2^{20}) is just over 16 KB. This is a 10.3× reduction in proof size compared to Aurora [BCR$^+$19], a post-quantum IOP-based SNARK with short proofs. If we restrict our attention to post-quantum zkSNARKs with *sublinear* verification, our construction is 13.1× shorter than Fractal [COS20]. Compared to the lattice-based construction of Gennaro et al. [GMNO18], our zkSNARKs are 42.1× shorter. However, there remains a large gap (131×) compared to the shortest *pre-quantum* zkSNARK by Groth [Gro16]. We refer to Table 1 for the full comparison and describe our experimental setup in detail in Section 4.3.

The prover and verifier complexities of our new zkSNARK compare favorably with other post-quantum schemes. Our construction is over 4.5× faster for the prover compared to Aurora and Fractal on R1CS instances of similar size. Compared to the Gennaro et al. lattice-based candidate [GMNO18], our construction is 60× faster for the prover and 5.1× faster for the verifier. Compared to the pre-quantum pairing-based construction of Groth [Gro16], our construction is 1.2× faster for the prover and 2.8× faster for the verifier. Using an alternative instantiation of our construction with *longer* proofs (20.8 KB vs. 16.4 KB), our construction is 1.4× faster than the pairing-based construction.

Further improvements to the proof size and prover complexity are possible if we relax zero knowledge. For instance, a variant of our construction that is sound but *not* provably zero knowledge is over 2.3× faster for the prover than the pairing-based construction of Groth and has a proof size of 11.1 KB (for verifying an R1CS instance with 2^{20} constraints). This construction is suitable for applications that do not require zero knowledge, or alternatively, can tolerate a small amount of leakage. Note that while we do not *prove* zero knowledge of this variant, the construction can still provide full zero knowledge assuming that the underlying information-theoretic building block we use (linear PCPs) remains zero knowledge in the presence of leakage. We provide more details in Remark 3.24 and Lemma 3.26. We leave the question of determining whether the linear PCPs we use (see Appendix A) or variants thereof satisfy the stronger notion of zero knowledge with leakage to future work. In Section 4.3, we show trade-offs between the number of bits of *provable* zero knowledge provided by our construction and its concrete efficiency.

Scheme	Structure	PQ	TP	PV	R1CS	Size		Time		
					Size	CRS	Proof	Setup	Prover	Verifier
[Gro16]	Pairings	○	○	●	2^{16} 2^{20}	12.4 MB 199 MB	128 B 128 B	5.6 s 72 s	5.5 s 79 s	3.3 ms 3.4 ms
[GMNO18]*	Lattices	●	○	○	2^{16}	17.3 MB†	640 KB	167 s	235 s	3.5 ms
Ligero [AHIV17]	Random Oracle	●	●	●	2^{16} 2^{20}	— —	4.3 MB 14 MB	— —	2.5 s 38 s	1.3 s 22 s
Aurora [BCR+19]	Random Oracle	●	●	●	2^{16} 2^{20}	— —	121 KB 169 KB	— —	18 s 304 s	380 ms 6.3 s
Fractal [COS20]‡	Random Oracle	●	●	●	2^{16} 2^{19}	1.4 GB 11 GB	178 KB 215 KB	12 s 116 s	21 s 184 s	8.3 ms 9.5 ms
This work (Shorter Proofs)	Lattices	●	○	○	2^{16} 2^{20}	191 MB 5.3 GB	15.2 KB 16.4 KB	88 s 2240 s	3.9 s 68 s	0.69 ms 1.2 ms
This work (Shorter CRS)	Lattices	●	○	○	2^{16} 2^{20}	104 MB 1.9 GB	19.9 KB 20.8 KB	53 s 877 s	3.4 s 56 s	0.37 ms 0.43 ms

*As we discuss in Appendix D (Remark D.4), the parameter instantiation proposed in Gennaro et al. [GMNO18] only provides 15 bits of provable soundness. If we use parallel repetition to amplify to 128-bits of soundness, then all of the parameters should be scaled by a factor of 8.5×. In the table, we report the numbers as they were presented in the original paper. Their work also does not provide measurements for instances with more than 2^{16} gates.

†Gennaro et al. [GMNO18] do not report the CRS size for an instance of size 2^{16}. We estimate the size by doubling the size of the CRS for an instance of size 2^{15}.

‡The "Setup" time and "CRS" size for Fractal refers to the running time of the indexer and the size of the resulting proving state. Our system ran out of memory when running Fractal on an R1CS instance of size 2^{20}. Thus, we report the results for an instance of size 2^{19} instead.

Table 1: Concrete performance comparison of our zkSNARK to the pairing-based construction of Groth [Gro16] and several recent post-quantum zkSNARKs with polylogarithmic-size proofs. For each scheme, we report the running time and parameter sizes for an R1CS instance with 2^{16} and 2^{20} constraints. We measure the running times for an R1CS instance over each scheme's preferred field. With the exception of the Gennaro et al. [GMNO18] construction, all measurements are taken on the *same* system (see Section 4.3 for details of our setup). For our scheme, we consider two different parameter settings. The "Shorter Proofs" instantiation works over the field $\mathbb{F}_{p^2}$ where $p = 2^{13} - 1$ and the "Shorter CRS" instantiation works over the field $\mathbb{F}_{p^2}$ where $p = 2^{19} - 1$ (see Table 2 for the lattice parameters in these instantiations). The "PQ" column specifies whether the construction is post-quantum (●) or pre-quantum (○), the "TP" column specifies whether the construction has a transparent setup (●) or relies on a trusted setup (○), and the "PV" column specifies whether the scheme is publicly-verifiable (●) or designated-verifier (○).

Compared to other post-quantum constructions based on "MPC-in-the-head" [IKOS07, AHIV17, BFH+20] or the "GKR" approach [GKR08, ZXZS20, ZWZZ20], our prover times are generally higher. For instance, compared to Ligero [AHIV17], our prover is about 1.8× more expensive, but our proofs and verification times are over 874× better (see Table 1). In general, these alternative approaches typically enjoy smaller concrete prover costs, but either have longer proofs or higher verification costs for general, *unstructured* computations. We provide more details and comparisons with other zkSNARKs in Section 5.

The IOP-based constructions have the advantage of being *publicly-verifiable* and *transparent*. Our scheme is designated-verifier and requires an expensive trusted setup. For verifying R1CS systems with 2^{20} constraints, we need to sample a CRS of size 5.3 GB which takes 37 minutes. An alternative instantiation of our construction over a larger finite field reduces the CRS size to 1.9 GB and the setup time to 15 minutes. This leads to a modest increase in proof size from 16.4 KB to 20.8 KB (see Table 1).

Limitations of our construction. While our lattice-based zkSNARK achieve better succinctness compared to other post-quantum zkSNARK candidates, they have several limitations that give rise to natural directions for improvement. We highlight some of these here:

- **Reusable soundness and public verification.** As noted above, our lattice-based zkSNARK is in the *designated-verifier* model where a *secret* verification key is needed to verify proofs. Moreover, like existing lattice-based designated-verifier zkSNARKs [GMNO18, BISW18], our construction does not provide *reusable soundness* where soundness holds even against a malicious prover who can make an arbitrary polynomial number of queries to the verification oracle. Constructing a lattice-based zkSNARK with comparable concrete efficiency and reusable soundness is an interesting direction. As we discuss in greater detail in Remark 3.22, even without a provable notion of reusable soundness, our construction still suffices for some applications to verifiable computation. In particular, breaking soundness requires the prover to submit a *super-constant* number of "bad" proofs to the verifier. This means that the verifier is able to detect a malicious prover trying to attack the scheme (this is reminiscent of the notion of *covert security* from [AL07]). Alternatively, these "selective failure" attacks can be avoided altogether if the verifier does not reveal whether a proof is valid or not to the prover.

  More generally, it is a fascinating question to construct *publicly-verifiable* lattice-based zkSNARKs with comparable concrete efficiency. Existing lattice-based constructions [BBC+18, BLNS20] that are publicly verifiable have an expensive verifier (i.e., the verifier runs in time linear in the size of the underlying NP relation). We refer to Section 5 for further comparison with related work.

- **Field characteristic.** In this work, we consider lattice-based zkSNARKs for R1CS systems over finite fields of moderate characteristic (i.e., between 12 and 20 bits). Specifically, we consider quadratic extension fields, which enable a number of concrete optimizations (see Sections 1.2 and 4.3).

  For some applications, it may be helpful to consider R1CS systems over fields of higher characteristic. For example, validity of a 32-bit addition gate (on 32-bit inputs) can be expressed as a single R1CS constraint over any finite field with characteristic $p > 2^{32}$. Thus, when verifying computations that make extensive use of 32-bit or 64-bit integer arithmetic, it can be advantageous to encode the computation in an R1CS system over a higher characteristic field. Both the pairing-based construction [Gro16] as well as the hash-based constructions [Gro16, AHIV17, BCR+19, COS20] operate over base fields of high characteristic (i.e., at least 128 bits). The hash-based constructions [AHIV17, BCR+19, COS20] also efficiently support R1CS systems on high-degree extensions of the binary field, or more generally, any field that supports efficient fast Fourier transforms.

  While our lattice-based instantiation can in principle support fields of higher characteristic, doing so will require using larger lattice parameters, which in turn, increases the proof size. Moreover, some of our concrete optimizations (e.g., implementing all arithmetic operations using 128-bit integer arithmetic) can no longer be applied when the field characteristic increases. We refer to Section 4.3 and Fig. 2 for more discussion on how the field characteristic affects the concrete efficiency of our scheme.

1.1 Background

The basis of our work is the compiler of Bitansky et al. [BCI+13] (also implicit in the work of Gennaro et al. [GGPR13]), and more specifically, the vector generalization by Boneh et al. [BISW17]. These works provided a general template for constructing SNARKs in the preprocessing model by combining a "linear PCP" with a "linear-only" encryption scheme. A linear PCP [IKO07] for an NP language $\mathcal{L}$ is defined by a linear oracle $\boldsymbol{\pi} \colon \mathbb{F}^\ell \to \mathbb{F}$ over a finite field $\mathbb{F}$. On input a statement x, a verifier can submit a query matrix $\mathbf{Q} \in \mathbb{F}^{\ell \times k}$ to the oracle and obtain the responses $\mathbf{a} \leftarrow \mathbf{Q}^\mathsf{T} \boldsymbol{\pi} \in \mathbb{F}^k$. Based on the responses, the verifier decides whether to accept or reject. We refer to k as the number of queries and ℓ as the query length of the linear PCP. The linear PCP is sound if for a false statement $x \notin \mathcal{L}$ and any proof vector $\boldsymbol{\pi} \in \mathbb{F}^\ell$, the probability that the verifier accepts is negligible (where the probability is taken over the choice of $\mathbf{Q}$). Concretely-efficient 4-query linear PCPs for R1CS can be constructed using the quadratic arithmetic programs (QAPs) introduced by Gennaro et al. [GGPR13]. QAPs are the basis for the most succinct pairing-based preprocessing zkSNARKs [PHGR13, GGPR13, BCI+13, BCG+13, Gro16].

To obtain a preprocessing zkSNARK for $\mathcal{L}$ from a linear PCP for $\mathcal{L}$, the Bitansky et al. compiler encrypts the linear PCP queries (i.e., the entries of $\mathbf{Q}$) using a "linear-only" encryption scheme and publishes the resulting ciphertexts as part of the common reference string (CRS). As the name suggests, a linear-only encryption scheme is an encryption scheme that only supports linear homomorphism (i.e., it is possible to add ciphertexts, but no other homomorphic operation on ciphertexts is supported). Given the encrypted queries, the prover can homomorphically compute the encrypted responses $\mathbf{a} = \mathbf{Q}^{\mathsf{T}} \boldsymbol{\pi}$. Here, the linear-only property restricts the prover to linear strategies and by semantic security, the prover's choice of linear combination is essentially independent of the linear PCP queries. This binds the prover to respect the constraints of the linear PCP model. To verify the proof, the verifier decrypts the encrypted responses and evaluates the linear PCP verification procedure. This yields a designated-verifier preprocessing SNARK. For zero knowledge, it suffices that the linear PCP be *honest-verifier* zero knowledge and the linear-only encryption scheme be "re-randomizable" (i.e., ciphertexts output by the homomorphic evaluation are computationally indistinguishable from fresh ciphertexts).

Lattice-based instantiations of Bitansky et al. Gennaro et al. [GMNO18], following Boneh et al. [BISW17, BISW18], introduced candidate linear-only encryption schemes based on lattices. In these works, the underlying linear-only encryption scheme is adapted from basic Regev encryption [Reg05]. For our purposes, a Regev-based encryption of a value $x \in \mathbb{Z}_p$ is a pair $(\mathbf{a}, \mathbf{c})$ where $\mathbf{a} \in \mathbb{Z}_q^n$ and $\mathbf{c} = \mathbf{s}^{\mathsf{T}} \mathbf{a} + pe + x \in \mathbb{Z}_q$, where $\mathbf{s} \in \mathbb{Z}_q^n$ is the secret key, $e \in \mathbb{Z}_q$ is an error term, and n, q are lattice parameters. Observe that this scheme is linearly homomorphic: if $(\mathbf{a}_1, \mathbf{c}_1)$ and $(\mathbf{a}_2, \mathbf{c}_2)$ encrypt values x_1, x_2, respectively, then $(\mathbf{a}_1 + \mathbf{a}_2, \mathbf{c}_1 + \mathbf{c}_2)$ encrypts the value $x_1 + x_2 \bmod p$, albeit with slightly larger error. As long as the error magnitude in the final ciphertext is less than $q/(2p)$, decryption succeeds.

Gennaro et al. [GMNO18] provided the first lattice-based implementation of the Bitansky et al. compiler using Regev encryption.[1] Compared to the best pairing-based constructions that followed a similar methodology [Gro16], the lattice-based implementation is significantly less efficient. For an R1CS instance of size 2^{16}, the proof size is 640 KB, over 5000× larger than the pairing-based construction of Groth [Gro16]; similarly, the prover time for a similar-sized instance is roughly 40× slower than the pairing-based analog. In fact, as we discuss in Appendix D, because Gennaro et al. consider linear PCPs over a *small* field $\mathbb{F}$ ($\log |\mathbb{F}| = 32$), the specific parameter instantiation they consider provides at most 15 bits of provable soundness. Working over a larger field or using parallel repetition for soundness amplification would incur even more overhead.

Lattice parameter sizes. The main obstacle to the concrete efficiency of lattice-based zkSNARKs following the Bitansky et al. compiler [BCI+13] is the size of the lattice parameters. The length of a QAP for an R1CS instance with N constraints over a finite field $\mathbb{F}$ is $O(N)$, and the soundness error is $O(N/|\mathbb{F}|)$. This means we need to work over a field $\mathbb{F}$ where $|\mathbb{F}| > N$, and we need a linear-only encryption scheme over $\mathbb{F}$ that supports $O(N)$ homomorphic operations. In the Gennaro et al. construction [GMNO18], they consider a prime field $\mathbb{F}_p$ where $p > N$. For correctness then, the modulus q for a Regev-based encoding must satisfy $q > 2p^2 N$. Zero knowledge adds a further multiplicative factor of 2^κ where κ is a statistical security parameter.

To achieve 128 bits of soundness, one approach is to set $p > 2^{128} N$. If we take $q \approx 2^{300}$ (and a typical error distribution), then the lattice dimension n needs to be at least 10^4 at the 128-bit security level (based on [APS15]). A *single* ciphertext is over 350 KB in this setting. This is a lower bound on the proof size since in the basic instantiation, the proof contains at least one ciphertext for each linear PCP response.

Alternatively, instead of working over a large field, we can work over a small field $\mathbb{F}_p$ where $p \approx N$ and amplify soundness through parallel repetition. For instance, if we take $p \approx 2^{20}$ and $q \approx 2^{100}$, then a single Regev ciphertext is roughly 45 KB. However, soundness amplification increases the proof size (and all other metrics), again leading to parameter sizes that are significantly worse than non-lattice-based zkSNARKs. The scheme of Gennaro et al. [GMNO18] considers a finite field of size 2^{32} *without* soundness amplification, and so their concrete instantiation provides very few bits of provable soundness (see Appendix D and Remark D.4). But even with this choice of parameters, the proof size in their construction is already 640 KB.

[1] Gennaro et al. used square span programs [DFGK14] instead of QAPs as the underlying linear PCP, but this distinction is not important for the main discussion here.

1.2 Technical Overview

The primary enablers of our concretely-efficient lattice-based zkSNARK are (1) using *vector encryption* [PVW08] instead of vanilla Regev encryption as our linear-only encryption scheme; and (2) working over *extension fields* of *moderate characteristic*. We provide an overview of our techniques and construction here.

Vector encryption. Our starting point in this work is the adaptation of the Bitansky et al. compiler using linear-only *vector encryption* introduced by Boneh et al. [BISW17]. As the name suggests, a vector encryption scheme (over a field $\mathbb{F}$) supports encrypting a vector of field elements. Instead of encrypting each entry in the linear PCP query matrix $\mathbf{Q} \in \mathbb{F}^{\ell \times k}$ separately, the Boneh et al. compiler encrypt rows of $\mathbf{Q}$. The proof then consists of a single ciphertext encrypting the vector of linear PCP responses. The advantage of this approach is that we can take advantage of amortization to reduce the ciphertext expansion for lattice-based vector encryption. In more detail, with vanilla Regev encryption, the overhead of encrypting a *single* $\mathbb{Z}_p$ value is $O(n)$, where n is the lattice dimension. Using the extension by Peikert et al. [PVW08], we can encrypt a vector of ℓ $\mathbb{Z}_p$-values with a ciphertext containing $(n + \ell)$ $\mathbb{Z}_q$-elements. This approach confers several improvements for concrete efficiency:

- **Soundness amplification:** We can now amplify soundness of the linear PCP using parallel repetition (i.e., using multiple independent sets of linear PCP queries). This increases the dimensions of the vectors we encrypt, but using the Peikert et al. vector encryption scheme, the overhead is *additive* in the dimension rather than multiplicative (as with vanilla Regev encryption).

- **Number of lattice ciphertexts:** For an encryption scheme to plausibly satisfy the "linear-only" property, the ciphertext space must be sparse, and in particular, the adversary should not be able to obliviously sample a valid ciphertext *without* knowledge of the corresponding plaintext value. The heuristic from earlier works [GGPR13, BCI$^+$13, BISW17] is to use "double encryption" where a valid ciphertext encrypting a message x consists of a pair of independent ciphertexts encrypting x (Gennaro et al. [GMNO18] also use a variant of this encoding method). In Section 3.3, we describe an alternative approach to sparsify the ciphertext space by first embedding the plaintext vector $\mathbf{v} \in \mathbb{Z}_p^\ell$ within a (secret) subspace $T \subseteq \mathbb{Z}_p^{\ell+\tau}$ and then encrypting the embedded vector $\mathbf{v}' \in \mathbb{Z}_p^{\ell+\tau}$. Here, τ is a "sparsification" parameter. A ciphertext is valid only if it encrypts an element of the subspace T. When T has negligible density in $\mathbb{Z}_p^{\ell+\tau}$, we conjecture that an adversary (that does not know T) cannot obliviously sample a valid ciphertext without knowledge of the corresponding plaintext vector. The advantage of this approach is that it incurs a small *additive* overhead on ciphertext/proof size rather than a multiplicative one. Thus, using vector encryption, we can encrypt a vector of plaintext values using a *single* lattice ciphertext (and still plausibly prevent oblivious sampling of ciphertexts).

Reducing ciphertext size with modulus switching. Homomorphic operations on lattice ciphertexts increase the noise in the ciphertexts. To ensure decryption correctness, the ciphertext modulus q must be large enough to accommodate the accumulated noise from the homomorphic operations. The modulus switching technique developed in the context of fully homomorphic encryption [BV11, BGV12, CNT12, AP14, DM15] provides a way to reduce the size of the ciphertexts *after* performing homomorphic operations. Specifically, modulus switching takes a ciphertext with respect to a modulus q and scales it down to a new ciphertext with respect to a modulus $q' < q$ (while preserving decryption correctness). This technique applies to most Regev-based encryption schemes, including the vector encryption scheme we use. In our specific setting, after the prover homomorphically computes the encrypted vector of linear PCP responses, the prover applies modulus switching to the resulting ciphertext. For our parameter settings, this yields a 2× to 3× reduction in ciphertext size (and correspondingly, in proof size).

Instantiating our vector encryption scheme over $\mathbb{F}_p$ using a 23-bit characteristic p yields a zkSNARK where the proof size is 27 KB and the CRS size is 9.6 GB (for verifying R1CS instances with 2^{20} constraints). Using a larger 28-bit characteristic, the proof size increases to 29 KB and the CRS size decreases to 2.7 GB for the same setting. Without modulus switching, the proof sizes for these two settings are 66 KB and 72 KB, respectively. While these basic instantiations already improve on previous post-quantum zkSNARKs in terms of proof size, the improvements come at the expense of needing a very large CRS. Below, we show how to use extension fields to obtain instantiations with a shorter CRS and a shorter proof.

Extension fields of moderate characteristic. The second ingredient in our construction is a way to reduce the lattice parameters themselves by considering linear PCPs over extension fields of moderate characteristic. The key observation we make is that the size of the modulus q (and other lattice parameters) scale with the plaintext modulus (i.e., the field *characteristic*) but *not* necessarily the *size* of the field. To take advantage of this, we first note that linear PCPs based on QAPs are agnostic to the choice of the field, and work equally well over extension fields $\mathbb{F}_{p^k}$. We develop two instantiations of this approach:

- **Compile linear PCPs over $\mathbb{F}_{p^k}$ to $\mathbb{F}_p$:** Our first instantiation shows how to compile a linear PCP over $\mathbb{F}_{p^k}$ to a zkSNARK using linear-only vector encryption over the base field $\mathbb{F}_p$ (i.e., the same encryption scheme from above). To do so, we first show how to transform a linear PCP over $\mathbb{F}_{p^k}$ to a linear PCP over $\mathbb{F}_p$. The transformation increases the query length and the number of queries by a factor of k, and relies on the fact that $\mathbb{F}_{p^k}$-operations correspond to linear transformations over the vector space $\mathbb{F}_p^k$. We describe our construction in Section 3.1. For concrete efficiency reasons, we focus exclusively on quadratic extensions (see Remark 3.17). Using one instantiation of this approach, we obtain a construction with shorter proofs (21 KB) *and* a shorter CRS (3.8 GB) compared to working over the prime field.[2] With a longer CRS (10.5 GB), we can bring the proof size down to just 16 KB.

- **Vector encryption over extension fields.** We next consider a *direct* compilation from linear PCPs over the extension field to a zkSNARK using a linear-only vector encryption scheme whose plaintext space coincides with the extension field. To do so, we generalize our variant of the Peikert et al. [PVW08] encryption scheme to operate over the cyclotomic ring $R = \mathbb{Z}[x]/(x^2 + 1)$. In this case, the plaintext space is $R_p = R/pR$. When $p = 3 \bmod 4$, $R_p \cong \mathbb{F}_{p^2}$. Under the conjecture that the vector encryption scheme is linear-only over R_p, this gives a direct compilation from a linear PCP over a quadratic extension $\mathbb{F}_{p^2}$ to a zkSNARK over $\mathbb{F}_{p^2}$. By relying on linear PCPs and linear-only vector encryption over the quadratic extension, we obtain a zkSNARK with similar proof size as the above construction, but with a 2× reduction in the CRS size (previously incurred by transforming the linear PCP from $\mathbb{F}_{p^2}$ to $\mathbb{F}_p$). We show the concrete performance in Table 1 and in Section 4.3. Leveraging encryption schemes over extension fields and higher-rank modules has also been useful for improving the asymptotic and concrete efficiency of other lattice-based constructions [Gen09, GHS12a, GHS12b].

Parameter selection. In this work, we consider quadratic extension fields with two different characteristics: (1) $p = 2^{13} - 1$ which yields a construction with shorter proofs but a longer CRS; and (2) $p = 2^{19} - 1$ which yields a construction with a shorter CRS and slightly longer proofs. We choose p of the form $2^t - 1$ so that $\mathbb{F}_{p^2}^*$ has a multiplicative subgroup of order 2^{t+1} (i.e., the subgroup of 2^{t+1}-th roots of unity). This enables us to take advantage of fast Fourier transforms (FFT) to implement the linear PCP prover [BCG⁺13]. Note that when p is sufficiently small (e.g., $p = 2^{13} - 1$), the extension field does not contain a sufficiently-large subgroup of roots of unity to directly leverage power-of-two FFTs for the linear PCP. In Section 4.1, we describe a simple approach using multiple small power-of-two FFTs on different cosets of the roots of unity that still enables an efficient implementation of the linear PCP prover.

Working over extension fields also allows us to use a smaller ciphertext modulus q in the lattice-based encryption scheme. When $q < 2^{128}$, we can use compiler intrinsic types for 128-bit integer arithmetic for our computations. This is significantly faster than using multi-precision arithmetic or even fixed-precision arithmetic over slightly larger integers. We provide more discussion and microbenchmarks in Section 4.2 and Table 3.

[2] Even though the linear PCP transformation doubles the query length of the linear PCP, working over the extension field allows us to achieve the same level of soundness with fewer parallel repetitions, and *reduces* the overall size of the CRS.

Zero knowledge and circuit privacy. As noted above, the Bitansky et al. compiler yields a zero-knowledge SNARK if the underlying linear PCP is honest-verifier zero knowledge and the linear-only encryption scheme is re-randomizable. However, the lattice-based schemes are not directly re-randomizable (due to the accumulation of noise through homomorphic operations). In this work, we show that a weaker notion of circuit privacy [Gen09] suffices to argue zero knowledge for the SNARK (i.e., the ciphertext obtained from taking a linear combination of ciphertexts hide the coefficients of the linear combination). Using noise smudging [Gen09, AJLA$^+$12, MW16] and the module learning with errors assumption (MLWE) [BGV12, LS15], it is straightforward to augment our linear-only vector encryption scheme to provide circuit privacy. We give the details in Section 3.3. We additionally note in Remark 3.24 that even without circuit privacy, a direct compilation from a linear PCP satisfying honest-verifier zero knowledge to a zkSNARK can still provide full zero knowledge if the underlying linear PCP remains zero knowledge given some additional information on the linear PCP coefficients. This variant without provable zero knowledge enables a further 30-40% reduction in prover time and a 45-50% reduction in proof size.

Implementation and evaluation. In Section 4, we describe our implementation of our lattice-based zkSNARK. We provide a comprehensive evaluation of the different trade-offs in parameter sizes and computational costs for the different settings described here. We also give fine-grained microbenchmarks of the different components of our system in Section 4.3. Finally, we conclude with additional comparisons against other zkSNARK candidates in Section 5.

2 Preliminaries

Throughout this work, we write λ (oftentimes implicitly) to denote the security parameter. For a positive integer $n \in \mathbb{N}$, we write $[n]$ to denote the set $\{1, \dots, n\}$. We write $\{x_i\}_{i \in [n]}$ to denote the ordered multi-set of values $x_1, \dots, x_n$. We will typically use bold lowercase letters (e.g., $\mathbf{v}, \mathbf{w}$) to denote vectors and bold uppercase letters (e.g., $\mathbf{A}, \mathbf{B}$) to denote matrices. For a vector $\mathbf{v} \in \mathbb{Z}_p^n$, we will use non-boldface letters to refer to its components; namely, we write $\mathbf{v} = (v_1, \dots, v_n)$. For a vector $\mathbf{v} \in \mathbb{R}^n$, we write $\|\mathbf{v}\|_\infty$ to denote the ℓ_∞ norm of $\mathbf{v}$. For a finite set S, we write $x \xleftarrow{\text{R}} S$ to denote that x is sampled uniformly from S. For a distribution $\mathcal{D}$, we write $x \leftarrow \mathcal{D}$ to denote that x is sampled from $\mathcal{D}$.

We say that a function f is negligible in λ if $f(\lambda) = o(1/\lambda^c)$ for all $c \in \mathbb{N}$; we denote this $f(\lambda) = \mathsf{negl}(\lambda)$. We write $\mathsf{poly}(\lambda)$ to denote a function bounded by a fixed polynomial in λ. We say an event happens with negligible probability if the probability that the event occurs is negligible, and that it happens with overwhelming probability if its complement occurs with negligible probability. We say an algorithm $\mathcal{A}$ is efficient if it runs in probabilistic polynomial time in the length of its input. We say that two families of distributions $\mathcal{D}_1 = \{\mathcal{D}_{1,\lambda}\}_{\lambda \in \mathbb{N}}$ and $\mathcal{D}_2 = \{\mathcal{D}_{2,\lambda}\}_{\lambda \in \mathbb{N}}$ are computationally indistinguishable if no efficient adversary can distinguish samples from $\mathcal{D}_1$ and $\mathcal{D}_2$ except with negligible probability. We say that $\mathcal{D}_1$ and $\mathcal{D}_2$ are statistically indistinguishable if the statistical distance between $\mathcal{D}_1$ and $\mathcal{D}_2$ is negligible; we denote this by writing $\mathcal{D}_1 \overset{s}{\approx} \mathcal{D}_2$. For an algorithm $\mathcal{A}$, we write $\mathcal{A}(x; r)$ to denote the output of running $\mathcal{A}$ on input x and randomness r. In settings where we do not need to specify the randomness explicitly, we write $\mathcal{A}(x)$ to denote the output distribution of $\mathcal{A}$ on input x where the randomness is drawn from the uniform distribution.

We also recall the Schwartz-Zippel lemma [Sch80, Zip79] and a standard "smudging" lemma:

Lemma 2.1 (Schwartz-Zippel [Sch80, Zip79])**.** *Let $f \in \mathbb{F}[x_1, \dots, x_n]$ be a multivariate polynomial of total degree at most d over $\mathbb{F}$, not identically zero. Then for any set $S \subseteq \mathbb{F}$,*

$$\Pr[f(\alpha_1, \dots, \alpha_n) = 0 \mid \alpha_1, \dots, \alpha_n \xleftarrow{\text{R}} S] \le d / |S| \,.$$

Lemma 2.2 (Smudging Lemma)**.** *Let B, B' be integers. Fix any value $|e_1| \le B'$ and sample $e_2 \xleftarrow{\text{R}} [-B, B]$. The statistical distance between the distributions of $e_1 + e_2$ and e_2 is at most B'/B.*

Discrete Gaussians and tail bounds. We also recall some preliminaries on the discrete Gaussian distribution. We refer to Peikert's survey [Pei16] for additional details and references. For a real value $s > 0$, the Gaussian function $\rho_s \colon \mathbb{R} \to \mathbb{R}^+$ with width s is the function $\rho_s(x) := \exp(-\pi x^2/s^2)$. The discrete Gaussian distribution $D_{\mathbb{Z},s}$ over $\mathbb{Z}$ with mean 0 and width s is the distribution where

$$\Pr[X = x : X \leftarrow D_{\mathbb{Z},s}] = \frac{\rho_s(x)}{\sum_{y \in \mathbb{Z}} \rho_s(y)}. \tag{2.1}$$

A real random variable X is *subgaussian* with parameter s if for every $t \geq 0$, $\Pr[|X| > t] \leq 2\exp(-\pi t^2/s^2)$. The following two facts will be useful in our analysis.

- If X is subgaussian with parameter s and $a \in \mathbb{R}$, then aX is subgaussian with parameter $|a|\, s$.
- If $X_1, \ldots, X_m$ are independent subgaussian random variables with parameters $s_1, \ldots, s_m$, respectively, then $\sum_{i \in [m]} X_i$ is subgaussian with parameter $\|\mathbf{s}\|_2$ where $\mathbf{s} = (s_1, \ldots, s_m)$.

Rank-1 constraint satisfiability. We recall the definition of the R1CS language introduced implicitly by Gennaro et al. [GGPR13] and formalized explicitly in [SBV+13, BCG+13, BCR+19]:

Definition 2.3 (Rank-1 Constraint Satisfiability [GGPR13, SBV+13, BCR+19])**.** A rank-1 constraint satisfiability (R1CS) system over a finite field $\mathbb{F}$ is specified by a tuple $\mathcal{CS} = \big(n, N_g, N_w, \{\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i\}_{i \in [N_g]}\big)$ where $n, N_g, N_w \in \mathbb{N}$, $n \leq N_w$, and $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i \in \mathbb{F}^{N_w + 1}$. The system $\mathcal{CS}$ is *satisfiable* for a *statement* $\mathbf{x} \in \mathbb{F}^n$ if there exists a *witness* $\mathbf{w} \in \mathbb{F}^{N_w}$ such that

- $\mathbf{x} = (w_1, \ldots, w_n)$ and
- $[1 \mid \mathbf{w}^\mathsf{T}]\mathbf{a}_i \cdot [1 \mid \mathbf{w}^\mathsf{T}]\mathbf{b}_i = [1 \mid \mathbf{w}^\mathsf{T}]\mathbf{c}_i$ for all $i \in [N_g]$.

We denote this by writing $\mathcal{CS}(\mathbf{x}, \mathbf{w}) = 1$, and refer to n as the statement size, N_w as the number of variables, and N_g as the number of constraints. Given an R1CS system $\mathcal{CS}$, we define the corresponding relation $\mathcal{R}_{\mathcal{CS}} = \{(\mathbf{x}, \mathbf{w}) \in \mathbb{F}^n \times \mathbb{F}^{N_w} : \mathcal{CS}(\mathbf{x}, \mathbf{w}) = 1\}$.

Remark 2.4 (Boolean and Arithmetic Circuit Satisfiability)**.** As shown in [GGPR13, BCG+13], the language of R1CS capture Boolean and arithmetic circuit satisfiability as special cases. Namely, a Boolean circuit satisfiability instance for a Boolean circuit $C \colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ with α wires and β bilinear gates yields an R1CS instance with $N_w = \alpha$ variables and $N_g = \beta + h + 1$ constraints. Similarly, an arithmetic circuit $C \colon \mathbb{F}^n \times \mathbb{F}^h \to \mathbb{F}^\ell$ with α wires and β bilinear gates corresponds to an R1CS instance with $N_w = \alpha$ variables and $N_g = \beta + \ell$ constraints. In this work, we focus exclusively on linear PCPs and SNARKs for R1CS.

2.1 Linear PCPs

We now recall the notion of a linear PCP (LPCP) from [IKO07, BCI+13]. In this work, we only consider linear PCPs for R1CS systems, so we specialize all of our definitions to this setting:

Definition 2.5 (Linear PCP [IKO07, BCI+13, adapted])**.** Let p be a polynomial and let $\mathcal{CS} = \{\mathcal{CS}_\kappa\}_{\kappa \in \mathbb{N}}$ be a family of R1CS systems over a finite field $\mathbb{F}$ where each system $\mathcal{CS}_\kappa = \big(n_\kappa, N_{g,\kappa}, N_{w,\kappa}, \{\mathbf{a}_{i,\kappa}, \mathbf{b}_{i,\kappa}, \mathbf{c}_{i,\kappa}\}_{i \in [N_{g,\kappa}]}\big)$ has size at most $|\mathcal{CS}_\kappa| \leq p(\kappa)$. In the following, we write $n = n(\kappa)$ to denote a polynomially-bounded function where $n(\kappa) = n_\kappa$ for all $\kappa \in \mathbb{N}$. We define $N_g = N_g(\kappa)$ and $N_w = N_w(\kappa)$ similarly. A k-query input-independent linear PCP for $\mathcal{CS}$ with query length $\ell = \ell(\kappa)$ and knowledge error $\varepsilon = \varepsilon(\kappa)$ is a tuple of algorithms $\Pi_{\mathsf{LPCP}} = (\mathcal{Q}_{\mathsf{LPCP}}, \mathcal{P}_{\mathsf{LPCP}}, \mathcal{V}_{\mathsf{LPCP}})$ with the following properties:

- $\mathcal{Q}_{\mathsf{LPCP}}(1^\kappa) \to (\mathsf{st}, \mathbf{Q})$: The query-generation algorithm takes as input the system index $\kappa \in \mathbb{N}$ and outputs a query matrix $\mathbf{Q} \in \mathbb{F}^{\ell \times k}$ and a verification state st.

- $\mathcal{P}_{\mathsf{LPCP}}(1^\kappa, \mathbf{x}, \mathbf{w}) \to \boldsymbol{\pi}$: On input the system index $\kappa \in \mathbb{N}$, a statement $\mathbf{x} \in \mathbb{F}^n$, and a witness $\mathbf{w} \in \mathbb{F}^{N_w}$, the prove algorithm outputs a proof $\boldsymbol{\pi} \in \mathbb{F}^\ell$.
- $\mathcal{V}_{\mathsf{LPCP}}(\mathsf{st}, \mathbf{x}, \mathbf{a})$: On input the verification state st, the statement $\mathbf{x} \in \mathbb{F}^n$, and a vector of responses $\mathbf{a} \in \mathbb{F}^k$, the verification algorithm outputs a bit $b \in \{0, 1\}$.

In addition, Π_{LPCP} should satisfy the following properties:

- **Completeness:** For all $\kappa \in \mathbb{N}$, $\mathbf{x} \in \mathbb{F}^n$, and $\mathbf{w} \in \mathbb{F}^{N_w}$ where $\mathcal{CS}_\kappa(\mathbf{x}, \mathbf{w}) = 1$,

$$\Pr[\mathcal{V}_{\mathsf{LPCP}}(\mathsf{st}, \mathbf{x}, \mathbf{Q}^\mathsf{T}\boldsymbol{\pi}) = 1 \mid (\mathsf{st}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}(1^\kappa), \boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(1^\kappa, \mathbf{x}, \mathbf{w})] = 1.$$

- **Knowledge:** There exists an efficient extractor $\mathcal{E}_{\mathsf{LPCP}}$ such that for all $\kappa \in \mathbb{N}$, $\mathbf{x} \in \mathbb{F}^n$, and $\boldsymbol{\pi}^* \in \mathbb{F}^\ell$, if

$$\Pr[\mathcal{V}_{\mathsf{LPCP}}(\mathsf{st}, \mathbf{x}, \mathbf{Q}^\mathsf{T}\boldsymbol{\pi}^*) = 1 \mid (\mathsf{st}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}(1^\kappa)] > \varepsilon,$$

then

$$\Pr[\mathcal{CS}_\kappa(\mathbf{x}, \mathbf{w}) = 1 \mid \mathbf{w} \leftarrow \mathcal{E}_{\mathsf{LPCP}}^{\langle \boldsymbol{\pi}^*, \cdot \rangle}(1^\kappa, \mathbf{x})] = 1.$$

We refer to ε as the *knowledge error* of the linear PCP.

- **Perfect honest-verifier zero knowledge (HVZK):** There exists an efficient simulator $\mathcal{S}_{\mathsf{LPCP}} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for all $\kappa \in \mathbb{N}$ and all instances $(\mathbf{x}, \mathbf{w})$ where $\mathcal{CS}_\kappa(\mathbf{x}, \mathbf{w}) = 1$,

$$\{(\mathsf{st}, \mathbf{Q}, \mathbf{Q}^\mathsf{T}\boldsymbol{\pi})\} \equiv \{(\widetilde{\mathsf{st}}, \tilde{\mathbf{Q}}, \tilde{\mathbf{a}})\},$$

where $(\mathsf{st}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}(1^\kappa)$, $\boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(1^\kappa, \mathbf{x}, \mathbf{w})$, $(\widetilde{\mathsf{st}}, \tilde{\mathbf{Q}}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}_1(1^\kappa)$, and $\tilde{\mathbf{a}} \leftarrow \mathcal{S}_2(\mathsf{st}_{\mathcal{S}}, \mathbf{x})$.[3]

Linear PCPs for R1CS. The quadratic arithmetic programs (QAPs) introduced by Gennaro et al. [GGPR13] immediately imply a 4-query linear PCP for R1CS [BCG+13]. Note that Ben-Sasson et al. [BCG+13] described the construction as a 5-query linear PCP with statistical HVZK (over large fields); however, it is straightforward to adapt the construction to obtain a 4-query LPCP with perfect HVZK (over *any* field). These changes incur a slight increase in the verification complexity and the knowledge error. We state the main result below and describe the construction from [BCG+13] and our modifications in Appendix A.

Claim 2.6 (Linear PCPs for R1CS [GGPR13, BCG+13, adapted])**.** *Let $\mathcal{CS} = \{\mathcal{CS}_\kappa\}_{\kappa \in \mathbb{N}}$ be a family of R1CS instances over a finite field $\mathbb{F}$, where $\mathcal{CS}_\kappa = \big(n_\kappa, N_{g,\kappa}, N_{w,\kappa}, \{\mathbf{a}_{i,\kappa}, \mathbf{b}_{i,\kappa}, \mathbf{c}_{i,\kappa}\}_{i \in [N_{g,\kappa}]}\big)$. We write $n = n(\kappa)$ to denote a function where $n(\kappa) = n_\kappa$ for all $\kappa \in \mathbb{N}$; we define $N_g = N_g(\kappa)$ and $N_w = N_w(\kappa)$ correspondingly. Then, there exists a 4-query linear PCP for $\mathcal{CS}$ with knowledge error $2N_g/(|\mathbb{F}| - N_g)$, query length $4 + N_w + N_g - n$, and satisfying perfect HVZK.*

Remark 2.7 (Knowledge Amplification for Linear PCPs)**.** Claim 2.6 gives a 4-query linear PCP for any R1CS system with N_g constraints that has knowledge error $\varepsilon = 2N_g/(|\mathbb{F}| - N_g)$. To achieve negligible knowledge error, this necessitates working over a field of super-polynomial size. In our lattice-based instantiation, it is more efficient to work over smaller fields. To amplify knowledge, we use standard parallel repetition. Namely, for a k-query LPCP with query length m and knowledge error ε, we can obtain a $(k\rho)$-query LPCP with the same query length and knowledge error ε^ρ. In more detail, the setup algorithm samples ρ independent sets of queries $\mathbf{Q}_1, \ldots, \mathbf{Q}_\rho \in \mathbb{F}^{m \times k}$ and constructs its query matrix $\mathbf{Q}$ as $\mathbf{Q} = [\mathbf{Q}_1 \mid \cdots \mid \mathbf{Q}_\rho] \in \mathbb{F}^{m \times k\rho}$. The verifier accepts a response $\mathbf{a} = [\mathbf{a}_1 \mid \cdots \mid \mathbf{a}_\rho]$ only if all ρ sets of responses are valid.

[3] This definition separates the simulator $\mathcal{S}_{\mathsf{LPCP}}$ into a *statement-independent* algorithm $\mathcal{S}_1$ that simulates the query matrix and a *statement-dependent* algorithm $\mathcal{S}_2$. This separation is important for arguing *multi-theorem* zero knowledge of our zkSNARKs. In the multi-theorem setting, the verifier's query matrix is *reused* for multiple proofs.

2.2 Succinct Non-Interactive Arguments

We recall the definitions of a succinct non-interactive argument of knowledge (SNARK) for R1CS:

Definition 2.8 (Succinct Non-Interactive Argument of Knowledge)**.** Let $\mathcal{CS} = \{\mathcal{CS}_\kappa\}_{\kappa \in \mathbb{N}}$ be a family of R1CS systems over a finite field $\mathbb{F}$, where $|\mathcal{CS}_\kappa| \le s(\kappa)$ for some fixed polynomial $s(\cdot)$. A succinct non-interactive argument (SNARK) in the preprocessing model[4] for $\mathcal{CS}$ is a tuple $\Pi_{\mathsf{SNARK}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ with the following properties:

- $\mathsf{Setup}(1^\lambda, 1^\kappa) \to (\mathsf{crs}, \mathsf{st})$: On input the security parameter λ and the system index κ, the setup algorithm outputs a common reference string crs and verification state st.
- $\mathsf{Prove}(\mathsf{crs}, \mathbf{x}, \mathbf{w}) \to \pi$: On input a common reference string crs, a statement $\mathbf{x}$, and a witness $\mathbf{w}$, the prove algorithm outputs a proof π.
- $\mathsf{Verify}(\mathsf{st}, \mathbf{x}, \pi) \to \{0,1\}$: On input the verification state st, a statement $\mathbf{x}$ and a proof π, the verification algorithm outputs a bit $b \in \{0,1\}$.

Moreover, Π_{SNARK} should satisfy the following properties:

- **Completeness:** For all security parameters $\lambda \in \mathbb{N}$, system indices $\kappa \in \mathbb{N}$, and instances $(\mathbf{x}, \mathbf{w})$ where $\mathcal{CS}_\kappa(\mathbf{x}, \mathbf{w}) = 1$,
$$\Pr[\mathsf{Verify}(\mathsf{st}, \mathbf{x}, \pi) = 1] = 1,$$
where $(\mathsf{crs}, \mathsf{st}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa), \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathbf{x}, \mathbf{w})$.
- **Knowledge:** For all polynomial-size provers $\mathcal{P}^*$, there exists a polynomial-size extractor $\mathcal{E}$ such that for all security parameters $\lambda \in \mathbb{N}$, system indices $\kappa \in \mathbb{N}$, and auxiliary inputs $z \in \{0,1\}^{\mathsf{poly}(\lambda)}$,
$$\Pr[\mathsf{Verify}(\mathsf{st}, \mathbf{x}, \pi) = 1 \wedge \mathcal{CS}_\kappa(\mathbf{x}, \mathbf{w}) \ne 1] = \mathsf{negl}(\lambda),$$
where $(\mathsf{crs}, \mathsf{st}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa)$, $(\mathbf{x}, \pi) \leftarrow \mathcal{P}^*(1^\lambda, 1^\kappa, \mathsf{crs}; z)$, and $\mathbf{w} \leftarrow \mathcal{E}(1^\lambda, 1^\kappa, \mathsf{crs}, \mathsf{st}, \mathbf{x}; z)$.
- **Efficiency:** There exist a universal polynomial p (independent of $\mathcal{CS}$) such that Setup and Prove run in time $p(\lambda + |\mathcal{CS}_\kappa|)$, Verify runs in time $p(\lambda + |\mathbf{x}| + \log|\mathcal{CS}_\kappa|)$, and the proof size is $p(\lambda + \log|\mathcal{CS}_\kappa|)$.

Remark 2.9 (Public Verification vs. Designated Verifier)**.** We say a SNARK is *publicly-verifiable* if st can be efficiently computed from crs (i.e., verification only depends on the public common reference string). Otherwise, the SNARK is *designated-verifier* (i.e., only the holder of the *secret* verification state st can check proofs). In this work, we focus on designated-verifier SNARKs.

Definition 2.10 (Zero Knowledge)**.** A SNARK $\Pi_{\mathsf{SNARK}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ for an R1CS system $\mathcal{CS} = \{\mathcal{CS}_\kappa\}_{\kappa \in \mathbb{N}}$ is computational zero knowledge (i.e., a zkSNARK) if there exists an efficient simulator $\mathcal{S}_{\mathsf{SNARK}} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for all $\kappa \in \mathbb{N}$ and all efficient and stateful adversaries $\mathcal{A}$, we have that

$$\Pr[\mathsf{ExptZK}_{\Pi_{\mathsf{SNARK}}, \mathcal{A}, \mathcal{S}_{\mathsf{SNARK}}}(1^\lambda, 1^\kappa) = 1] \le 1/2 + \mathsf{negl}(\lambda), \tag{2.2}$$

where the experiment $\mathsf{ExptZK}_{\Pi_{\mathsf{SNARK}}, \mathcal{A}, \mathcal{S}_{\mathsf{SNARK}}}(1^\lambda, 1^\kappa)$ is defined as follows:

1. The challenger samples $b \xleftarrow{\mathrm{R}} \{0,1\}$. If $b = 0$, the challenger computes $(\mathsf{crs}, \mathsf{st}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa)$ and gives $(\mathsf{crs}, \mathsf{st})$ to $\mathcal{A}$. If $b = 1$, the challenger computes $(\widetilde{\mathsf{crs}}, \widetilde{\mathsf{st}}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}_1(1^\lambda, 1^\kappa)$ and gives $(\widetilde{\mathsf{crs}}, \widetilde{\mathsf{st}})$ to $\mathcal{A}$.
2. The adversary $\mathcal{A}$ outputs a statement $\mathbf{x}$ and a witness $\mathbf{w}$.

[4] In the preprocessing model, we allow for a *statement-independent* setup algorithm that runs in time polynomial in the size of the instance $\mathcal{CS}_\kappa$. In contrast, a "fully-succinct" SNARK also requires that the setup run in time sublinear (or polylogarithmic) in the size of $\mathcal{CS}_\kappa$. Using recursive composition [BCCT13], it is possible to obtain fully succinct SNARKs from preprocessing SNARKs.

3. If $\mathcal{S}_\kappa(\mathbf{x}, \mathbf{w}) \neq 1$, then the experiment halts with output 0. Otherwise, the challenger proceeds as follows:
   - If $b = 0$, the challenger replies with $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathbf{x}, \mathbf{w})$.
   - If $b = 1$, the challenger replies with $\tilde{\pi} \leftarrow \mathcal{S}_2(\mathsf{st}_{\mathcal{S}}, \mathbf{x})$.

   At the end of the experiment, $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$. The output of the experiment is 1 if $b' = b$ and is 0 otherwise.

When the probability in Eq. (2.2) is bounded by $1/2 + \varepsilon$, we say that the scheme satisfies ε-computational zero knowledge.

3 Lattice-Based Succinct Arguments

In this section, we introduce the main information-theoretic building block (linear PCPs over extension fields) and the cryptographic compiler (linear-only vector encryption) that underlie our lattice-based zkSNARK. We then show how to combine these ingredients to obtain our designated-verifier zkSNARK by invoking the Bitansky et al. [BCI+13, BISW17] compiler (see Section 1.2).

3.1 Linear PCPs over Extension Fields

Claim 2.6 gives a linear PCP for R1CS over any (sufficiently-large) field $\mathbb{F}$. In our work, we consider linear PCPs over quadratic extensions $\mathbb{F}_{p^2}$. As discussed in Section 1.2, we consider compilers based on vector encryption over the extension $\mathbb{F}_{p^2}$ as well as over the base field $\mathbb{F}_p$. For the latter setting, we need to first transform a linear PCP over $\mathbb{F}_{p^2}$ to a linear PCP over $\mathbb{F}_p$. We describe this transformation here.

Field extensions. Recall that a degree-d field extension $\mathbb{F}_{p^d}$ of $\mathbb{F}_p$ is a d-dimensional vector space over $\mathbb{F}_p$. For a field element $s \in \mathbb{F}_{p^d}$, we write $\mathbf{v}_s \in \mathbb{F}_p^d$ to denote its representation in $\mathbb{F}_p^d$. There is an efficiently-computable isomorphism between $s \in \mathbb{F}_{p^d}$ and $\mathbf{v}_s \in \mathbb{F}_p^d$. In particular, this means that for all $s, t \in \mathbb{F}_{p^d}$, $\mathbf{v}_s + \mathbf{v}_t = \mathbf{v}_{s+t} \in \mathbb{F}_p^d$. We write $\mathbf{M}_s \in \mathbb{F}_p^{d \times d}$ to denote the linear transformation over $\mathbb{F}_p^d$ corresponding to scalar multiplication by s over $\mathbb{F}_{p^d}$. Namely, for all $s, t \in \mathbb{F}_{p^d}$, we have that $\mathbf{M}_s \mathbf{v}_t = \mathbf{v}_{st}$.

Construction 3.1 ($\mathbb{F}_{p^d}$-Linear PCP to $\mathbb{F}_p$-Linear PCP)**.** Let $\Pi'_{\mathsf{LPCP}} = (\mathcal{Q}'_{\mathsf{LPCP}}, \mathcal{P}'_{\mathsf{LPCP}}, \mathcal{V}'_{\mathsf{LPCP}})$ be a k-query linear PCP for a family of R1CS systems $\mathcal{CS} = \{\mathcal{CS}_\kappa\}_{\kappa \in \mathbb{N}}$ over an extension field $\mathbb{F}_{p^d}$ with query length ℓ. We construct a (dk)-query linear PCP $\Pi_{\mathsf{LPCP}} = (\mathcal{Q}_{\mathsf{LPCP}}, \mathcal{P}_{\mathsf{LPCP}}, \mathcal{V}_{\mathsf{LPCP}})$ for $\mathcal{CS}$ with query length $d\ell$ over the base field $\mathbb{F}_p$:

- $\mathcal{Q}_{\mathsf{LPCP}}(1^\kappa)$: Run $(\mathsf{st}, \mathbf{Q}') \leftarrow \mathcal{Q}'_{\mathsf{LPCP}}(1^\kappa)$, where $\mathbf{Q}' \in \mathbb{F}_{p^d}^{\ell \times k}$. Let $\mathbf{Q} \in \mathbb{F}_p^{d\ell \times dk}$ be the matrix formed by taking each component $q'_{i,j} \in \mathbb{F}_{p^d}$ in $\mathbf{Q}'$ and replacing it with the transpose of the "multiplication-by-$q'_{i,j}$" matrix $\mathbf{M}_{q'_{i,j}}^{\mathsf{T}} \in \mathbb{F}_p^{d \times d}$. Namely,

$$\mathbf{Q}' = \begin{bmatrix} q'_{1,1} & \cdots & q'_{1,k} \\ \vdots & \ddots & \vdots \\ q'_{\ell,1} & \cdots & q'_{\ell,k} \end{bmatrix} \in \mathbb{F}_{p^d}^{\ell \times k} \text{ and } \mathbf{Q} = \begin{bmatrix} \mathbf{M}_{q'_{1,1}}^{\mathsf{T}} & \cdots & \mathbf{M}_{q'_{1,k}}^{\mathsf{T}} \\ \vdots & \ddots & \vdots \\ \mathbf{M}_{q'_{\ell,1}}^{\mathsf{T}} & \cdots & \mathbf{M}_{q'_{\ell,k}}^{\mathsf{T}} \end{bmatrix} \in \mathbb{F}_p^{d\ell \times dk}.$$

  Output st and $\mathbf{Q}$.

- $\mathcal{P}_{\mathsf{LPCP}}(1^\kappa, \mathbf{x}, \mathbf{w})$: Compute $\boldsymbol{\pi}' \leftarrow \mathcal{P}'_{\mathsf{LPCP}}(1^\kappa, \mathbf{x}, \mathbf{w}) \in \mathbb{F}_{p^d}^{\ell}$. Let $\boldsymbol{\pi} \in \mathbb{F}_p^{d\ell}$ be the vector formed by taking each component $\pi'_i \in \mathbb{F}_{p^d}$ in $\boldsymbol{\pi}'$ and replacing it with the vector $\mathbf{v}_{\pi'_i} \in \mathbb{F}_p^d$ representing π_i. Output the proof vector $\boldsymbol{\pi}$.

- $\mathcal{V}_{\mathsf{LPCP}}(\mathsf{st}, \mathbf{x}, \mathbf{a})$: First, parse $\mathbf{a} \in \mathbb{F}_p^{dk}$ as $[\mathbf{v}_{a'_1} | \cdots | \mathbf{v}_{a'_k}]$ for some $\mathbf{a}' = (a'_1, \ldots, a'_k) \in \mathbb{F}_{p^d}^k$. Output $\mathcal{V}'_{\mathsf{LPCP}}(\mathsf{st}, \mathbf{x}, \mathbf{a}')$.

Theorem 3.2 ($\mathbb{F}_{p^d}$-Linear PCP to $\mathbb{F}_p$-Linear PCP)**.** *If Π'_{LPCP} is complete, perfect HVZK, and has knowledge error ε, then the same holds for Π_{LPCP} from Construction 3.1.*

Proof. We analyze each property individually:

- **Completeness:** Take any $\mathbf{x}, \mathbf{w}$ where $\mathcal{R}(\mathbf{x}, \mathbf{w}) = 1$, and let $(\mathsf{st}, \mathbf{Q}') \leftarrow \mathcal{Q}'_{\mathsf{LPCP}}(1^\kappa)$, $\boldsymbol{\pi}' \leftarrow \mathcal{P}'_{\mathsf{LPCP}}(1^\kappa, \mathbf{x}, \mathbf{w})$. Let $\mathbf{Q} \in \mathbb{F}_p^{d\ell \times dk}$ and $\boldsymbol{\pi} \in \mathbb{F}_p^{d\ell}$ be as specified in $\mathcal{Q}_{\mathsf{LPCP}}$ and $\mathcal{P}_{\mathsf{LPCP}}$ and let $\mathbf{a} \leftarrow \mathbf{Q}^\mathsf{T} \boldsymbol{\pi}$. Write $\mathbf{a} = [\mathbf{a}_1, \ldots, \mathbf{a}_k]$. By construction, for all $i \in [k]$,

$$\mathbf{a}_i = \sum_{j \in [\ell]} \mathbf{M}_{q'_{j,i}} \mathbf{v}_{\pi'_j} = \sum_{j \in [\ell]} \mathbf{v}_{q'_{j,i} \pi'_j} = \mathbf{v}_{(\mathbf{q}'_i)^\mathsf{T} \boldsymbol{\pi}'} \in \mathbb{F}_p^d,$$

  where $\mathbf{q}'_i \in \mathbb{F}_{p^d}^\ell$ denotes the i^{th} column of $\mathbf{Q}'$. This means that the vector $\mathbf{a}'$ computed by $\mathcal{V}_{\mathsf{LPCP}}$ satisfies $\mathbf{a}' = (\mathbf{Q}')^\mathsf{T} \boldsymbol{\pi}'$. Completeness now follows by completeness of Π'_{LPCP}.

- **Knowledge:** Let $(\mathsf{st}, \mathbf{Q}') \leftarrow \mathcal{Q}'_{\mathsf{LPCP}}(1^\kappa)$ and let $\mathbf{Q}$ be the matrix $\mathcal{Q}_{\mathsf{LPCP}}$ constructs from $\mathbf{Q}'$. Take any proof $\boldsymbol{\pi} \in \mathbb{F}_p^{d\ell}$, and let $\boldsymbol{\pi}' \in \mathbb{F}_{p^d}^\ell$ be the vector obtained by viewing each contiguous block of d elements of $\boldsymbol{\pi}$ as an element of $\mathbb{F}_{p^d}$. By construction, $\mathcal{V}_{\mathsf{LPCP}}(\mathsf{st}, \mathbf{x}, \mathbf{Q}^\mathsf{T} \boldsymbol{\pi}) = 1$ if and only if $\mathcal{V}'_{\mathsf{LPCP}}(\mathsf{st}, \mathbf{x}, (\mathbf{Q}')^\mathsf{T} \boldsymbol{\pi}') = 1$. The claim now follows by knowledge soundness of Π'_{LPCP}. Namely, the extractor $\mathcal{E}_{\mathsf{LPCP}}$ for Π_{LPCP} simply invokes the extractor $\mathcal{E}'_{\mathsf{LPCP}}$ for Π'_{LPCP}. Any linear query $\mathbf{q}' \in \mathbb{F}_{p^d}^\ell$ that $\mathcal{E}'_{\mathsf{LPCP}}$ makes to $\langle \boldsymbol{\pi}', \cdot \rangle$ can be simulated via d linear queries to $\langle \boldsymbol{\pi}, \cdot \rangle$ by expanding each component in $\mathbf{q}'$ into a matrix over $\mathbb{F}_p^{d \times d}$.

- **Perfect HVZK:** Let $\mathcal{S}'_{\mathsf{LPCP}} = (\mathcal{S}'_{\mathsf{LPCP},1}, \mathcal{S}'_{\mathsf{LPCP},2})$ be the linear PCP simulator for Π'_{LPCP}. We define $\mathcal{S}_{\mathsf{LPCP}} = (\mathcal{S}_{\mathsf{LPCP},1}, \mathcal{S}_{\mathsf{LPCP},2})$ for Π_{LPCP} as follows:
  - $\mathcal{S}_{\mathsf{LPCP},1}(1^\kappa)$: On input $\kappa \in \mathbb{N}$, run the simulator $\mathcal{S}'_{\mathsf{LPCP},1}(1^\kappa)$ to obtain a pair $(\mathsf{st}, \mathbf{Q}')$ where $\mathbf{Q}' \in \mathbb{F}_{p^d}^{\ell \times k}$. The simulator constructs $\mathbf{Q} \in \mathbb{F}_p^{d\ell \times dk}$ by expanding replacing each component $q'_{i,j}$ of $\mathbf{Q}'$ with $\mathbf{M}'^\mathsf{T}_{q_{i,j}}$ (as in $\mathcal{Q}_{\mathsf{LPCP}}$). It outputs $(\mathsf{st}, \mathbf{Q})$.
  - $\mathcal{S}_{\mathsf{LPCP},2}(\mathsf{st}, \mathbf{x})$: On input the simulation state st and a statement $\mathbf{x}$, run $\mathcal{S}'_{\mathsf{LPCP},2}(\mathsf{st}, \mathbf{x})$ to obtain $\mathbf{a}' \in \mathbb{F}_{p^d}^k$. Then, compute and output $\mathbf{a} \in \mathbb{F}_p^{dk}$ by expanding each component $\pi'_i \in \mathbb{F}_{p^d}$ as a vector $\mathbf{v}_{\pi'_i} \in \mathbb{F}_p^d$ (as in $\mathcal{P}_{\mathsf{LPCP}}$).

  Perfect HVZK now follows by perfect HVZK of Π'_{LPCP}. □

3.2 Linear-Only Vector Encryption

We begin with the definition of a vector encryption scheme (adapted from [BISW17]), and then define the linear-only [BCI+13, BISW17] property we rely on for our zkSNARK constructions.

Definition 3.3 (Vector Encryption)**.** Let $\mathbb{F}$ be a finite field. A secret-key additively-homomorphic vector encryption scheme over a vector space $\mathbb{F}^\ell$ consists of a tuple of algorithms $\Pi_{\mathsf{Enc}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Add})$ with the following properties:

- $\mathsf{Setup}(1^\lambda, 1^\ell) \to (\mathsf{pp}, \mathsf{sk})$: On input the security parameter λ and the plaintext dimension ℓ, the setup algorithm outputs public parameters pp and a secret key sk.

- $\mathsf{Encrypt}(\mathsf{sk}, \mathbf{v}) \to \mathsf{ct}$: On input the secret key sk and a vector $\mathbf{v} \in \mathbb{F}^\ell$, the encryption algorithm outputs a ciphertext ct.

- $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}) \to \mathbf{v}/\bot$: On input the secret key sk and a ciphertext ct, the decryption algorithm either outputs a vector $\mathbf{v} \in \mathbb{F}^\ell$ or a special symbol $\bot$.

- $\mathsf{Add}(\mathsf{pp}, \{\mathsf{ct}_i\}_{i \in [n]}, \{c_i\}_{i \in [n]}) \to \mathsf{ct}^*$: On input the public parameters, a collection of ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ and scalars $c_1, \ldots, c_n \in \mathbb{F}$, the addition algorithm outputs a new ciphertext ct^*.

Moreover, Π_{Enc} should satisfy the following properties:

- **Additive homomorphism:** For all security parameters $\lambda \in \mathbb{N}$, vectors $\mathbf{v}_1, \dots, \mathbf{v}_k \in \mathbb{F}^\ell$, and scalars $y_1, \dots, y_k \in \mathbb{F}$, where $k = k(\lambda)$,

$$\Pr\left[\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}^*) = \sum_{i \in [k]} y_i \mathbf{v}_i\right] = 1 - \mathsf{negl}(\lambda), \tag{3.1}$$

  where $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{v}_i)$ for all $i \in [k]$, and $\mathsf{ct}^* \leftarrow \mathsf{Add}(\mathsf{pp}, \{\mathsf{ct}_i\}_{i\in[k]}, \{y_i\}_{i\in[k]})$. We say that Π_{Enc} is additively homomorphic with respect to a set $S \subseteq R_p^k$ if Eq. (3.1) holds for all $(y_1, \dots, y_k) \in S$. Note that additive homomorphism implies correctness of decryption.

- **CPA security:** For all security parameters $\lambda \in \mathbb{N}$ and all efficient adversaries $\mathcal{A}$,

$$\Pr\left[\mathcal{A}^{\mathcal{O}_b(\mathsf{sk},\cdot,\cdot)}(1^\lambda, \mathsf{pp}) = b\right] = 1/2 + \mathsf{negl}(\lambda), \tag{3.2}$$

  where $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, $b \xleftarrow{\text{R}} \{0,1\}$, and oracle $\mathcal{O}_b$ takes inputs $(\mathsf{sk}, \mathbf{v}_0, \mathbf{v}_1)$ and outputs $\mathsf{ct}_b \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{v}_b)$. If Eq. (3.2) holds against all efficient adversaries $\mathcal{A}$ making at most Q queries to $\mathcal{O}_b$, then we say Π_{Enc} is Q-query CPA secure.

Definition 3.4 (Linear-Only Vector Encryption [BCI+13, adapted])**.** A vector encryption scheme $\Pi_{\mathsf{Enc}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Add})$ over $\mathbb{F}^\ell$ is *strictly linear-only* if for all polynomial-size adversaries $\mathcal{A}$, there is a polynomial-size extractor $\mathcal{E}$ such that for all security parameters $\lambda \in \mathbb{N}$, auxiliary inputs $z \in \{0,1\}^{\mathsf{poly}(\lambda)}$, and any efficient plaintext generator $\mathcal{M}$,

$$\Pr[\mathsf{ExptLinearExt}_{\Pi_{\mathsf{Enc}},\mathcal{A},\mathcal{M},\mathcal{E},z}(1^\lambda) = 1] = \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{ExptLinearExt}_{\Pi_{\mathsf{Enc}},\mathcal{A},\mathcal{M},\mathcal{E},z}(1^\lambda)$ is defined as follows:

1. The challenger starts by sampling $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$ and $(\mathbf{v}_1, \dots, \mathbf{v}_m) \leftarrow \mathcal{M}(1^\lambda, \mathsf{pp})$. It computes $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{v}_i)$ for each $i \in [m]$ and runs $\mathcal{A}(\mathsf{pp}, \mathsf{ct}_1, \dots, \mathsf{ct}_m; z)$ to obtain a tuple $(\mathsf{ct}'_1, \dots, \mathsf{ct}'_k)$.
2. The challenger computes $\mathbf{\Pi} \leftarrow \mathcal{E}(\mathsf{pp}, \mathsf{ct}_1, \dots, \mathsf{ct}_m; z)$ and $\mathbf{V}' \leftarrow \mathbf{\Pi} \cdot [\mathbf{v}_1 \mid \cdots \mid \mathbf{v}_m]^\mathsf{T}$, where $\mathbf{\Pi} \in \mathbb{F}^{k\times m}$ and $\mathbf{V}' \in \mathbb{F}^{k\times\ell}$. The experiment outputs 1 if there exists an index $i \in [k]$ such that $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}'_i) \neq \perp$ and $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}'_i) \neq \mathbf{v}'_i$, where $\mathbf{v}'_i \in \mathbb{F}^\ell$ is the i^{th} row of $\mathbf{V}'$. Otherwise, the experiments outputs 0.

Remark 3.5 (Linear vs. Affine Strategies)**.** Definition 3.4 requires that all adversarial strategies which produce a valid ciphertext correspond to taking a *linear* combination of the given ciphertexts. Previous definitions [BCI+13, BISW17] considered a weaker requirement that allows the extractor to explain the adversary's strategy using an *affine* function. Indeed, in the public-key setting, the encryption scheme can at best be affine-only, since the adversary can always encrypt an arbitrary vector $\mathbf{v}$ of its choosing (using the public key) and add it to the ciphertext. However, in the secret-key setting, it is plausible that the adversary cannot even produce new ciphertexts on messages of its choosing. In this case, we can conjecture that the only way the adversary can construct valid ciphertexts is by computing linear combinations of existing ciphertexts. To distinguish between our more stringent notion and the previous notion, we refer to ours as *strict* linear-only encryption. Making this stronger linear-only conjecture for a secret-key encryption scheme enables a direct compiler from a linear PCP with knowledge against linear strategies (as opposed to linear PCPs with knowledge against affine strategies; see Remark A.3 for further discussion).

Remark 3.6 (Auxiliary Input Distributions)**.** Definition 3.4 requires that the extractor succeed for *arbitrary* auxiliary inputs z. While this formulation is convenient for definitional purposes, the requirement may be too strong in certain settings (e.g., when z encodes a hard cryptographic problem that the extractor must solve to explain the adversary's behavior [BCPR14]). In such cases, it suffices to consider a relaxation where Definition 3.4 holds only for auxiliary inputs sampled from "benign" distributions (e.g., in our applications, it suffices to consider auxiliary inputs that are uniform). We refer to [BCPR14, BCI+13] for further discussion.

Circuit privacy. In addition to the above properties, we additionally require a *circuit privacy* property [Gen09]. Circuit privacy says that the ciphertext output by Add can be simulated given only the underlying plaintext value, *without* knowledge of the linear combination used to construct the ciphertext. This is important for arguing zero knowledge (see Section 3.4 and Theorem 3.23).

Definition 3.7 (Circuit Privacy)**.** Let $\Pi_{\mathsf{Enc}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Add})$ be a secret-key vector encryption scheme over $\mathbb{F}^\ell$. We say that Π_{Enc} satisfies circuit privacy if for all efficient and stateful adversaries $\mathcal{A}$, there exists an efficient simulator $\mathcal{S}$ such that for all security parameters $\lambda \in \mathbb{N}$,

$$\Pr[\mathsf{ExptCircuitPriv}_{\Pi_{\mathsf{Enc}},\mathcal{A},\mathcal{S}}(1^\lambda) = 1] = 1/2 + \mathsf{negl}(\lambda), \tag{3.3}$$

where the experiment $\mathsf{ExptCircuitPriv}_{\Pi_{\mathsf{Enc}},\mathcal{A},\mathcal{S}}(1^\lambda)$ is defined as follows:

1. The challenger samples $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$ and gives $(\mathsf{pp}, \mathsf{sk})$ to the adversary. The adversary replies with a collection of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathbb{F}^\ell$.
2. The challenger constructs ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{v}_i)$ for all $i \in [k]$ and gives $(\mathsf{ct}_1, \ldots, \mathsf{ct}_k)$ to $\mathcal{A}$. The adversary replies with a collection of coefficients $y_1, \ldots, y_k \in \mathbb{F}$.
3. The challenger computes $\mathsf{ct}_0^* \leftarrow \mathsf{Add}(\mathsf{pp}, \{\mathsf{ct}_i\}_{i\in[k]}, \{y_i\}_{i\in[k]})$ and $\mathsf{ct}_1^* \leftarrow \mathcal{S}(1^\lambda, \mathsf{pp}, \mathsf{sk}, \sum_{i\in[k]} y_i\mathbf{v}_i)$. It also samples a random bit $b \xleftarrow{\text{R}} \{0,1\}$ and replies to the adversary with ct_b^*.
4. The adversary outputs a bit $b' \in \{0,1\}$. The output of the experiment is 1 if $b' = b$ and 0 otherwise.

In this work, we also consider a weaker notion of circuit privacy where we additionally constrain the adversary to choosing the coefficients from an *a priori* specified set $S \subseteq \mathbb{F}$. In this case, we say that Π_{Enc} satisfies circuit privacy with respect to S. In addition, when the probability in Eq. (3.3) is bounded by $1/2 + \varepsilon$, we say that Π_{Enc} is ε-circuit private.

Remark 3.8 (Multi-Query Circuit Privacy)**.** We can define a multi-query variant of Definition 3.7 where the adversary can adaptively choose *multiple* collections of coefficients $y_1, \ldots, y_k \in R_p$ and on each query, the adversary learns either the homomorphically-evaluated ciphertext (from Add) or the simulated ciphertext (from $\mathcal{S}$). This multi-query notion is useful to argue *multi-theorem* zero knowledge when compiling a linear PCP into a preprocessing SNARG [BCI+13]. Definition 3.7 implies this multi-query variant by a standard hybrid argument.

3.3 Candidate Linear-Only Vector Encryption

Our constructions work over the ring $R = \mathbb{Z}[x]/(x^d + 1)$ where d is a power of 2. We specifically consider the cases where $d = 1$ ($R = \mathbb{Z}$) and $d = 2$ ($R = \mathbb{Z}[x]/(x^2 + 1)$). For a positive integer $p \in \mathbb{N}$, we write $R_p = R/pR$. We represent elements of R as a vector of coefficients (i.e., as a vector $\mathbb{Z}^d$). For an element $r \in R$, we write $\|r\|_\infty$ to denote the ℓ_∞ norm of the vector of coefficients of r. We write γ_R to denote the expansion constant where for all $r, s \in R$, we have that $\|rs\|_\infty \le \gamma_R \|r\|_\infty \|s\|_\infty$. In particular, $\gamma_R = 1$ when $d = 1$ and $\gamma_R = 2$ when $d = 2$. Finally, for a vector $\mathbf{v} \in R^n$, we write $\|\mathbf{v}\|_p$ to denote the ℓ_p norm $\|\mathbf{v}'\|_p$ of the vector $\mathbf{v}' \in \mathbb{Z}^{dn}$ formed by concatenating the vector of coefficients of each element in $\mathbf{v}$.

(Module) learning with errors. Security of our construction relies on the module learning with errors (MLWE) assumption [BGV12, LS15] (in addition to our linear-only conjecture). We state the MLWE assumption in "normal form" where the secret is sampled from the error distribution. This form of the problem is as hard as the version where the secret key is sampled uniformly at random [ACPS09].

Definition 3.9 (Module Learning With Errors (MLWE) [BGV12, LS15])**.** Fix a security parameter λ, integers $n = n(\lambda)$, $m = m(\lambda)$, $q = q(\lambda)$, $d = d(\lambda)$ where d is a power of two. Let $R = \mathbb{Z}[x]/(x^d + 1)$, $R_q = R/qR$, and $\chi = \chi(\lambda)$ be an error distribution over R_q. The (decisional) module learning with errors (MLWE) assumption

$\mathsf{MLWE}_{n,m,d,q,\chi}$ states that for $\mathbf{A} \xleftarrow{\text{R}} R_q^{n\times m}$, $\mathbf{s} \leftarrow \chi^n$, $\mathbf{e} \leftarrow \chi^m$, and $\mathbf{u} \xleftarrow{\text{R}} R_q^m$, the following two distributions are computationally indistinguishable:

$$(\mathbf{A}, \mathbf{s}^{\mathsf{T}}\mathbf{A} + \mathbf{e}^{\mathsf{T}}) \text{ and } (\mathbf{A}, \mathbf{u}^{\mathsf{T}})$$

Remark 3.10 (Relation to LWE and RLWE)**.** The module-LWE assumption generalizes both the classic learning with errors (LWE) assumption [Reg05] as well as the ring learning with errors (RLWE) assumption [LPR10]. In particular, LWE is MLWE instantiated with $d = 1$ and RLWE is MLWE instantiated with $n = 1$.

Vector encryption construction. We now describe our vector encryption scheme. Our scheme is an adaptation of the Regev-based [Reg05] scheme of Peikert et al. [PVW08], generalized to modules and with the following additions/modifications:

- **Secret-key encryption:** Since a secret-key vector encryption suffices for our designated-verifier zkSNARK,[5] we consider a secret-key version of the scheme. This reduces the concrete cost for encryption (we can substitute a random vector in each ciphertext in place of a matrix-vector product with the public key). Note that there are still public parameters in our scheme. These are used for *re-randomization* of homomorphically-evaluated ciphertexts, and are *not* used for encryption.

- **Message encoding:** We encode the message in the least significant bits of the ciphertext rather than the most significant bits. When the plaintext modulus p and ciphertext modulus q are coprime, these approaches are equivalent up to scaling [AP13]. In our implementation, encoding a value k in the least significant bits of the ciphertext is more convenient since we avoid the need to compute the value $\lfloor k \cdot q/p \rceil \bmod q$ (which if implemented improperly, can overflow our integer representation).

- **Ciphertext re-randomization:** For zero knowledge, we require an additional circuit privacy property. Ciphertexts in this scheme consist of pairs of vectors $\mathsf{ct} = (\mathbf{a}, \mathbf{c})$. Homomorphic operations on ciphertexts correspond to computing component-wise linear combinations. In our construction, we include a public MLWE matrix as part of the public parameters to re-randomize the vector $\mathbf{a}$, and we use standard noise smudging techniques (see Lemma 2.2) to re-randomize the vector $\mathbf{c}$. Previously, Gennaro et al. [GMNO18] suggest that the first component $\mathbf{a}$ is already random by appealing to the leftover hash lemma; unfortunately, this only applies in the setting where the coefficients of the linear combination have sufficient min-entropy (which is not necessarily the case in the zkSNARK construction). We show that in our case and under the MLWE assumption,[6] our construction provably satisfies circuit privacy without needing any additional assumption on the choice of linear combination.

- **Ciphertext sparsification.** Our linear-only definition (Definition 3.4) essentially requires that the only way an efficient adversary can generate a *valid* ciphertext is by taking linear combinations of valid ciphertexts. This means that the set of valid ciphertexts must be *sparse* (to prevent *oblivious* sampling of a valid ciphertext). Previous works [GGPR13, BCI+13, BISW17] suggest *double encryption* to realize this property. With double encryption, valid ciphertexts $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ are defined as pairs of ciphertexts that both encrypt identical messages. While this approach is applicable in our setting, it doubles the length of the ciphertexts.

  We propose a similar, but more efficient, approach tailored for vector encryption. Namely, if our goal is to encrypt elements from a vector space $\mathbb{F}^\ell$, we enlarge the plaintext space to $\mathbb{F}^{\ell+\tau}$, where τ is a sparsification parameter. During setup, we sample a random matrix $\mathbf{T} \xleftarrow{\text{R}} \mathbb{F}^{\ell\times\tau}$ which is included as part of the secret key. Then, to encrypt a vector $\mathbf{v} \in \mathbb{F}^\ell$, we instead encrypt the vector $\mathbf{u}^{\mathsf{T}} = [\mathbf{v}^{\mathsf{T}} \mid (\mathbf{T}\mathbf{v})^{\mathsf{T}}]$. During decryption, after recovering $\mathbf{u}^{\mathsf{T}} = [\mathbf{u}_1^{\mathsf{T}} \mid \mathbf{u}_2^{\mathsf{T}}]$, the decryption algorithm

[5] Using a public-key encryption scheme does *not* imply a publicly-verifiable zkSNARK in this setting. There is no advantage to using a public-key encryption scheme to instantiate the underlying encryption scheme.

[6] We could make this step statistical by relying on the leftover hash lemma, but this requires much larger parameters. Instead, we rely on MLWE and settle for computational circuit privacy (which translates to computational zero knowledge).

outputs $\perp$ if $\mathbf{u}_2 \neq \mathbf{T}\mathbf{u}_1$. Semantic security of the vector encryption scheme ensures that the secret transformation $\mathbf{T}$ is computationally hidden from the view of the adversary. By setting the sparsification parameter τ accordingly, we can ensure that for any fixed vector $\mathbf{u}^{\mathsf{T}} = [\mathbf{u}_1^{\mathsf{T}} \mid \mathbf{u}_2^{\mathsf{T}}]$, the probability that $\mathbf{u}_2 = \mathbf{T}\mathbf{u}_1$ is negligible (over the randomness of $\mathbf{T}$). We conjecture that our approach also yields an encryption scheme that satisfies the linear-only assumption. The advantage of this approach is that the ciphertext size in the underlying vector encryption scheme grows *additively* with the plaintext dimension (i.e., the resulting ciphertext size is $n + \ell + \tau$ rather than $2(n + \ell)$ as with "encrypting twice").

We now describe our vector encryption scheme:

Construction 3.11 (Vector Encryption)**.** Let $d = d(\lambda)$ be a power of two and let $R = \mathbb{Z}[x]/(x^d + 1)$. Fix lattice parameters $p = p(\lambda)$, $q = q(\lambda)$, $n = n(\lambda)$ and an error distribution $\chi = \chi(\lambda)$ over R_q. We additionally define the following parameters:

- ℓ: the plaintext dimension
- τ: the sparsification parameter
- B: the noise smudging bound

Let $\ell' = \ell + \tau$. We construct a secret-key vector encryption scheme $\Pi_{\mathsf{Enc}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Add})$ over R_p as follows:

- $\mathsf{Setup}(1^\lambda, 1^\ell)$: Sample matrices $\mathbf{A} \xleftarrow{\text{R}} R_q^{n \times n}$, $\mathbf{S} \leftarrow \chi^{n \times \ell'}$, $\mathbf{T} \xleftarrow{\text{R}} R_p^{\tau \times \ell}$, and $\mathbf{E} \leftarrow \chi^{n \times \ell'}$. Compute $\mathbf{D} \leftarrow \mathbf{S}^{\mathsf{T}}\mathbf{A} + p\mathbf{E}^{\mathsf{T}} \in R_q^{\ell' \times n}$. Output the secret key $\mathsf{sk} = (\mathbf{S}, \mathbf{T})$ and the public parameters $\mathsf{pp} = (\mathbf{A}, \mathbf{D})$.

- $\mathsf{Encrypt}(\mathsf{sk}, \mathbf{v})$: On input the secret key $\mathsf{sk} = (\mathbf{S}, \mathbf{T})$ and a vector $\mathbf{v} \in R_p^\ell$, construct the concatenated vector $\mathbf{u}^{\mathsf{T}} = [\mathbf{v}^{\mathsf{T}} \mid (\mathbf{T}\mathbf{v})^{\mathsf{T}}] \in R_p^{\ell'}$. Sample $\mathbf{a} \xleftarrow{\text{R}} R_q^n$, $\mathbf{e} \leftarrow \chi^{\ell'}$ and compute $\mathbf{c} \leftarrow \mathbf{S}^{\mathsf{T}}\mathbf{a} + p\mathbf{e} + \mathbf{u} \in R_q^{\ell'}$. Output the ciphertext $\mathsf{ct} = (\mathbf{a}, \mathbf{c})$.

- $\mathsf{Add}(\mathsf{pp}, \{\mathsf{ct}_i\}_{i \in [k]}, \{y_i\}_{i \in [k]})$: On input the public parameters $\mathsf{pp} = (\mathbf{A}, \mathbf{D})$, ciphertexts $\mathsf{ct}_i = (\mathbf{a}_i, \mathbf{c}_i)$ for $i \in [k]$, and scalars $y_i \in R_p$, sample $\mathbf{r} \leftarrow \chi^n$, $\mathbf{e}_a \leftarrow \chi^n$, $\mathbf{e}_c \xleftarrow{\text{R}} [-B, B]^{d\ell'}$ and output the ciphertext
$$\mathsf{ct}^* = \Big(\sum_{i \in [k]} y_i \mathbf{a}_i + \mathbf{A}\mathbf{r} + p\mathbf{e}_a, \ \sum_{i \in [k]} y_i \mathbf{c}_i + \mathbf{D}\mathbf{r} + p\mathbf{e}_c \Big). \tag{3.4}$$

- $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct})$: On input the secret key $\mathsf{sk} = (\mathbf{S}, \mathbf{T})$ and a ciphertext $\mathsf{ct} = (\mathbf{a}, \mathbf{c})$, compute $\mathbf{z} \leftarrow \mathbf{c} - \mathbf{S}^{\mathsf{T}}\mathbf{a} \in R_q^{\ell'}$. Compute $\mathbf{u} = \mathbf{z} \bmod p$, and parse $\mathbf{u}^{\mathsf{T}} = [\mathbf{v}_1^{\mathsf{T}} \mid \mathbf{v}_2^{\mathsf{T}}]$ where $\mathbf{v}_1 \in R_p^\ell$ and $\mathbf{v}_2 \in R_p^\tau$. Output $\mathbf{v}_1$ if $\mathbf{v}_2 = \mathbf{T}\mathbf{v}_1 \in R_p^\tau$ and $\perp$ otherwise.

Correctness and security analysis. Below, we state our main theorems on the correctness and security of Construction 3.11. We defer the formal analysis to Appendix B.1.

Theorem 3.12 (Additive Homomorphism)**.** *Let λ be a security parameter and p, q, n, ℓ', χ, B be as defined in Construction 3.11. Suppose χ is subgaussian*[7] *with parameter s. If $n, \ell', s, d, \gamma_R = \mathsf{poly}(\lambda)$, then for all $k = k(\lambda)$, there exists $q = (pB + kp^2) \cdot \mathsf{poly}(\lambda)$ such that Construction 3.11 is additively homomorphic with respect to R_p^k. Concretely, let C be a correctness parameter and let B_1, B_2 be bounds. Define the set $S = \{\mathbf{y} \in R_p^k : \|\mathbf{y}\|_1 \leq B_1$ and $\|\mathbf{y}\|_2 \leq B_2\}$. If $\ell', n > 8$, and*
$$q > 2p(B + \gamma_R B_2 C s + \gamma_R B_1/2 + 2\gamma_R n C^2 s^2) + p, \tag{3.5}$$
then Eq. (3.1) holds with probability $1 - (4n+2)d\ell' \exp(-\pi C^2)$ for all $(y_1, \ldots, y_k) \in S$.

[7] When $d > 1$, we assume that χ is the concatenation of d *independent* subgaussian distributions over $\mathbb{Z}$, each with parameter at most s. This is true for discrete Gaussian distributions over a power-of-two cyclotomic ring.

Theorem 3.13 (CPA Security)**.** *Fix a security parameter λ and let p, q, n, ℓ', χ be as defined in Construction 3.11. Take any $Q = \mathsf{poly}(\lambda)$ and suppose that p, q are coprime. Under the $\mathsf{MLWE}_{n,m,d,q,\chi}$ assumption with $m = n + Q$, Construction 3.11 is Q-query CPA secure.*

Theorem 3.14 (Circuit Privacy)**.** *Let λ be a security parameter and p, q, n, ℓ', χ be as defined in Construction 3.11. Suppose that χ is subgaussian with parameter s. If $n, \ell', s, d, \gamma_R = \mathsf{poly}(\lambda)$, and $B = 2^{\omega(\log \lambda)} \cdot kp^2$, then under the $\mathsf{MLWE}_{n,m,d,q,\chi}$ assumption with $m = n$, Construction 3.11 is circuit private with respect to the set $S = R_p^k$. Concretely, let C be a correctness parameter and let B_1, B_2 be bounds. Let $S = \{\mathbf{y} \in R_p^k : \|\mathbf{y}\|_1 \le B_1 \text{ and } \|\mathbf{y}\|_2 \le B_2\}$. Then under the $\mathsf{MLWE}_{n,m,d,q,\chi}$ assumption with $m = n$, for every efficient adversary $\mathcal{A}$ restricted to strategies in S, there exists an efficient simulator $\mathcal{S}$ where*

$$\Pr[\mathsf{ExptCircuitPriv}_{\Pi_{\mathsf{Enc}},\mathcal{A},\mathcal{S}}(1^\lambda) = 1] \le 1/2 + \varepsilon + \mathsf{negl}(\lambda),$$

and

$$\varepsilon = (4n+2)d\ell' \exp(-\pi C^2) + \frac{d\ell'(\gamma_R B_2 C s + \gamma_R B_1/2 + 2\gamma_R n C^2 s^2)}{B}. \tag{3.6}$$

Conjecture 3.15 (Linear-Only)**.** *Fix a security parameter λ and let p, d, τ be defined as in Construction 3.11. If $|R_p|^\tau = p^{\tau d} = \lambda^{\omega(1)}$, then Construction 3.11 is strictly linear-only (Definition 3.4).*

Remark 3.16 (Plausibility of Linear-Only Conjecture)**.** We make a few remarks on the plausibility of Conjecture 3.15.

- **Affine strategies:** The linear-only definition (Definition 3.4) rules out the possibility of the adversary implementing affine strategies. In particular, the adversary should not be able to obliviously sample a valid ciphertext (for a non-zero vector) nor should the adversary be able to craft a valid encryption of a (non-zero) vector that is not the result of applying a linear function to existing ciphertexts. The approach we take in Construction 3.11 is to sparsify the ciphertext space by defining valid ciphertexts to be those that encrypt vectors of the form $\mathbf{u}^\mathsf{T} = [\mathbf{v}^\mathsf{T} \mid (\mathbf{T}\mathbf{v})^\mathsf{T}]$, where $\mathbf{T}$ is a uniformly random matrix that is computationally hidden from the view of the adversary.

- **General homomorphic operations:** Regev-based encryption schemes are the basis of many somewhat and fully homomorphic encryption (FHE) schemes, which are certainly *not* linear-only. However, all existing constructions of FHE rely on making some algebraic modifications to either the message encoding, homomorphic evaluation, or decryption operations. It is not known that *vanilla* Regev encryption (like Construction 3.11) supports higher-degree homomorphisms. Evaluating whether Construction 3.11 supports more general homomorphic operations *without* modification is an intriguing open question and has a win-win flavor: either the linear-only conjecture holds and we can use it as the basis of zkSNARKs, or we discover new homomorphic capabilities on *standard* Regev encryption.

Remark 3.17 (Higher-Degree Extensions)**.** Construction 3.11 naturally extends to general cyclotomic rings $R = \mathbb{Z}[x]/\Phi_m(x)$, where $\Phi_m(x)$ is the m^{th} cyclotomic polynomial. The prime p can then be chosen so that R_p is isomorphic to one or more copies of $\mathbb{F}_{p^k}$ for some $k \ge 1$. This allows us to directly compile linear PCPs over a higher-degree extension field into preprocessing zkSNARKs.

Much like the case with fully homomorphic encryption based on RLWE [SV10, GHS12a, SV14], when R_p splits into ℓ copies of $\mathbb{F}_{p^k}$ (i.e., when the polynomial $\Phi_m(x)$ splits into ℓ irreducible factors $F_1(x), \ldots, F_\ell(x)$ of degree k over $\mathbb{F}_p$), it is possible to pack ℓ sets of queries (for *different* R1CS systems) into a single ciphertext (i.e., by associating each R1CS system with an irreducible factor $F_i(x)$). Likewise, the prover can now homomorphically compute the encrypted responses to all ℓ R1CS systems. Then, a *single* ciphertext contains the packed responses to ℓ different statements across ℓ different (and independent) R1CS systems.

When working with a module R of rank $d > 1$, homomorphic operations map to polynomial additions and multiplications over R_q. For efficiency reasons (discussed in Section 4.3), we set the modulus q to be a power of two. As such, it is more challenging to use fast polynomial multiplication algorithms (i.e., based on fast

Fourier transforms) to implement the homomorphic operations.[8] In this work, we focus on modules of rank 1 and 2 where the additional cost of the polynomial arithmetic is small. Extending to modules of higher rank and taking advantage of batching in a *concretely-efficient* manner is an interesting direction for future work.

Remark 3.18 (Reducing Coefficient Magnitudes)**.** In Theorems 3.12 and 3.14, the magnitude of q grows with the ℓ_1 and ℓ_2 norms of the vector of coefficients in the prover's linear combination. When $R = \mathbb{Z}$, we can reduce the magnitude of the individual coefficients in the linear combination over R_p from p to $p^{1/m}$ at the expense of increasing the length of the linear combination (i.e., the number of ciphertexts) by a factor of m. In particular, an encryption of a vector $\mathbf{v} \in R_p^\ell$ consists of encryptions of the vectors $\mathbf{v}, p^{1/m}\mathbf{v}, \ldots, p^{(m-1)/m}\mathbf{v} \in R_p^\ell$. To compute $\alpha\mathbf{v}$, the evaluator then computes $\sum_{i\in[m]} \alpha_{i-1} p^{(i-1)/m}\mathbf{v}$, where $\alpha_{m-1}\cdots\alpha_1\alpha_0$ are the "digits" of α expressed in base $p^{1/m}$. In this case, a linear combination with k coefficients from R_p translates to a linear combination of mk elements, each of magnitude $p^{1/m}$. Then, the ℓ_1 norm of the coefficients decreases from kp to $mkp^{1/m}$; similarly, the ℓ_2 norm decreases from $\sqrt{k}p$ to $\sqrt{mk}p^{1/m}$.

Modulus switching. The size of the ciphertext in Construction 3.11 is determined by three main parameters: the ring dimension d, the module dimension n, and the ciphertext modulus q. According to Theorem 3.12, the modulus q must be sufficiently large to support the required number of homomorphic operations. However, the modulus switching technique developed in the context of fully homomorphic encryption [BV11, BGV12, CNT12, AP14, DM15] provides a way to reduce the size of the ciphertexts *after* performing homomorphic operations. Specifically, modulus switching allows one to take a ciphertext over R_q and convert it to one over $R_{q'}$ where $q' < q$ while preserving the correctness of decryption. This technique applies to most Regev-based encryption schemes, including Construction 3.11. Reducing the size of the ciphertexts after homomorphic evaluation translates to a reduction in the proof size of the resulting zkSNARK. We begin by defining the ciphertext rescaling operation Scale from Brakerski et al. [BGV12]:

- $\mathsf{Scale}(\mathbf{x}, q, q', p) \to \mathbf{x}'$: On input integers $q > q' > p$ and a vector $\mathbf{x} \in R_q^n$, the scale operation outputs the vector $\mathbf{x}' \in R_{q'}^n$ that is closest to $(q'/q)\cdot\mathbf{x}$ such that $\mathbf{x}' = \mathbf{x} \pmod p$.

We now state the main theorem, adapted from [BGV12]. We provide the proof in Appendix B.1.4.

Theorem 3.19 (Modulus Switching [BGV12, adapted])**.** *Let λ be a security parameter and p, q, n, d, ℓ', χ be as defined in Construction 3.11. Let C be a correctness parameter. Suppose that χ is subgaussian with parameter s. Let $q' < q$ be a positive integer where $q' = q \pmod p$. Take any vector $\mathbf{a} \in R_q^n$, $\mathbf{c} \in R_q^{\ell'}$, and let $\mathbf{a}' \leftarrow \mathsf{Scale}(\mathbf{a}, q, q', p)$, $\mathbf{c}' \leftarrow \mathsf{Scale}(\mathbf{c}, q, q', p)$. Sample $\mathbf{S} \leftarrow \chi^{n\times\ell'}$. Let $\mathbf{z} = \mathbf{c} - \mathbf{S}^{\mathsf{T}}\mathbf{a} \in R_q^{\ell'}$ and suppose that*

$$\|\mathbf{z}\|_\infty < q/2 - (1 + n\gamma_R C s)\cdot(p/2)\cdot(q/q'). \tag{3.7}$$

Then, with probability $1 - 2dn\ell'\exp(-\pi C^2)$, $\mathbf{z} = \mathbf{z}' \pmod p$, where $\mathbf{z}' = \mathbf{c}' - \mathbf{S}^{\mathsf{T}}\mathbf{a}' \in R_{q'}^{\ell'}$.

In our construction, after performing homomorphic operations (via Add), the evaluator takes the final ciphertext $\mathsf{ct} = (\mathbf{a}, \mathbf{c})$, computes $\mathbf{a}' \leftarrow \mathsf{Scale}(\mathbf{a}, q, q', p)$ and $\mathbf{c}' \leftarrow \mathsf{Scale}(\mathbf{c}, q, q', p)$, and outputs the rescaled ciphertext $\mathsf{ct}' = (\mathbf{a}', \mathbf{c}')$. In particular, the components of ct' are elements of $R_{q'}$ rather than R_q. We additionally conjecture that Conjecture 3.15 holds even if we introduce this additional rescaling operation.

3.4 zkSNARKs from Linear-Only Encryption

In this section, we start by recalling the Bitansky et al. [BCI+13] compiler for constructing zkSNARKs from linear PCPs and linear-only vector encryption. We specifically describe the variant by Boneh et al. [BISW17] based on linear-only vector encryption.

[8] Systems for fully homomorphic encryption that take advantage of batching [GHS12a, HS14] work over a modulus q that splits into a product of many small primes $p_1, \ldots, p_t$; the primes p_i are moreover chosen so that $\mathbb{F}_{p_i}^*$ has sufficiently many roots of unity to invoke standard FFT algorithms for polynomial multiplication. These optimizations do not directly extend to the setting where q is a power of two.

Construction 3.20 (SNARKs from Linear-Only Vector Encryption [BCI+13, BISW17]). Let $\mathcal{CS} = \{\mathcal{CS}_\kappa\}_{\kappa \in \mathbb{N}}$ be a family of R1CS systems over a finite field $\mathbb{F}$. The construction relies on the following building blocks:

- Let $\Pi_{\mathsf{LPCP}} = (\mathcal{Q}_{\mathsf{LPCP}}, \mathcal{P}_{\mathsf{LPCP}}, \mathcal{V}_{\mathsf{LPCP}})$ be a k-query input-oblivious linear PCP for $\mathcal{CS}$. Let m be the query length of Π_{LPCP}.
- Let $\Pi_{\mathsf{Enc}} = (\mathsf{Setup}_{\mathsf{Enc}}, \mathsf{Encrypt}_{\mathsf{Enc}}, \mathsf{Decrypt}_{\mathsf{Enc}}, \mathsf{Add}_{\mathsf{Enc}})$ be a secret-key additively-homomorphic vector encryption scheme for $\mathbb{F}^k$.

The single-theorem, designated-verifier zkSNARK $\Pi_{\mathsf{SNARK}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ for $\mathcal{R}_{\mathcal{CS}}$ is defined as follows:

- $\mathsf{Setup}(1^\lambda, 1^\kappa) \to (\mathsf{crs}, \mathsf{st})$: On input the security parameter λ and the system index κ, run $(\mathsf{st}_{\mathsf{LPCP}}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}(1^\kappa)$ where $\mathbf{Q} \in \mathbb{F}^{m \times k}$. For each $i \in [m]$, let $\mathbf{q}_i^\mathsf{T} \in \mathbb{F}^k$ denote the i^{th} row of $\mathbf{Q}$. Then sample $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{Setup}_{\mathsf{Enc}}(1^\lambda, 1^k)$ and compute $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}_{\mathsf{Enc}}(\mathsf{sk}, \mathbf{q}_i^\mathsf{T})$ for each $i \in [m]$. Output the common reference string $\mathsf{crs} = (\kappa, \mathsf{pp}, \mathsf{ct}_1, \ldots, \mathsf{ct}_m)$ and the verification state $\mathsf{st} = (\mathsf{st}_{\mathsf{LPCP}}, \mathsf{sk})$.
- $\mathsf{Prove}(\mathsf{crs}, \mathbf{x}, \mathbf{w}) \to \pi$: On input the common reference string $\mathsf{crs} = (\kappa, \mathsf{pp}, \mathsf{ct}_1, \ldots, \mathsf{ct}_m)$, a statement $\mathbf{x}$, and a witness $\mathbf{w}$, the prover constructs an LPCP proof $\boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(1^\kappa, \mathbf{x}, \mathbf{w})$. The prover then homomorphically computes the linear PCP response $\mathsf{ct}^* \leftarrow \mathsf{Add}_{\mathsf{Enc}}(\mathsf{pp}, \{\mathsf{ct}_1, \ldots, \mathsf{ct}_m\}, \{\pi_1, \ldots, \pi_m\})$. It outputs the proof $\pi = \mathsf{ct}^*$.
- $\mathsf{Verify}(\mathsf{st}, \mathbf{x}, \pi)$: On input the verification state $\mathsf{st} = (\mathsf{st}_{\mathsf{LPCP}}, \mathsf{sk})$, the statement $\mathbf{x}$, and the proof $\pi = \mathsf{ct}^*$, the verifier first decrypts $\mathbf{a} \leftarrow \mathsf{Decrypt}_{\mathsf{Enc}}(\mathsf{sk}, \mathsf{ct}^*)$. If $\mathbf{a} = \perp$, the verifier outputs 0. Otherwise, it outputs $\mathcal{V}_{\mathsf{LPCP}}(\mathsf{st}_{\mathsf{LPCP}}, \mathbf{x}, \mathbf{a})$.

Completeness and knowledge soundness. Completeness of Construction 3.20 follows immediately from correctness of the underlying additively homomorphic encryption scheme and completeness of the linear PCP. Computational knowledge follows from the linear-only property together with knowledge soundness of the underlying linear PCP. The analysis follows closely from the corresponding analysis from Bitansky et al. [BCI+13] and Boneh et al. [BISW17], so we simply state the theorem here.

Theorem 3.21 (SNARKs from Linear-Only Vector Encryption [BCI+13, BISW17]). *If Π_{LPCP} is statistically sound against linear provers and Π_{Enc} is CPA-secure (for up to m messages) and strictly linear-only, then Π_{SNARK} from Construction 3.20 is a designated-verifier succinct argument of knowledge for $\mathcal{R}_{\mathcal{CS}}$ in the preprocessing model.*

The one difference in Theorem 3.21 and the corresponding statement from the previous works [BCI+13, BISW17] is that the previous works consider linear-only encryption schemes with support for affine strategies, and thus, require a linear PCP (or linear interactive proof) that provide knowledge against affine strategies. In this work, we make the stronger (but still plausible) conjecture that our *secret-key* vector encryption scheme is *strictly* linear-only (without support for affine strategies), which allows us to rely on a simpler information-theoretic primitive. We do note that it is straightforward to augment our linear PCP to provide knowledge against affine strategies with small overhead (see Remark A.3).

Remark 3.22 (Reusability). A limitation of Construction 3.20 is that it only provides one-time knowledge soundness in the designated-verifier model. A stronger notion is *reusable* knowledge soundness where knowledge holds even if the malicious prover has access to the verification oracle. Bitansky et al. [BCI+13] showed that combining a linear PCP satisfying reusable knowledge (also known as "strong knowledge") with an encryption scheme satisfying an *interactive* linear-only assumption yields a designated-verifier SNARK with reusable knowledge. Under this stronger security notion, the adversary has access to an oracle that can be used to check well-formedness of ciphertexts.[9] While it is straightforward to realize strong knowledge for the linear

[9] More precisely, the adversary has oracle access to the extractor; that is, on input a ciphertext, the oracle responds with either a linear combination that explains the adversary's query or $\perp$ if the ciphertext is invalid.

PCP, our vector encryption scheme (Construction 3.11) does not satisfy the stronger linear-only assumption. In particular, the ciphertext well-formedness oracle enables a key-recovery attack (similar to how a decryption oracle enables a CCA-1 key-recovery attack on Regev-based encryption schemes [LMSV11, CT14]). A similar issue arises in the previous lattice-based zkSNARK by Gennaro et al. [GMNO18]. Note though that single-theorem knowledge (as we consider here) does imply knowledge for *logarithmically-many* proofs by a standard hybrid argument.

In cases where the malicious prover does not learn the verifier's decision, the same CRS can be reused to check proofs of multiple statements. Even in the setting where the malicious prover has access to the verification oracle, the "verifier rejection" attack needed to break knowledge is always *detectable*; namely, to break knowledge, the malicious prover has to first submit *multiple* proofs that cause the verifier to reject (i.e., a super-constant number of invalid proofs). Thus, the zkSNARK still provides covert security [AL07] if the CRS is reused *and* the prover can observe the verifier's decisions. This is sufficient in many practical applications where there are out-of-band consequences when malicious behavior is detected. Another way to mitigate the impact of the verifier rejection attack is to have the verifier check *multiple* proofs and only reveal a single aggregate decision for all of the proofs in the batch. This reduces the leakage on the secret key from any single verification query.

Zero knowledge. Bitansky et al. [BCI+13] showed that combining a linear PCP satisfying HVZK with *re-randomizable* linear-only encryption yields a zkSNARK. An encryption scheme is re-randomizable if there is a public procedure that transforms *any* valid encryption of m into a *fresh* encryption of m. Our lattice-based vector encryption does not satisfy this property (due to the variability in the amount of noise in ciphertexts). Instead, we show that the *weaker* property of circuit privacy suffices to argue zero knowledge.

At a high-level, the argument goes as follows. First, by HVZK of the linear PCP, the linear PCP responses can be simulated given only the statement. Circuit privacy then says that the encrypted linear PCP responses can be simulated given only the simulated LPCP responses. We give the theorem below and defer the proof to Appendix B.2.

Theorem 3.23 (zkSNARKs via Circuit-Private Linear-Only Encryption)**.** *If Π_{LPCP} satisfies perfect honest-verifier zero knowledge and Π_{Enc} is CPA-secure (for up to m messages) and computationally (resp., statistically) circuit private, then Π_{SNARK} from Construction 3.20 is computationally (resp., statistically) zero knowledge. More precisely, if Π_{LPCP} is perfect* HVZK *and Π_{Enc} is ε-circuit private, then Π_{SNARK} from Construction 3.20 is 2ε-zero-knowledge.*

Remark 3.24 (Zero Knowledge without Circuit Privacy)**.** While circuit privacy plays a central role in the analysis of Theorem 3.23, augmenting our linear-only vector encryption scheme (Construction 3.11) with circuit privacy incurs a non-trivial concrete cost. As shown in Fig. 4, instantiating the encryption scheme in a setting without circuit privacy (corresponds to the setting where $\kappa = 0$), we can achieve a 30–40% reduction in prover time and a 45–50% reduction in proof size for the resulting zkSNARK (for verifying R1CS systems with 2^{20} constraints). A natural question to ask is whether we can still hope to argue zero knowledge for the resulting zkSNARK without relying on full circuit privacy. Consider a variant of Construction 3.11 where we modify Eq. (3.4) to instead output

$$\mathsf{ct}^* = (\mathbf{a}^*, \mathbf{c}^*) = \left(\sum_{i \in [k]} y_i \mathbf{a}_i,\ \sum_{i \in [k]} y_i \mathbf{c}_i\right). \tag{3.8}$$

Namely, we remove all of the re-randomization that the prover applies. Following the same notation and analysis as in the proof of Theorem 3.12 (see Appendix B.1),

$$\mathbf{c}^* = \mathbf{S}^\mathsf{T} \mathbf{a}^* + \sum_{i \in [k]} y_i \mathbf{u}_i + p \sum_{i \in [k]} y_i \mathbf{e}_i \in R_q^{\ell'},$$

In the standard circuit-privacy setting, the simulator is only given $\sum_{i \in [k]} y_i \mathbf{u}_i \bmod p$ and must be able to simulate the ciphertext ct^* given only this information. However, if the circuit privacy simulator is given *additional* information about the coefficients y_i (namely, the values of $\sum_{i\in[k]} y_i\mathbf{a}_i$, $\sum_{i\in[k]} y_i\mathbf{u}_i$, and $\sum_{i\in[k]} y_i\mathbf{e}_i$ over R_q), then ct^* can be simulated.

If we now consider a linear PCP where the honest-verifier zero-knowledge simulator can *additionally* simulate these additional components, then we can still apply the Bitansky et al. [BCI+13] compiler to obtain a zkSNARK. We define this property more precisely in Definition 3.25 and state the corresponding version of Theorem 3.23 in Lemma 3.26. It is interesting to analyze whether the QAP-based linear PCPs we consider in this work (or a simple adaptation thereof) satisfy HVZK in the presence of this leakage. A positive result would yield an improvement to the concrete efficiency of our zkSNARK. In our construction, the leakage on the linear PCP coefficients consists of several linear combinations (over R) of the components of the linear PCP proof. Importantly, these linear combinations are randomly sampled and *not* adversarially chosen. Moreover, the number of such linear combinations is significantly smaller than the length of the linear PCP.

Definition 3.25 (Honest-Verifier Zero Knowledge with Leakage)**.** Let $R = \mathbb{Z}[x]/f(x)$ be a polynomial ring where $\deg(f) = d$. Let p be a prime such that $R_p \cong \mathbb{F}_{p^d}$ is a finite field. Let $\Pi_{\mathsf{LPCP}} = (\mathcal{Q}_{\mathsf{LPCP}}, \mathcal{P}_{\mathsf{LPCP}}, \mathcal{V}_{\mathsf{LPCP}})$ be a linear PCP for a family of R1CS systems $\mathcal{CS} = \{\mathcal{CS}_\kappa\}_{\kappa\in\mathbb{N}}$ over R_p. Let $\mathcal{D}$ be a distribution on matrices over R and $q > p$ be a modulus. We say that Π_{LPCP} satisfies honest-verifier zero knowledge with $(\mathcal{D}, q)$-leakage if there exists an efficient simulator $\mathcal{S}_{\mathsf{LPCP}} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for all $\kappa \in \mathbb{N}$ and all instances $(\mathbf{x}, \mathbf{w})$ where $\mathcal{CS}_\kappa(\mathbf{x}, \mathbf{w}) = 1$,

$$\{(\mathsf{st}, \mathbf{Q}, [\mathbf{Q}^{\mathsf{T}}\boldsymbol{\pi}]_q, \mathbf{Z}, [\mathbf{Z}^{\mathsf{T}}\boldsymbol{\pi}]_q)\} \overset{s}{\approx} \{(\tilde{\mathsf{st}}, \tilde{\mathbf{Q}}, \tilde{\mathbf{a}}, \tilde{\mathbf{Z}}, \tilde{\mathbf{b}})\}, \tag{3.9}$$

where $(\mathsf{st}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}(1^\kappa)$, $\mathbf{Z} \leftarrow \mathcal{D}$, $\boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(1^\kappa, \mathbf{x}, \mathbf{w})$, $(\tilde{\mathsf{st}}, \tilde{\mathbf{Q}}, \tilde{\mathbf{Z}}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}_1(1^\kappa)$, and $(\tilde{\mathbf{a}}, \tilde{\mathbf{b}}) \leftarrow \mathcal{S}_2(\mathsf{st}_{\mathcal{S}}, \mathbf{x})$, and we write $[\mathbf{Q}^{\mathsf{T}}\boldsymbol{\pi}]_q$ and $[\mathbf{Z}^{\mathsf{T}}\boldsymbol{\pi}]_q$ to denote computations over the ring R_q (i.e., the elements of R_p are first lifted to R and the value of the matrix-vector product is then reduced modulo q). When the statistical distance between the two distributions in Eq. (3.9) is δ, we say that Π_{LPCP} is δ-HVZK with $(\mathcal{D}, q)$-leakage.

Lemma 3.26 (Zero Knowledge without Ciphertext Re-Randomization)**.** *Let $d = d(\lambda)$ and let $R = \mathbb{Z}[x]/(x^d + 1)$. Let p, q, n be lattice parameters, χ be an error distribution over R_q, and ℓ' be the plaintext dimension as defined in Construction 3.11. Let Π_{LPCP} be a linear PCP for a family of R1CS systems $\mathcal{CS} = \{\mathcal{CS}_\kappa\}_{\kappa\in\mathbb{N}}$ over R_p with query length t. Let $\mathcal{D}$ be the following distribution on matrices over R:*

- *Sample $\mathbf{A} \xleftarrow{\text{R}} R_q^{t\times n}$ and $\mathbf{E} \leftarrow \chi^{t\times\ell'}$.*
- *Output the matrix $\mathbf{Z} = [\mathbf{A} \mid \mathbf{E}] \in R^{t\times(n+\ell')}$.*

If Π_{LPCP} is δ-HVZK with $(\mathcal{D}, q)$-leakage, then the zkSNARK obtained by instantiating Construction 3.20 with Π_{LPCP} and the vector encryption scheme from Construction 3.11 without ciphertext re-randomization *(i.e., where we replace Eq. (3.4) with Eq. (3.8)) satisfies 2δ-statistical zero knowledge.*

4 Implementation and Evaluation

In this section, we provide an overview of our lattice-based zkSNARK implementation (by combining Claim 2.6 with Construction 3.11) and then describe our experimental evaluation.

4.1 Linear PCP Implementation

The prover's computation in the zkSNARK from Construction 3.20 consists of two main components: computing the linear PCP proof (Claim 2.6, Appendix A) and homomorphically computing the encrypted linear PCP responses. Computing the linear PCP responses (over a finite field $\mathbb{F}$) requires the prover to compute the coefficients of a polynomial $H(z) := (A(z)B(z) - C(z))/Z(z)$, where A, B, C are polynomials of degree $N_g - 1$ (over $\mathbb{F}$) determined by the R1CS system (which has N_g constraints), the statement, and the witness, and Z is a fixed polynomial. We refer to Appendix A for the full details of this construction.

Ben-Sasson et al. [BCG+13] described an efficient approach to compute the coefficients of H using fast Fourier transforms (FFTs) over $\mathbb{F}$. To use standard Cooley-Tukey FFTs for powers of two [CT65] (which we refer to as "radix-2 FFTs"), we require that $\mathbb{F}$ contains a multiplicative subgroup of order 2^d where $2^d > N_g$. Indeed, the construction of Ben-Sasson et al. uses a specially-chosen elliptic curve group whose order is divisible by a large power of 2. In our setting, we consider linear PCPs over a quadratic extension $\mathbb{F}_{p^2}$, whose order is $p^2 - 1 = (p+1)(p-1)$. In the best case, if $p = 2^d \pm 1$, then $\mathbb{F}_{p^2}$ has a subgroup of order 2^{d+1}. However, if $N_g > 2p$, the field $\mathbb{F}_{p^2}$ *never* has a sufficiently large subgroup to directly compute radix-2 FFTs.[10]

Our approach. When the field $\mathbb{F}$ contains a multiplicative subgroup whose order is a moderately large power of two (e.g., 2^d), we can still leverage (multiple) radix-2 FFTs to efficiently implement multipoint polynomial evaluation and interpolation over a domain $D \subset \mathbb{F}$ of size $|D| = k \cdot 2^d$ for a (small) $k > 1$. We give a brief overview of our approach here and defer the full details to Appendix C. Let $\omega \in \mathbb{F}$ be a primitive 2^d-th root of unity and let $H = H_1 = \langle \omega \rangle \subset \mathbb{F}$ be the subgroup of order 2^d generated by ω (corresponding to the 2^d-th roots of unity). We define our domain D (for multipoint evaluation and interpolation) to be $\bigcup_{i \in [k]} H_i$, where $H_2, \ldots, H_k$ are pairwise disjoint *cosets* of H. Polynomial evaluation over D can be implemented using k degree-2^d FFTs (over H_1), along with 2^d multipoint evaluations of polynomials of degree-k (over a fixed basis determined by the cosets). An analogous result holds for interpolation. As long as $k < 2^d$, the smaller evaluation/interpolations can be implemented in $k \log k$ time using standard FFTs. In this case, the running time of our algorithm for evaluating a polynomial on a domain of size $2^d k$ is $O(2^d k(d + \log k))$, which matches the asymptotic complexity of a standard FFT over a domain of the same size. We give more details in Appendix C. We use this approach to implement the linear PCP prover when working over fields with insufficient roots of unity to support a standard radix-2 FFT.

4.2 Lattice-Based zkSNARK Implementation

In this section, we describe our overall zkSNARK implementation. We begin by describing our methodology for setting the lattice parameters n, q, χ for our lattice-based vector encryption scheme (Construction 3.11). We then describe a few optimizations to improve the concrete efficiency of the resulting construction.

Lattice parameter selection. In the following description, let N_g denote the number of constraints in the R1CS system, p denote the plaintext modulus, and κ be a statistical security parameter for zero knowledge. We set $\kappa = 40$ for our primary experiments. We choose the parameters as follows:

- The plaintext modulus p is chosen so that $\mathbb{F}_{p^2}^*$ has a large power-of-two subgroup. In our specific instantiations, we choose $p = 2^{13} - 1$ (just large enough so the linear PCP from Construction 3.1 supports R1CS systems with over 2^{20} constraints without needing too many repetitions) and $p = 2^{19} - 1$ (a larger field so $\mathbb{F}_{p^2}^*$ contains 2^{20}-th roots of unity).
- The module rank d is chosen based on whether we are working with a linear PCP over $\mathbb{F}_{p^2}$ ($d = 2$) or if we are working over a linear PCP over $\mathbb{F}_p$ ($d = 1$). Since we work over $R = \mathbb{Z}[x]/(x^d + 1)$, $\gamma_R = 1$ if $d = 1$ and $\gamma_R = 2$ if $d = 2$.
- The plaintext dimension ℓ is the query length of the linear PCP for the R1CS system (Claim 2.6 and Remark 2.7). The query length of the linear PCP is chosen to achieve knowledge error at most 2^{-128}. The linear PCP from Claim 2.6 has 4 queries and when instantiated over $\mathbb{F}_{p^2}$, has knowledge error $2N_g/(p^2 - N_g)$. This is repeated ρ times to amplify knowledge (ρ is chosen such that $(2N_g/(p^2 - N_g))^\rho \le 2^{-128}$). The total number of queries is then $\ell = 4\rho$. In the case where we first apply Construction 3.1 to obtain a linear PCP over the base field $\mathbb{F}_p$, then $\ell = 8\rho$.
- The sparsification parameter τ is chosen based on Conjecture 3.15: namely, we choose τ to be the smallest value where $p^{\tau d} \le 2^{-128}$.

[10] While more general algorithms for FFT can be used for multipoint evaluation and interpolation over a domain whose size is a prime power [Rad68] or a product of coprime values [Goo58, Tho63], these algorithms are more complex to implement and worse in terms of concrete efficiency compared to basic radix-2 FFTs. We show how to implement our approach using a small number of radix-2 FFTs.

- The noise smudging bound B is chosen so that $\varepsilon = 2^{-\kappa}$ in Eq. (3.6) of Theorem 3.14 (which ensures roughly κ bits of zero knowledge). We set the constant $C = 6$, in which case $\exp(-\pi C^2) < 2^{-163}$. We use an upper bound for the ring dimension $n < 2^{12}$ (the ring dimension will be set (later) based on the security parameter and the modulus q, and in all of our cases, $n < 2^{12}$).[11]

  In Construction 3.20, the number of homomorphic operations the prover performs is equal to the length k of the linear PCP query. From Claim 2.6, we set $k = 2N_g$ when considering a linear PCP over $\mathbb{F}_{p^2}$ and $k = 4N_g$ if we first apply Construction 3.1 to obtain a linear PCP over $\mathbb{F}_p$; recall that we set the number of variables N_w to roughly coincide with the number of constraints N_g in our evaluation. The coefficients the prover uses are elements from R_p, so we use the bounds $B_1 = dkp$ and $B_2 = \sqrt{dk}p$. In Remark 3.18, we describe a trade-off where we replace each R_p coefficient with z coefficients, each of magnitude $p^{1/z}$ for any constant $z > 1$. In this case, $B_1 = dkzp^{1/z}$ and $B_2 = \sqrt{dkz}p^{1/z}$.

  The value of ε in Eq. (3.6) is essentially determined by $\gamma_R B_2 Cs + \gamma_R B_1/2 + 2\gamma_R nC^2 s^2$. This term is effectively dominated by the first two terms $\gamma_R B_2 Cs + \gamma_R B_1/2 = \gamma_R(\sqrt{k}pCs + pk/2)$. We take our noise distribution to be a discrete Gaussian distribution with noise rate s, and we choose s to balance the terms $\sqrt{k}pCs$ and $pk/2$. Since the security of LWE is determined by the modulus-to-noise ratio, using a larger noise rate reduces the lattice dimension; balancing these two terms allows us to use a higher noise rate without needing to increase the modulus size needed for correctness.

- We choose the modulus q to be the smallest power of two that satisfies Eq. (3.5). In all the cases we consider, $q \le 2^{128}$, so a power-of-two modulus allows us to implement all of the arithmetic using 128-bit integer arithmetic *without* needing to perform modular reductions after each arithmetic operation. As we elaborate below, compiler intrinsics on 64-bit architectures enable highly-optimized 128-bit arithmetic and is important for reducing the prover cost.

- As discussed in Section 3.3, we use modulus switching to reduce the ciphertext size *after* performing homomorphic operations (i.e., after the prover constructs the proof). We choose the reduced modulus q' to be the minimum value satisfying Theorem 3.19. Specifically, we bound $\|\mathbf{z}\|_\infty$ in Theorem 3.19 using Eqs. (B.2) and (B.3). In settings where there is very little slack between $\|\mathbf{z}\|_\infty$ and $q/2$ (as required by Theorem 3.19), we increase the modulus q by 1 bit. This enables a smaller q'. For R1CS systems with 2^{20} constraints, the reduced modulus q' is 2.5× to 2.7× smaller than the original modulus q (see Table 2). This translates to a corresponding reduction in the proof size of the resulting zkSNARK.

- Given the modulus q and noise rate s for the discrete Gaussian distribution, we use the LWE Estimator[12] by Albrecht et al. [APS15] to determine the smallest ring dimension n that provides 128-bits of security against the best-known *quantum* attacks. For our MLWE instantiation, we work under the assumption here that the best attack on MLWE over $(\mathbb{Z}_q[x]/(x^2+1))^n$ coincides with the best attack on LWE on a lattice of dimension $2n$.

We provide some example parameters we use in Table 2 (these include the parameter settings used for the "Shorter Proofs" and "Shorter CRS" instantiations for R1CS instances of size 2^{20} in Table 1). The classical hardness estimates are based on the estimated cost of the Chen-Nguyen algorithm [CN11] and the Becker et al. [BDGL16] algorithm. The quantum hardness estimate is based on the estimated cost of the quantum sieving algorithm by Laarhoven et al. [LMvdP15]. These estimates are all computed using the LWE Estimator tool [APS15].

Reducing the CRS size. Like most Regev-based encryption schemes, the ciphertexts in Construction 3.11 have the form $\mathsf{ct} = (\mathbf{a}, \mathbf{c})$ where $\mathbf{a} \in R_q^n$ is *uniformly random* and $\mathbf{c} \in R_q^{\ell'}$ encodes the message. For

[11] In one of our instantiations we use for comparison purposes (which does *not* use quadratic extensions), the ring dimension n is slightly larger than 2^{12}. For this setting, we use a larger bound on n when deriving parameters.

[12] https://lwe-estimator.readthedocs.io/en/latest/

Fields*	λ_q	λ_c	p	(n, d)	$\log q$	$\log q'$	s	ℓ	τ
$\mathbb{F}_p$, $\mathbb{F}_p$	128	138	$5 \cdot 2^{25} + 1$	$(4700, 1)$	123	49	80	82	5
$\mathbb{F}_{p^2}$, $\mathbb{F}_p$	128	138	$2^{19} - 1$	$(4050, 1)$	107	41	36	71	7
$\mathbb{F}_{p^2}$, $\mathbb{F}_{p^2}$	128	138	$2^{13} - 1$	$(1815, 2)$	98	35	64	109	5
	128	138	$2^{19} - 1$	$(2045, 2)$	108	41	40	36	4

[*] The first field listed is the base field for the linear PCP Π_{LPCP} and the second is the plaintext field for the linear-only vector encryption scheme Π_{Enc}.

Table 2: Lattice parameters for zkSNARK instantiations obtained by combining the linear PCP Π_{LPCP} from Claim 2.6 and Remark 2.7 with the linear-only vector encryption scheme Π_{Enc} from Construction 3.11. Here, λ_q is the estimated number of bits of quantum security, λ_c is the estimated number bits of classical security, p is the plaintext modulus, n is the ring dimension, d is the module rank, q is the ciphertext modulus, q' is the reduced modulus (after modulus switching), s is the width parameter for the discrete Gaussian noise distribution, ℓ is the dimension of the plaintext space, and τ is the sparsification parameter. Parameters shown are based on supporting an R1CS system with 2^{20} constraints. The final two rows correspond to the "Shorter Proofs" and the "Shorter CRS" instantiations in Table 1, respectively.

typical parameter settings, the ring dimension n is much larger than the (extended) plaintext dimension ℓ'. A *heuristic* approach to reduce the ciphertext size is to derive the random vector $\mathbf{a}$ as the output of a pseudorandom function (PRF) and include the PRF key in place of the vector $\mathbf{a}$ (or alternatively, take them to be the outputs of a public hash function). Security of these heuristics can be justified in the random oracle model [Gal13]. We adopt this approach in our implementation. Namely, instead of including the ciphertexts $\mathsf{ct}_1 = (\mathbf{a}_1, \mathbf{c}_1), \ldots, \mathsf{ct}_N = (\mathbf{a}_N, \mathbf{c}_N)$ in the CRS, the setup algorithm samples a PRF key k and sets $\mathbf{a}_i \leftarrow \mathsf{PRF}(k, i)$. The sequence of ciphertexts in the CRS is then $(k, \mathbf{c}_1, \ldots, \mathbf{c}_N)$. In our implementation, we use AES (in counter mode) as the underlying PRF. Similar approaches for reducing the size of the public components of lattice-based cryptosystems has been used for both lattice-based key-exchange [BCD+16] as well as previous lattice-based zkSNARKs [GMNO18].

As a reference point, in our implementation, $q \approx 2^{100}$ and each R_q^n element can be represented by roughly 50 KB (see Table 2 for some specific sets of parameters). The number of ciphertexts in the CRS is proportional to the size of the R1CS system. For a system of 2^{20} constraints, the CRS would contain around 2^{21} ciphertexts; in this case, the $\mathbf{a}$'s in the CRS would be roughly 100 GB in size. Deriving these components from a PRF (or random oracle) is necessary to reduce the CRS size.

Noise distribution. We take our noise distribution χ to be a discrete Gaussian distribution $\chi = \chi_s$ with mean 0 and width s (Eq. (2.1)). Note that in the case of the ring $R = \mathbb{Z}[x]/(x^2 + 1)$, the discrete Gaussian distribution decomposes into the product of two independent discrete Gaussian distributions over the integers. To efficiently sample from the discrete Gaussian distribution, we first truncate the distribution to the interval $[-6s, 6s] \cap \mathbb{Z}$; with probability $1 - 2^{-163}$, a sample from χ_s will fall into this interval. We then pre-compute a table of the cumulative density function for the truncated discrete Gaussian distribution $\tilde{\chi}_s$. We use inversion sampling to sample from $\tilde{\chi}_s$ given a uniformly-random 64-bit value. This is similar to the approach used in lattice-based key-exchange [BCD+16].

Big integer support. In our implementation, the ciphertext modulus q is around 100 bits. We implement all of the homomorphic operations (over the ring R_q) using 128-bit arithmetic. Since we choose q to be a power-of-two, we can compute over $\mathbb{Z}_{2^{128}}$ and defer the modular reduction to the end of the computation. Here, the modular reduction just corresponds to dropping the $(128 - \log q)$ most significant bits of the input.

We use the compiler intrinsic type `__uint128_t` for 128-bit arithmetic on a 64-bit architecture. Internally, each 128-bit value is represented by two 64-bit words. Multiplication of a 128-bit value and a 64-bit value (i.e.,

	Time (s)	**Rate (muls/s)**
Native (`uint64_t`)	2.1	$4.68 \cdot 10^9$
Compiler Intrinsic (`__uint128_t`)	6.8	$1.47 \cdot 10^9$
Boost Fixed Precision (128-Bit)	6.8	$1.47 \cdot 10^9$
Boost Fixed Precision (192-Bit)	53.6	$0.19 \cdot 10^9$
Boost Fixed Precision (256-Bit)	61.9	$0.16 \cdot 10^9$
GMP Multi-precision (mod 2^{128})	114.9	$0.087 \cdot 10^9$

Table 3: Time and effective rate to compute 10^{10} multiplications between an n-bit integer ($n = 64, 128, 192, 256$) and a 64-bit integer using different big-integer implementations on a 64-bit architecture. This models the primary cost in the prover's homomorphic evaluation.

scaling a ciphertext in R_q by a plaintext coefficient in R_p) requires just three x86-64 arithmetic operations (2 multiplications and 1 addition). Our microbenchmarks for performing multiplications (Table 3) indicate that using the compiler intrinsic representation for 128-bit arithmetic is over 16× faster than using a general-purpose multi-precision arithmetic library such as GMP for the same computation. Similarly, there is a large increase in concrete cost (around 8× to 9×) when going from 128-bit arithmetic to 192-bit or 256-bit *fixed precision* arithmetic (implemented in the Boost C++ libraries). Thus, being able to rely solely on 128-bit arithmetic to implement our scheme confers considerable advantages when working on a standard 64-bit architecture, and plays an important role for reducing the prover cost.

For instance, based on the cost breakdowns for CRS setup and prover complexity in Fig. 3 and taking into consideration these microbenchmarks for elementary arithmetic operations, using a modulus even slightly larger than 2^{128} would increase the prover computation time by a factor of 3× to 4×.[13] An even larger penalty would be incurred in CRS setup, where for the larger R1CS systems, the query encryption time (consisting of matrix-vector products over R_q) is over 99% of the overall setup time. Using larger integers would increase this by at least 8× to 9× based on our microbenchmarks.

4.3 Experimental Evaluation

We now describe our implementation and experimental evaluation of our lattice-based zkSNARK from Section 3.4.

System implementation. Our implementation is written in C++. We use `libsnark` [SCI21c] and `libfqfft` [SCI21a] to implement the linear PCP for R1CS satisfiability (Claim 2.6). In particular, we use the linear PCP implementation from `libsnark` (with the minor changes from Appendix A), and the implementation of the standard radix-2 FFT [CT65] (over a finite field) as well as the Bostan-Schost algorithms for multipoint evaluation and interpolation on points from a geometric sequence [BS05] from `libfqfft`. These building blocks suffice to implement our approach described in Section 4.1 for evaluating the linear PCP.

Metrics and evaluation methodology. Following previous works [BCR+19, COS20, SL20], we measure the performance of our system on R1CS systems with different number of constraints m (ranging from $m = 2^{10}$ to $m = 2^{20}$). Like previous works, we keep the number of variables n in each R1CS system to be roughly m (i.e., $n \approx m$), and we consider statements of a fixed length $k = 100$. The statement length only has a mild effect on the verification complexity (which is already very fast) and we do not focus on it in our evaluation.

[13] This penalty is only from the increased cost of arithmetic operations. The actual overhead will be even higher due to the need for larger lattice parameters to accommodate the larger modulus.

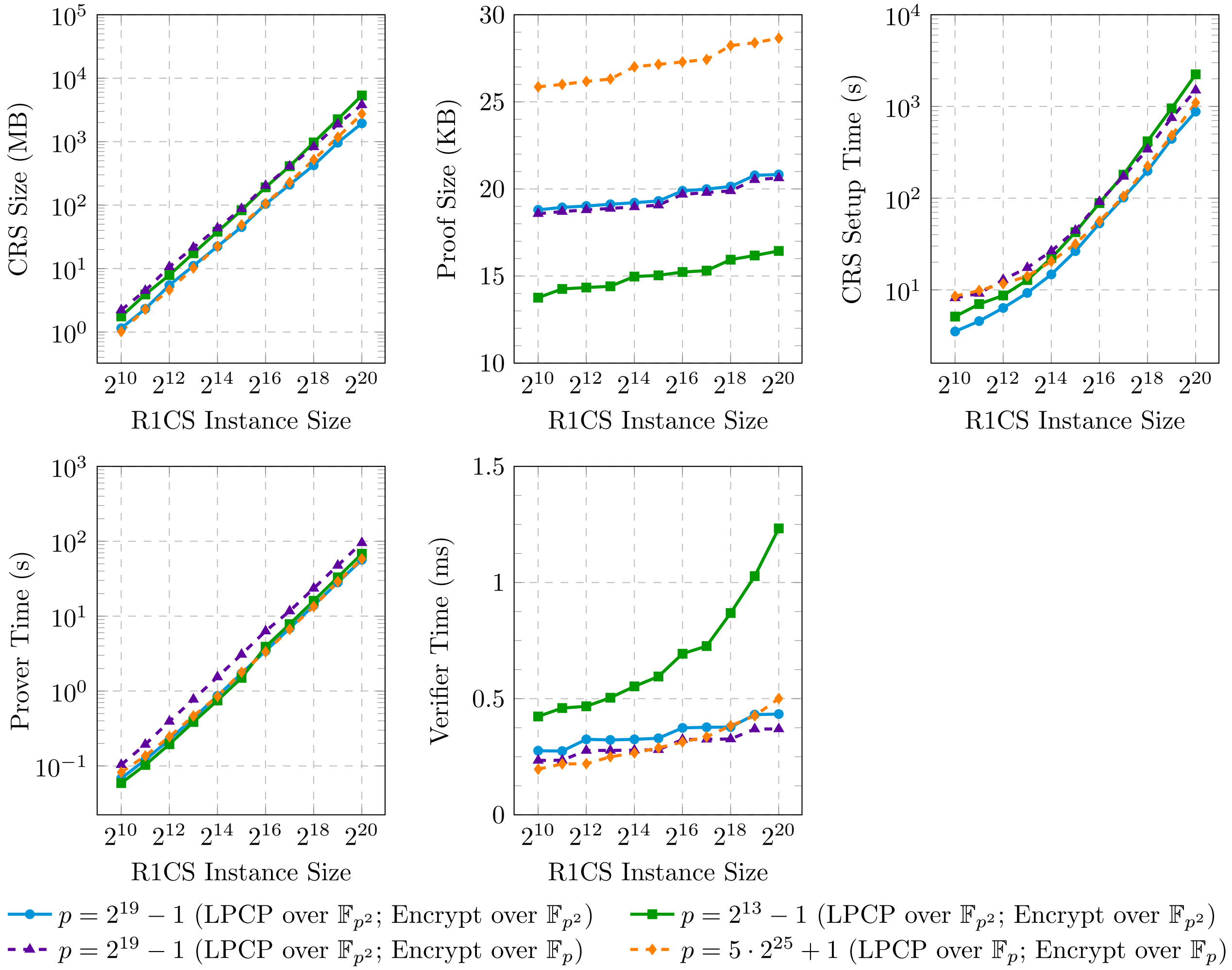

Figure 1: Performance comparison for different instantiations of our scheme for supporting R1CS instances of different sizes. The solid lines correspond to our primary instantiations using a linear PCP over $\mathbb{F}_{p^2}$ with a vector encryption scheme over $\mathbb{F}_{p^2}$. The dashed lines represent alternative instantiations using a vector encryption over the base field $\mathbb{F}_p$. In the case where the linear PCP is over the extension field and the vector encryption is over the base field, we first apply Construction 3.1 to obtain a linear PCP over the base field. We also consider a direct compilation from a linear PCP over $\mathbb{F}_p$ using a vector encryption scheme over $\mathbb{F}_p$.

We run all of our experiments on an Amazon EC2 `c5.4xlarge` instance running Ubuntu 20.04. The machine has 16 vCPUs (Intel Xeon Platinum 8124M at 3.0 GHz) and 32 GB of RAM. The processor supports the AES-NI instruction set. We compile our code using `gcc 9.3.0` for a 64-bit x86 architecture (which supports the `__uint128_t` compiler intrinsic for 128-bit integer arithmetic). All of our measurements are taken with a single-threaded execution.

General benchmarks. In Fig. 1, we compare the performance of different instantiations of our zkSNARK on R1CS instances of varying sizes. We consider two instantiations using linear PCPs and vector encryption over the extension field $\mathbb{F}_{p^2}$ (for $p = 2^{13} - 1$ and $p = 2^{19} - 1$), as well as two alternative instantiations where we use a vector encryption over the base field $\mathbb{F}_p$. For the latter instantiations, we consider both the

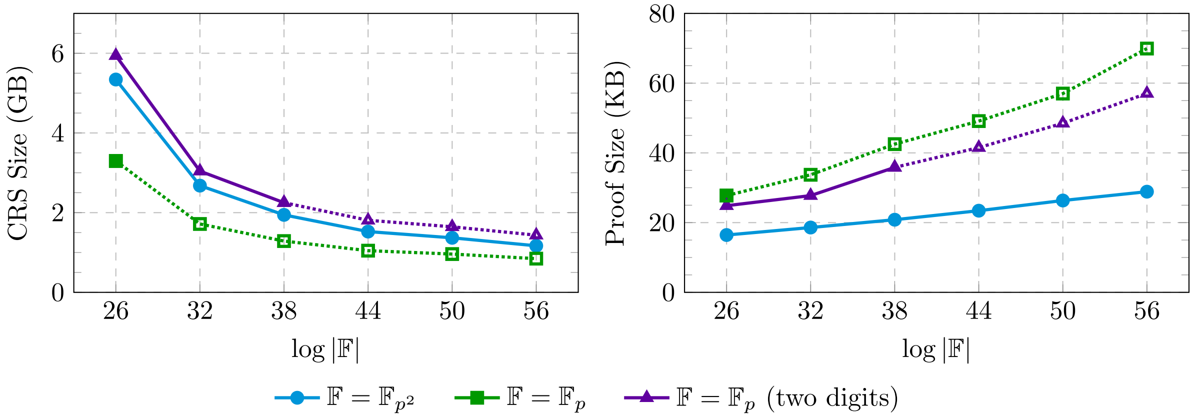

Figure 2: CRS size and proof size as a function of the field size $|\mathbb{F}|$, where $\mathbb{F}$ is either a quadratic extension $\mathbb{F}_{p^2}$ or a base field $\mathbb{F}_p$. The characteristic p is chosen so $\mathbb{F}$ has the prescribed size. Parameters based on instantiating Construction 3.20 over $\mathbb{F}$ for an R1CS system with 2^{20} constraints. For the $\mathbb{F} = \mathbb{F}_p$ setting, we also consider the case where each coefficient in the linear PCP is represented by two digits, each of size $\sqrt{p}$ (see Remark 3.18). Elements with a non-filled marker (and a dotted line) denote parameter settings where the modulus q exceeds 128 bits.

instantiation where we first compile a linear PCP over the extension field to a linear PCP over the base field (Construction 3.1) and a second instantiation where we directly construct a linear PCP over the base field. Across the board, the verifier time is small so we focus our discussion on the other metrics.

For our main instantiations (working over the extension field), the field size provides a trade-off in CRS size vs. proof size. Using a larger field decreases the CRS size (fewer repetitions needed for knowledge amplification at the linear PCP level), but leads to longer proofs (due to larger parameters). Concretely, for R1CS systems with 2^{20} constraints, increasing the characteristic from $p = 2^{13} - 1$ to $p = 2^{19} - 1$ decreases the CRS size by 2.8× (with a corresponding decrease in setup time), but increases the proof size by 1.2×. The prover complexity is essentially the same in the two cases.

In the case where we take a linear PCP over $\mathbb{F}_{p^2}$ and first apply Construction 3.1 to obtain a linear PCP over $\mathbb{F}_p$, the proof size still remains comparable to the case where we work exclusively over $\mathbb{F}_{p^2}$. However, the CRS size is doubled since Construction 3.1 increases the query length by the degree of the field extension, as is the prover complexity. The advantage of this construction is that it is based on *standard* lattices as opposed to *module* lattices, and thus, plausibly has better security.

Finally, if we consider the direct compilation of a linear PCP over the base field $\mathbb{F}_p$, the proof size is 1.4× to 1.8× longer than the constructions that use the extension field.

Extension field vs. base field. To quantify the concrete performance trade-off enabled by extension fields, we also compare our zkSNARKs over $\mathbb{F}_{p^2}$ with an instantiation over $\mathbb{F}_p$ (i.e., compile the linear PCP from Claim 2.6 over $\mathbb{F}_p$ using the linear-only vector encryption from Construction 3.11 over $\mathbb{F}_p$). The results are summarized in Fig. 2. We first note that most of the instantiations over $\mathbb{F}_p$ require working over a ring R_q with $q > 2^{128}$. As discussed in Section 4.2 (and Table 3), this will incur considerable computational overhead for the big-integer arithmetic. Working over the extension field allows us to consider instantiations over much larger fields without incurring the cost of big-integer arithmetic.

Fig. 2 shows that working over a quadratic extension field introduces a modest increase in the CRS size (by a factor of 1.4× to 1.6×) compared to working over a prime-order base field of similar size. In return, working over the extension field *reduces* the proof size by 1.7× to 2.4× (specifically, from nearly 70 KB to under 30 KB when considering a 56-bit field). As discussed in Section 1.2 (see also the formal analysis in Section 3.3), all of the lattice parameters grow with the field *characteristic*. Thus, for fields of comparable size, all of the lattice parameters will be larger if we work over a base field than if we work over an extension field (of smaller characteristic). This leads to longer proofs. The size of the CRS is smaller because of the CRS compression technique from Section 4.2. In particular, each lattice ciphertext in the CRS only consists of the *message-embedding* component. In this case, an element in R_q is represented by a *pair* of integers when $R = \mathbb{Z}[X]/(x^2+1)$ and by a *single* integer when $R = \mathbb{Z}$. Moreover, the dimension of the message space depends only on the field size and thus, is the same regardless of whether we work over a base field or an extension field.[14] As a result, when comparing instantiations over a base field vs. an extension field of similar size, the CRS in the extension field instantiation is longer.

If we consider the alternative instantiation over $\mathbb{F}_p$ from Remark 3.18 where the prover decomposes each $\mathbb{F}_p$ coefficient in the linear PCP proof into two separate coefficients, each of magnitude $\sqrt{p}$, then we can support a larger field size (i.e., up to 38 bits) without requiring a modulus q that exceeds 128 bits. The reduced parameter sizes translate to slightly shorter proofs (1.1×–1.2×) compared to the setting without the digit decomposition. However, this comes at the drawback of needing a longer CRS that is 1.7×–1.8× longer (since each component of the CRS is now decomposed into two components). Indeed, in this setting, the CRS size is comparable to the CRS size for the extension field instantiation; it is slightly worse due to the larger lattice parameters (some of which still scale based on the field characteristic). Despite the improvements in proof size obtained via the digit decomposition, the overall proof size is still 1.5×–2× longer than the proof size obtained from working over extension fields.

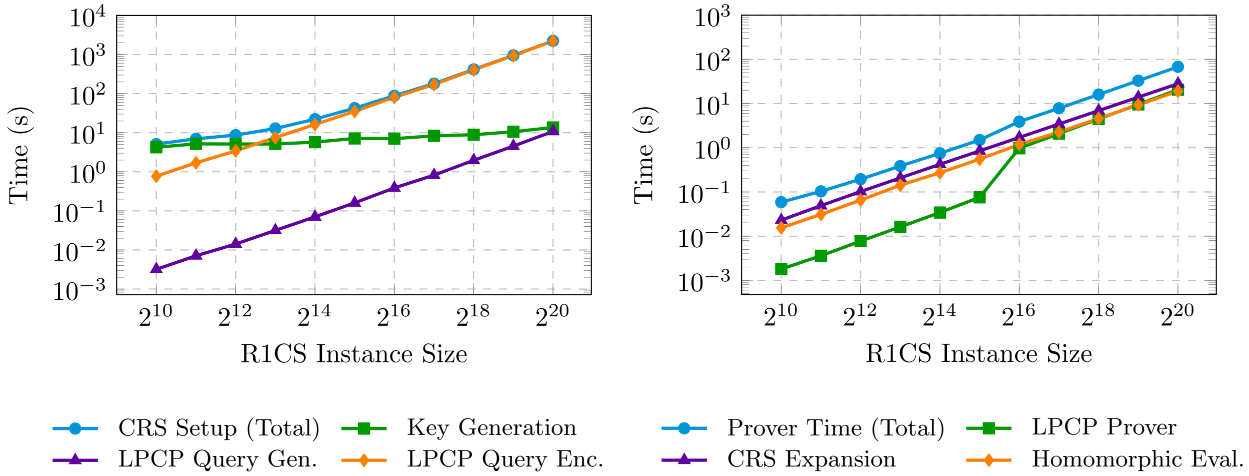

Figure 3: Cost breakdowns for CRS setup and prover for different R1CS instances. Measurements are based on instantiating Construction 3.20 with a linear PCP and a vector encryption scheme over $\mathbb{F}_{p^2}$ where $p = 2^{13} - 1$.

Microbenchmarks. For the setup and prover algorithms, we measure the concrete cost of each subcomponent. We show the breakdown for the construction over $\mathbb{F}_{p^2}$ where $p = 2^{13} - 1$ in Fig. 3 (the breakdown for other parameter settings are similar). For CRS generation, the cost is dominated by the time needed to encrypt the linear PCP queries. Namely, for an R1CS system with 2^{20} constraints, linear PCP query encryption constitutes 99% of the CRS generation time.

For the prover computation, we consider the cost of the linear PCP prover (Claim 2.6 and Appendix A), the time spent on CRS expansion (i.e., deriving the random ciphertext components $\mathbf{a} \in \mathbb{R}_q^n$ from the PRF

[14] The dimension (and size) of the *full* ciphertext is smaller when working over the extension field because the lattice parameters are smaller. If we only consider the message-embedding component of the ciphertext (which is a small fraction of the full ciphertext), then the size is smaller when working over a base field compared to working over the extension field.

p	Setting	Size		Time	
		CRS	Proof	Setup	Prover
$2^{13}-1$	Post-Quantum	5.3 GB	16.4 KB	2240 s	68 s
	Classical	5.3 GB	15.2 KB	2225 s	69 s
$2^{19}-1$	Post-Quantum	1.9 GB	20.8 KB	877 s	56 s
	Classical	1.9 GB	19.2 KB	865 s	56 s

Table 4: Performance comparison of zkSNARKs instantiated using parameters for 128-bits of classical vs. 128-bits of post-quantum security. For all measurements, we consider R1CS instances over $\mathbb{F}_{p^2}$ with 2^{20} constraints and compile them to zkSNARKs using linear-only vector encryption over $\mathbb{F}_{p^2}$.

key), and the cost of the homomorphic operations for computing the encrypted linear PCP response. The microbenchmarks show that about 40% of the time is spent on CRS expansion. For an R1CS instance of size 2^{20}, the expanded CRS is over 80 GB, and CRS expansion takes just under 30 s. Note that the vectors are generated on the fly and we do *not* need to store the full CRS in memory. For the larger instances, the remaining prover computation is evenly split between the homomorphic operations and computing the coefficients of the linear PCP; specifically, each of these components constitutes roughly 30% of the overall prover computation. In the case of the linear PCP prover, the computation is dominated by computing FFTs (see Appendix A). There is a jump in the cost of the FFTs when we switch to our modified FFT procedure (Section 4.1) for implementing the prover computation (for settings where $\mathbb{F}_{p^2}$ does not have enough primitive roots of unity to use standard power-of-two FFTs). By extrapolating the performance, our approach is about 7× slower than the basic radix-2 FFT.[15] When considering an R1CS system over $\mathbb{F}_{p^2}$ where $p = 2^{19} - 1$ (where there are sufficient roots of unity to invoke standard FFTs in the linear PCP prover algorithm), the linear PCP prover, homomorphic operations, and CRS expansion account for 6% (3.1 s), 38% (21.4 s), and 56% (31.8 s) of the total prover cost, respectively.

Zero knowledge. We also measure the concrete performance of our zkSNARKs for different choices of the zero-knowledge parameter κ. We provide the results in Fig. 4. In particular, when we work over $\mathbb{F}_{p^2}$ with $p = 2^{19} - 1$, and consider the setting *without* provable zero knowledge (i.e., setting $\kappa = 0$), the prover time (for an R1CS instance of size 2^{20}) is just 34 s. This represents an additional 1.6× speed-up over our construction with $\kappa = 40$ bits of zero knowledge. We see a 1.9× reduction in proof size (from 20.8 KB to 11.1 KB) in this setting. Working over a smaller base field, we can bring the proof size down to just 8 KB. This is around 20× shorter than previous post-quantum candidates (see Table 1). This reduction in proof size comes at the expense of a longer CRS (2.7 GB).

Classical vs. post-quantum security. If we instead instantiate our scheme to provide 128-bits of *classical* security (instead of post-quantum security), we obtain about a 5% reduction in proof size, setup time, and prover time. Realizing post-quantum security requires using a larger ring dimension n, but does not affect the modulus q. As such, the size of the CRS is unaffected (since we are deriving the random component of each ciphertext from a PRF). We provide more details in Table 4.

Comparison with other schemes. Finally, we compare the performance of our scheme with the most succinct pairing-based zkSNARK of Groth [Gro16] as well as several recent post-quantum zkSNARKs:

[15] When $p = 2^{13} - 1$, the field $\mathbb{F}_{p^2}$ contains a 2^{14}-th root of unity, so we can use standard radix-2 FFTs for R1CS instances with up to 2^{14} constraints. For instances of size 2^{15}, we use the approach from Section 4.1 and Appendix C, but directly inline the multipoint evaluation and interpolation on two points (this coincides with an existing implementation from `libfqfft` [SCI21a]). For instances larger than 2^{15}, we use the general Bostan-Schost algorithms [BS05] for the multipoint evaluation and interpolation. This introduces the 7× overhead in the cost of the FFT.

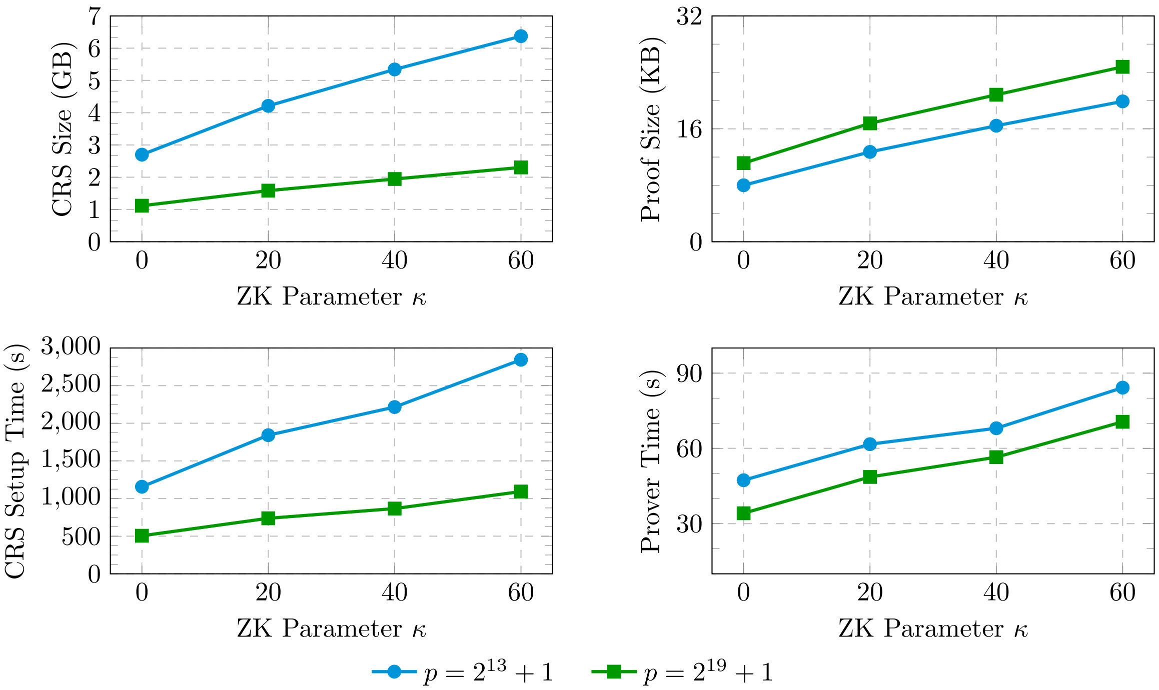

Figure 4: Cost breakdowns as a function of the zero-knowledge parameter κ (i.e., the zero-knowledge distinguishing advantage of any $\mathsf{poly}(\lambda)$ adversary is bounded by $2^{-\kappa} + \mathsf{negl}(\lambda)$). All measurements taken for an R1CS instance over $\mathbb{F}_{p^2}$ with 2^{20} constraints (and compiled using vector encryption over $\mathbb{F}_{p^2}$).

Ligero [AHIV17], Aurora [BCR$^+$19], Fractal [COS20], and Gennaro et al. [GMNO18]. With the exception of the lattice-based scheme of Gennaro et al. [GMNO18], we measure the performance on each scheme on the same system. We use `libsnark` [SCI21c] for the implementation of Groth's pairing-based construction [Gro16] and `libiop` [SCI21b] for the implementations of Ligero [AHIV17], Aurora [BCR$^+$19], and Fractal [COS20]. For each scheme, we consider the default implementation provided by the library. We note that these schemes export different base fields for the R1CS which makes a direct comparison challenging. In our benchmarks, we measure the performance of each scheme over their preferred field for an R1CS system with a fixed number of constraints. We give our results in Table 1 and refer to Section 1 for further discussion.

5 Related Work

There has been a flurry of recent works studying the asymptotic and concrete efficiency of succinct arguments. We survey several families of constructions here and also include a comparison with several representative schemes in Table 5. In the following, we use N to denote the size of the NP relation being verified.

Linear PCPs and QAP-based constructions. Gennaro et al. [GGPR13] and Bitansky et al. [BCI$^+$13] described general frameworks for constructing *constant-size* zkSNARKs from linear PCPs (specifically, from quadratic arithmetic programs (QAPs)). Several works have extended these frameworks [DFGK14, Gro16, BISW17, GMNO18, BISW18, BIOW20]. These constructions are the basis of numerous systems and implementations [PHGR13, BCG$^+$13, BFR$^+$13, BCTV14b, BCTV14a, FGP14, WSR$^+$15, BBFR15, CTV15, DFKP16, FFG$^+$16, BCG$^+$14]. These constructions offer the best *succinctness*, but this comes at the expense of needing an expensive, trusted, and language-dependent setup, as well as a quasilinear-time prover.

Interactive oracle proofs. Following the seminal works of Kilian [Kil92] and Micali [Mic00], a recent line of works [BCGV16, BBC+17, BCF+17, BBHR18b, BBHR18a, BCR+19, CHM+20, COS20, BCG20, BCL20, LSTW21] have shown how to construct zkSNARKs from short PCPs [BS08], and their generalization, interactive oracle proofs (IOPs) [BCS16, RRR16]. These constructions rely on the Fiat-Shamir heuristic [FS86] to obtain a non-interactive argument in the random oracle model. Many IOP constructions have a *transparent* (i.e., non-trusted) setup, and moreover, are plausibly post-quantum. Proof sizes for IOP-based constructions typically range in the hundreds of kilobytes.

Bünz et al. [BFS20] introduced *polynomial IOPs*, a generalization of linear PCPs to the IOP setting, where on each round, the verifier has oracle access to a bounded-degree polynomial. Polynomial IOPs can be compiled into succinct arguments [MBKM19, GWC19, CHM+20, SL20] via polynomial commitments. These schemes have excellent concrete succinctness (a few hundred bytes to a few kilobytes), a universal or transparent setup, but generally rely on pre-quantum assumptions.

MPC-in-the-head. Ishai et al. [IKOS07] introduced the "MPC-in-the-head" paradigm for building zero-knowledge proofs from general multiparty computation. The Ligero system [AHIV17] was the first argument with $\sqrt{N}$ size proofs in this framework. Bhadauria et al. [BFH+20] combined Ligero with IOPs to reduce the proof size to $\mathsf{polylog}(N)$. Both constructions support sublinear verification for *structured* circuits, but verification is linear for general circuits.

GKR-constructions. Another line of work starts from the succinct interactive argument for verifying arithmetic circuits by Goldwasser, Kalai, and Rothblum (GKR) [GKR08] A sequence of works [CMT12, Tha13, WTS+18, ZGK+17, Set20, XZZ+19, ZXZS20] have built on GKR to obtain efficient *non-interactive* arguments for (layered) circuits (and often tailoring to special structures for better concrete efficiency). In these constructions, the size of the proof (and the verifier complexity) typically scale with the depth of the circuit. An appealing feature of these constructions is their low prover complexities: namely, the cost of the prover scale *linearly* in the size of the NP relation (over large fields). Zhang et al. [ZWZZ20] recently showed how to leverage GKR to verify *general* arithmetic circuits while retaining a linear-time prover and sublinear verification (for structured circuits). The proof size in their construction scales with the depth of the circuit.

Inner product arguments. Building on works by Bayer and Groth [Gro09, BG12], Bootle et al. [BCC+16] introduced zero-knowledge arguments for arithmetic circuit satisfiability based on *inner product arguments*. Bünz et al. [BBB+18] improved the construction to achieve shorter proofs and verification times. While the proofs are short, the verification time scales linearly with the circuit size and these constructions rely on pre-quantum assumptions.

Lattice-based constructions. In the lattice-based setting, there have been several instantiations in the designated-verifier model based on linear PCPs [GMNO18, BISW17, BISW18]. Baum et al. [BBC+18] gave the first publicly-verifiable argument from standard lattice assumptions with $\tilde{O}(\sqrt{N})$-size proofs. Bootle et al. [BLNS20] reduced the proof size further to $\mathsf{polylog}(N)$. In both these cases, the verifier is not succinct and runs in *linear* time.

Acknowledgments

We thank Brennan Shacklett and Samir Menon for their help with an early prototype implementation of lattice-based zkSNARKs. Y. Ishai supported by ERC Project NTSC (742754), BSF grant 2018393, and ISF grant 2774/20. D. J. Wu is supported by NSF CNS-1917414, NSF CNS-2045180, and a Microsoft Research Faculty Fellowship.

	PQ	TP	PV	Proof Size		Runtime		Crypto.
				Asymptotic	Concrete	Prover	Verifier	Structure
Groth [Gro16]	○	○	●	1	128 B	$N \log N$	$\lvert x \rvert$	Pairings
Marlin [CHM+20]	○	●	●	1	704 B	$N \log N$	$\lvert x \rvert + \log N$	Pairings
Sonic [MBKM19]	○	◐	●	1	1.1 KB	$N \log N$	$\lvert x \rvert + \log N$	Pairings
Xiphos [SL20]	○	●	●	$\log N$	61 KB	N	$\lvert x \rvert + \log N$	Pairings
Spartan [Set20]	○	●	●	$\sqrt{N}$	142 KB	N	$\lvert x \rvert + \sqrt{N}$	Groups
Fractal [COS20]	●	●	●	$\log^2 N$	215 KB†	$N \log N$	$\lvert x \rvert + \log^2 N$	RO
[GMNO18]*	●	○	○	1	640 KB‡	$N \log N$	$\lvert x \rvert$	Lattices
STARK [BBHR18b]	●	●	●	$\log^2 N$	3.2 MB	$N \mathsf{polylog}(N)$	$\lvert x \rvert + \log^2 N$	RO
This work*	●	○	○	1	16 KB	$N \log N$	$\lvert x \rvert$	Lattices

*For the *asymptotic* estimates for the lattice-based constructions, we consider an instantiation over a field of size $2^{\Omega(\lambda)}$ (i.e., similar to the field sizes in the group-based and pairing-based constructions). For concrete efficiency, it is advantageous to work over smaller fields. When instantiated over a field $\mathbb{F}$, the lattice-based parameters scale with $\mathsf{polylog}(|\mathbb{F}|)$.

†Proof sizes for Fractal measured using the implementation from `libiop` [SCI21b] with the default configuration over a 181-bit prime field. The largest R1CS instance we could measure has 2^{19} constraints, so this is the proof size we report here.

‡This number is for a circuit with 2^{16} gates since the paper does not provide measurements for larger circuit sizes.

Table 5: Comparison with recent zkSNARKs for verifying an NP relations of size N and statements of length $|x|$. For brevity, we focus on schemes that have *sublinear* proof size and *sublinear* verification for general NP relations. Asymptotic running times and parameter sizes are given up to multiplicative $\mathsf{poly}(\lambda)$ factors (where λ is the security parameter). For the "Concrete Proof Size" column, we report the approximate size of a proof for verifying an NP relation of size $N \approx 2^{20}$ at the 128-bit security level (as reported in the respective works unless noted otherwise). The "PQ" column specifies whether the construction is post-quantum secure (●) or only classically secure (○). The "TP" column denotes whether the scheme is transparent (●), relies on a trusted setup for a *universal* CRS (◐), or relies on a *trusted* sampling of a *language-dependent* CRS (○). The "PV" column specifies whether the argument is publicly-verifiable (●) or designated-verifier (○). The "Crypto. Structure" column describes the primary (algebraic) structure underlying the construction. We distinguish between pairing groups and pairing-free groups by using "Groups" to denote the latter. We write "RO" to denote a construction based on random oracles.

References

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, pages 595–618, 2009.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *ACM CCS*, pages 2087–2104, 2017.

[AJLA+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. In *EUROCRYPT*, pages 483–501, 2012.

[AL07] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In *TCC*, pages 137–156, 2007.

[AP13] Jacob Alperin-Sheriff and Chris Peikert. Practical bootstrapping in quasilinear time. In *CRYPTO*, pages 1–20, 2013.

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *CRYPTO*, pages 297–314, 2014.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *J. Math. Cryptol.*, 9(3):169–203, 2015.

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

[BBC$^+$17] Eli Ben-Sasson, Iddo Bentov, Alessandro Chiesa, Ariel Gabizon, Daniel Genkin, Matan Hamilis, Evgenya Pergament, Michael Riabzev, Mark Silberstein, Eran Tromer, and Madars Virza. Computational integrity with a public random string from quasi-linear PCPs. In *EUROCRYPT*, pages 551–579, 2017.

[BBC$^+$18] Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël del Pino, Jens Groth, and Vadim Lyubashevsky. Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits. In *CRYPTO*, pages 669–699, 2018.

[BBFR15] Michael Backes, Manuel Barbosa, Dario Fiore, and Raphael M. Reischuk. ADSNARK: nearly practical and privacy-preserving proofs on authenticated data. In *IEEE Symposium on Security and Privacy*, pages 271–286, 2015.

[BBHR18a] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast Reed-Solomon interactive oracle proofs of proximity. In *ICALP*, pages 14:1–14:17, 2018.

[BBHR18b] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *IACR Cryptol. ePrint Arch.*, 2018:46, 2018.

[BCC$^+$16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *EUROCRYPT*, pages 327–357, 2016.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In *STOC*, pages 111–120, 2013.

[BCD$^+$16] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE. In *ACM CCS*, pages 1006–1018, 2016.

[BCF$^+$17] Eli Ben-Sasson, Alessandro Chiesa, Michael A. Forbes, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. Zero knowledge protocols from succinct constraint detection. In *TCC*, pages 172–206, 2017.

[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 *CRYPTO*, pages 90–108, 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 *IEEE Symposium on Security and Privacy*, pages 459–474, 2014.

[BCG20] Jonathan Bootle, Alessandro Chiesa, and Jens Groth. Linear-time arguments with sublinear verification from tensor codes. In *TCC*, pages 19–46, 2020.

[BCGT13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. Fast reductions from RAMs to delegatable succinct constraint satisfaction problems: extended abstract. In *ITCS*, pages 401–414, 2013.

[BCGV16] Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, and Madars Virza. Quasi-linear size zero knowledge from linear-algebraic pcps. In *TCC*, pages 33–64, 2016.

[BCI$^+$13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *TCC*, pages 315–333, 2013.

[BCL20] Jonathan Bootle, Alessandro Chiesa, and Siqi Liu. Zero-knowledge succinct arguments with a linear-time prover. *IACR Cryptol. ePrint Arch.*, 2020:1527, 2020.

[BCPR14] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In *STOC*, pages 505–514, 2014.

[BCR$^+$19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In *EUROCRYPT*, pages 103–128, 2019.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *TCC*, pages 31–60, 2016.

[BCTV14a] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. In *CRYPTO*, pages 276–294, 2014.

[BCTV14b] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von Neumann architecture. In *USENIX Security Symposium*, pages 781–796, 2014.

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *SODA*, pages 10–24, 2016.

[BFH$^+$20] Rishabh Bhadauria, Zhiyong Fang, Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, Tiancheng Xie, and Yupeng Zhang. Ligero++: A new optimized sublinear IOP. In *ACM CCS*, pages 2025—2038, 2020.

[BFR$^+$13] Benjamin Braun, Ariel J. Feldman, Zuocheng Ren, Srinath T. V. Setty, Andrew J. Blumberg, and Michael Walfish. Verifying computations with state. In *SOSP*, pages 341–357, 2013.

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent snarks from DARK compilers. In *EUROCRYPT*, pages 677–706, 2020.

[BG12] Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In *EUROCRYPT*, pages 263–280, 2012.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325, 2012.

[BHH$^+$15] Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. SPHINCS: practical stateless hash-based signatures. In *EUROCRYPT*, pages 368–397, 2015.

[BIOW20] Ohad Barta, Yuval Ishai, Rafail Ostrovsky, and David J. Wu. On succinct arguments and witness encryption from groups. In *CRYPTO*, pages 776–806, 2020.

[BISW17] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Lattice-based SNARGs and their application to more efficient obfuscation. In *EUROCRYPT*, pages 247–277, 2017.

[BISW18] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Quasi-optimal SNARGs via linear multi-prover interactive proofs. In *EUROCRYPT*, pages 222–255, 2018.

[BLNS20] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. A non-PCP approach to succinct quantum-safe zero-knowledge. In *CRYPTO*, pages 441–469, 2020.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In *ASIACRYPT*, pages 514–532, 2001.

[BS05] Alin Bostan and Éric Schost. Polynomial evaluation and interpolation on special sets of points. *J. Complex.*, 21(4):420–446, 2005.

[BS08] Eli Ben-Sasson and Madhu Sudan. Short PCPs with polylog query complexity. *SIAM J. Comput.*, 38(2):551–607, 2008.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011.

[CDG$^+$17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In *ACM CCS*, pages 1825–1842, 2017.

[CHM$^+$20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In *EUROCRYPT*, pages 738–768, 2020.

[CMT12] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *ITCS*, pages 90–112, 2012.

[CN11] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In *ASIACRYPT*, pages 1–20, 2011.

[CNT12] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 446–464, 2012.

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In *EUROCRYPT*, pages 769–793, 2020.

[CT65] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[CT14] Massimo Chenal and Qiang Tang. On key recovery attacks against existing somewhat homomorphic encryption schemes. In *LATINCRYPT*, pages 239–258, 2014.

[CTV15] Alessandro Chiesa, Eran Tromer, and Madars Virza. Cluster computing in zero knowledge. In *EUROCRYPT*, pages 371–403, 2015.

[CY21] Alessandro Chiesa and Eylon Yogev. Subquadratic SNARGs in the random oracle model. 2021.

[DFGK14] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In *ASIACRYPT*, pages 532–550, 2014.

[DFKP16] Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, and Bryan Parno. Cinderella: Turning shabby X.509 certificates into elegant anonymous credentials with the magic of verifiable computation. In *IEEE Symposium on Security and Privacy*, pages 235–254, 2016.

[DKL$^+$18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. In *EUROCRYPT*, pages 617–640, 2015.

[FFG$^+$16] Dario Fiore, Cédric Fournet, Esha Ghosh, Markulf Kohlweiss, Olga Ohrimenko, and Bryan Parno. Hash first, argue later: Adaptive verifiable computations on outsourced data. In *ACM CCS*, pages 1304–1316, 2016.

[FGP14] Dario Fiore, Rosario Gennaro, and Valerio Pastro. Efficiently verifiable computation on encrypted data. In *ACM CCS*, pages 844–855, 2014.

[FHK$^+$20] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-fourier lattice-based compact signatures over NTRU (specification v1.2). 2020.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.

[Gal13] Steven D Galbraith. Space-efficient variants of cryptosystems based on learning with errors. 2013.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *EUROCRYPT*, pages 626–645, 2013.

[GHS12a] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In *EUROCRYPT*, pages 465–482, 2012.

[GHS12b] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In *CRYPTO*, pages 850–867, 2012.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, pages 113–122, 2008.

[GMNO18] Rosario Gennaro, Michele Minelli, Anca Nitulescu, and Michele Orrù. Lattice-based zk-SNARKs from square span programs. In *ACM CCS*, pages 556–573, 2018.

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

[GNS21] Chaya Ganesh, Anca Nitulescu, and Eduardo Soria-Vazquez. Rinocchio: SNARKs for ring arithmetic. *IACR Cryptol. ePrint Arch.*, 2021:322, 2021.

[Goo58] Irving John Good. The interaction algorithm and practical fourier analysis. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):361–372, 1958.

[Gro09] Jens Groth. Linear algebra with sub-linear zero-knowledge arguments. In *CRYPTO*, pages 192–208, 2009.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT*, pages 321–340, 2010.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT*, pages 305–326, 2016.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, pages 99–108, 2011.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *IACR Cryptol. ePrint Arch.*, 2019:953, 2019.

[HS14] Shai Halevi and Victor Shoup. Algorithms in helib. In *CRYPTO*, pages 554–571, 2014.

[IKO07] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short pcps. In *CCC*, pages 278–291, 2007.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *STOC*, pages 21–30, 2007.

[ISW21] Yuval Ishai, Hang Su, and David J. Wu. Shorter and faster post-quantum designated-verifier zkSNARKs from lattices. In *ACM CCS*, 2021.

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

[LMSV11] Jake Loftus, Alexander May, Nigel P. Smart, and Frederik Vercauteren. On CCA-secure somewhat homomorphic encryption. In *Selected Areas in Cryptography*, pages 55–72, 2011.

[LMvdP15] Thijs Laarhoven, Michele Mosca, and Joop van de Pol. Finding shortest lattice vectors faster using quantum search. *Des. Codes Cryptogr.*, 77(2-3):375–400, 2015.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, pages 1–23, 2010.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptogr.*, 75(3):565–599, 2015.

[LSTW21] Jonathan Lee, Srinath Setty, Justin Thaler, and Riad Wahby. Linear-time zero-knowledge SNARKs for R1CS. *IACR Cryptol. ePrint Arch.*, 2021:30, 2021.

[MBKM19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updateable structured reference strings. *IACR Cryptol. ePrint Arch.*, 2019:99, 2019.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In *EUROCRYPT*, pages 735–763, 2016.

[Nit19] Anca Nitulescu. Lattice-based zero-knowledge snargs for arithmetic circuits. In *LATINCRYPT*, pages 217–236, 2019.

[Pei16] Chris Peikert. A decade of lattice cryptography. *Found. Trends Theor. Comput. Sci.*, 10(4):283–424, 2016.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *IEEE Symposium on Security and Privacy*, pages 238–252, 2013.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, pages 554–571, 2008.

[Rad68] Charles M Rader. Discrete fourier transforms when the number of data samples is prime. *Proceedings of the IEEE*, 56(6):1107–1108, 1968.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93, 2005.

[RRR16] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In *STOC*, pages 49–62, 2016.

[SBV$^+$13] Srinath T. V. Setty, Benjamin Braun, Victor Vu, Andrew J. Blumberg, Bryan Parno, and Michael Walfish. Resolving the conflict between generality and plausibility in verified computation. In *EuroSys*, pages 71–84, 2013.

[Sch80] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4), 1980.

[SCI21a] SCIPR Lab. `libfqfft`: C++ library for FFTs in finite fields. https://github.com/scipr-lab/libfqfft/, 2021.

[SCI21b] SCIPR Lab. `libiop`: a C++ library for IOP-based zkSNARKs. https://github.com/scipr-lab/libiop, 2021.

[SCI21c] SCIPR Lab. `libsnark`: a C++ library for zkSNARK proofs. https://github.com/scipr-lab/libsnark/, 2021.

[Set20] Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In *CRYPTO*, pages 704–737, 2020.

[SL20] Srinath T. V. Setty and Jonathan Lee. Quarks: Quadruple-efficient transparent zkSNARKs. *IACR Cryptol. ePrint Arch.*, 2020:1275, 2020.

[SV10] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *PKC*, pages 420–443, 2010.

[SV14] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *Des. Codes Cryptogr.*, 71(1):57–81, 2014.

[SVP$^+$12] Srinath T. V. Setty, Victor Vu, Nikhil Panpalia, Benjamin Braun, Andrew J. Blumberg, and Michael Walfish. Taking proof-based verified computation a few steps closer to practicality. In *USENIX*, pages 253–268, 2012.

[Tha13] Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In *CRYPTO*, pages 71–89, 2013.

[Tho63] Llewellyn H Thomas. Using a computer to solve problems in physics. *Applications of digital computers*, pages 44–45, 1963.

[WB15] Michael Walfish and Andrew J. Blumberg. Verifying computations without reexecuting them. *Commun. ACM*, 58(2):74–84, 2015.

[WSR$^+$15] Riad S. Wahby, Srinath T. V. Setty, Zuocheng Ren, Andrew J. Blumberg, and Michael Walfish. Efficient RAM and control flow in verifiable outsourced computation. In *NDSS*, 2015.

[WTS$^+$18] Riad S. Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *IEEE Symposium on Security and Privacy*, pages 926–943, 2018.

[XZZ$^+$19] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In *CRYPTO*, pages 733–764, 2019.

[ZGK$^+$17] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. A zero-knowledge version of vsql. *IACR Cryptol. ePrint Arch.*, 2017:1146, 2017.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSAM*, 1979.

[ZWZZ20] Jiaheng Zhang, Weijie Wang, Yinuo Zhang, and Yupeng Zhang. Doubly efficient interactive proofs for general arithmetic circuits with linear prover time. *IACR Cryptol. ePrint Arch.*, 2020:1247, 2020.

[ZXZS20] Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. Transparent polynomial delegation and its applications to zero knowledge proof. In *IEEE Symposium on Security and Privacy*, pages 859–876, 2020.

A Linear PCP for R1CS

In this section, we describe the linear PCP we use for R1CS. The construction is based on the quadratic arithmetic programs of Gennaro et al. [GGPR13], and is adapted from the 5-query linear PCP construction by Ben-Sasson et al. [BCG+13]. There are two minor differences in our construction:

- We remove the statement-dependent query and have the verifier introduce the statement-dependent components during verification. This yields a *4-query* linear PCP with shorter query length at the expense of a slightly more expensive verification step. A similar approach is used implicitly in [GGPR13, BCTV14b].
- The LPCP query-generation samples the random point from a smaller subset of the field. This introduces some knowledge error, but enables *perfect* HVZK. The construction of Ben-Sasson et al. provided statistical HVZK where the statistical distance was inversely proportional to the field size. The difference between statistical HVZK and perfect HVZK is negligible for super-polynomial size fields, but not for the moderate-size fields we use in this work.

For completeness, we provide the full description and analysis below. Our presentation and analysis is adapted from [BCG+13, Appendix E].

Construction A.1 (Linear PCP for R1CS [GGPR13, BCG+13, adapted])**.** Let $\mathcal{CS} = \{\mathcal{CS}_\kappa\}_{\kappa \in \mathbb{N}}$ be a family of R1CS instances over a finite field $\mathbb{F}$, where $\mathcal{CS}_\kappa = \big(n_\kappa, N_{g,\kappa}, N_{w,\kappa}, \{\mathbf{a}_{i,\kappa}, \mathbf{b}_{i,\kappa}, \mathbf{c}_{i,\kappa}\}_{i \in [N_{g,\kappa}]}\big)$, $\mathbf{a}_{i,\kappa}, \mathbf{b}_{i,\kappa}, \mathbf{c}_{i,\kappa} \in \mathbb{F}^{N_{w,\kappa}+1}$ (and entries indexed from 0 to $N_{w,\kappa}$). For notational convenience, we write $n = n(\kappa)$ to denote a function where $n(\kappa) = n_\kappa$ for all $\kappa \in \mathbb{N}$. We define $N_g = N_g(\kappa)$, $N_w = N_w(\kappa)$, $\mathbf{a}_i = \mathbf{a}_i(\kappa)$, $\mathbf{b}_i = \mathbf{b}_i(\kappa)$ and $\mathbf{c}_i = \mathbf{c}_i(\kappa)$ similarly. We additionally define the following components:

- Let $S = \{\alpha_1, \ldots, \alpha_{N_g}\} \subset \mathbb{F}$ be an arbitrary subset of $\mathbb{F}$.
- For each $i \in \{0, \ldots, N_w\}$, let $A_i, B_i, C_i \colon \mathbb{F} \to \mathbb{F}$ be the unique polynomial of degree $N_g - 1$ where for all $j \in [N_g]$,
$$A_i(\alpha_j) = \mathbf{a}_{j,i}, \quad B_i(\alpha_j) = \mathbf{b}_{j,i}, \quad C_i(\alpha_j) = \mathbf{c}_{j,i}.$$
- Let $Z_S \colon \mathbb{F} \to \mathbb{F}$ be the polynomial $Z_S(z) := \prod_{j \in [N_g]}(z - \alpha_j)$. Namely, Z_S is the polynomial whose roots are the elements of S.

The 4-query linear PCP $\Pi_{\mathsf{LPCP}} = (\mathcal{Q}_{\mathsf{LPCP}}, \mathcal{P}_{\mathsf{LPCP}}, \mathcal{V}_{\mathsf{LPCP}})$ for $\mathcal{CS}$ is defined as follows:

- $\mathcal{Q}_{\mathsf{LPCP}}(1^\kappa)$: On input $\kappa \in \mathbb{N}$, sample $\tau \xleftarrow{\mathrm{R}} \mathbb{F} \setminus S$. Let $\mathbf{a} = (A_1(\tau), \ldots, A_n(\tau))$, $\mathbf{b} = (B_1(\tau), \ldots, B_n(\tau))$, and $\mathbf{c} = (C_1(\tau), \ldots, C_n(\tau))$. Output the state $\mathsf{st} = (A_0(\tau), B_0(\tau), C_0(\tau), \mathbf{a}, \mathbf{b}, \mathbf{c}, Z_S(\tau))$ and the query matrix
$$\mathbf{Q} = \begin{bmatrix} Z_S(\tau) & 0 & 0 & A_{n+1}(\tau) & \cdots & A_{N_w}(\tau) & 0 & 0 & \cdots & 0 \\ 0 & Z_S(\tau) & 0 & B_{n+1}(\tau) & \cdots & B_{N_w}(\tau) & 0 & 0 & \cdots & 0 \\ 0 & 0 & Z_S(\tau) & C_{n+1}(\tau) & \cdots & C_{N_w}(\tau) & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & \tau & \cdots & \tau^{N_g} \end{bmatrix}^{\mathsf{T}} \in \mathbb{F}^{(4+N_w+N_g-n)\times 4}. \tag{A.1}$$

- $\mathcal{P}_{\mathsf{LPCP}}(1^\kappa, \mathbf{x}, \mathbf{w})$: On input $\kappa \in \mathbb{N}$ and an instance $(\mathbf{x}, \mathbf{w})$ where $\mathcal{CS}_\kappa(\mathbf{x}, \mathbf{w}) = 1$, sample $\delta_1, \delta_2, \delta_3 \xleftarrow{\text{R}} \mathbb{F}$. Construct polynomials $A, B, C \colon \mathbb{F} \to \mathbb{F}$, each of degree N_g, where

$$\begin{aligned} A(z) &:= \delta_1 Z_S(z) + A_0(z) + \sum\nolimits_{i \in [N_w]} w_i A_i(z) \\ B(z) &:= \delta_2 Z_S(z) + B_0(z) + \sum\nolimits_{i \in [N_w]} w_i B_i(z) \\ C(z) &:= \delta_3 Z_S(z) + C_0(z) + \sum\nolimits_{i \in [N_w]} w_i C_i(z). \end{aligned} \tag{A.2}$$

  Let $H(z) := (A(z)B(z) - C(z))/Z_S(z)$, and let $\mathbf{h} = (h_0, \ldots, h_{N_g}) \in \mathbb{F}^{N_g+1}$ be the coefficients of H. Parse $\mathbf{w}^\mathsf{T} = [\mathbf{x}^\mathsf{T} \mid \tilde{\mathbf{w}}^\mathsf{T}]$ Output the proof vector $\boldsymbol{\pi} = (\delta_1, \delta_2, \delta_3, \tilde{\mathbf{w}}, \mathbf{h}) \in \mathbb{F}^{4+N_w+N_g-n}$.

- $\mathcal{V}_{\mathsf{LPCP}}(\mathsf{st}, \mathbf{x}, \mathbf{a})$: On input $\mathsf{st} = (a_0, b_0, c_0, \mathbf{a}, \mathbf{b}, \mathbf{c}, z)$, $\mathbf{x} \in \mathbb{F}^n$ and $\mathbf{a} \in \mathbb{F}^4$, the verifier computes $a_1' = a_1 + a_0 + \mathbf{x}^\mathsf{T}\mathbf{a}$, $a_2' = a_2 + b_0 + \mathbf{x}^\mathsf{T}\mathbf{b}$, and $a_3' = a_3 + c_0 + \mathbf{x}^\mathsf{T}\mathbf{c}$. It accepts if

$$a_1' a_2' - a_3' - a_4 z = 0. \tag{A.3}$$

Theorem A.2 (Linear PCP for QAPs)**.** *Construction A.1 is complete, has knowledge error $2N_g/(|\mathbb{F}| - N_g)$, and is perfect HVZK.*

Proof. Let $\mathcal{CS} = \{\mathcal{CS}_\kappa\}_{\kappa \in \mathbb{N}}$ be an R1CS system over $\mathbb{F}$. We consider each property separately:

- **Completeness:** Take any $\kappa \in \mathbb{N}$ and $(\mathbf{x}, \mathbf{w})$ where $\mathcal{CS}_\kappa(\mathbf{x}, \mathbf{w}) = 1$. Let $(\mathsf{st}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}(1^\kappa)$, $\boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(1^\kappa, \mathbf{x}, \mathbf{w})$, $\mathbf{a} \leftarrow \mathbf{Q}^\mathsf{T}\boldsymbol{\pi}$. Consider the value of $\mathcal{V}_{\mathsf{LPCP}}(\mathsf{st}, \mathbf{x}, \mathbf{a})$. Let a_1', a_2', a_3' be the values computed by $\mathcal{V}_{\mathsf{LPCP}}$. By definition,

$$\begin{aligned} a_1' &= a_1 + a_0 + \mathbf{x}^\mathsf{T}\mathbf{a} \\ &= \delta_1 Z_S(\tau) + A_0(\tau) + \sum_{i \in [n]} x_i A_i(\tau) + \sum_{i \in [N_w - n]} w_{n+i} A_{n+i}(\tau) \\ &= \delta_1 Z_S(\tau) + A_0(\tau) + \sum_{i \in [N_w]} w_i A_i(\tau) \\ &= A(\tau). \end{aligned}$$

  since $w_i = x_i$ for $i \in [n]$, A is the polynomial in Eq. (A.2), and $\tau \in \mathbb{F} \setminus S$ is the element sampled by $\mathcal{Q}_{\mathsf{LPCP}}$. Similarly, we have that $a_2' = B(\tau)$ and $a_3' = C(\tau)$. Finally $a_4 = h_0 + \sum_{i \in [N_g]} h_i \tau^i = H(\tau)$, where $H(z) = (A(z)B(z) - C(z))/Z_S(z)$ is the polynomial constructed by the prover. The verification procedure now computes

$$a_1' a_2' - a_3' - a_4 z = A(\tau)B(\tau) - C(\tau) - H(\tau)Z_S(\tau) = 0,$$

  by definition of the polynomial H. Completeness follows.

- **Knowledge:** Define $\mathcal{E}_{\mathsf{LPCP}}^{\langle \boldsymbol{\pi}^*, \cdot \rangle}$ to be the algorithm that on input a statement $\mathbf{x}$ and given linear access to a proof vector $\boldsymbol{\pi}^* = (\delta_1^*, \delta_2^*, \delta_3^*, \tilde{\mathbf{w}}^*, \mathbf{h}^*)$, outputs $\mathbf{w}^\mathsf{T} = [\mathbf{x}^\mathsf{T} \mid (\tilde{\mathbf{w}}^*)^\mathsf{T}] \in \mathbb{F}^{N_w}$. To show that this extractor works, take any $\boldsymbol{\pi}^* = (\delta_1^*, \delta_2^*, \delta_3^*, \tilde{\mathbf{w}}^*, \mathbf{h}^*)$ where

$$\Pr[\mathcal{V}_{\mathsf{LPCP}}(\mathsf{st}, \mathbf{x}, \mathbf{Q}^\mathsf{T}\boldsymbol{\pi}^*) = 1 : (\mathsf{st}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}(1^\kappa)] > \frac{2N_g}{|\mathbb{F}| - N_g}.$$

We use $\boldsymbol{\pi}^*$ and $\mathcal{CS}$ to define polynomials $A, B, C, H \colon \mathbb{F} \to \mathbb{F}$:

$$
\begin{aligned}
A(z) &= \delta_1^* Z_S(z) + A_0(z) + \sum_{i \in [n]} x_i A_i(z) + \sum_{i \in [N_w - n]} \tilde{w}_i^* A_{n+i}(z) \\
B(z) &= \delta_2^* Z_S(z) + B_0(z) + \sum_{i \in [n]} x_i B_i(z) + \sum_{i \in [N_w - n]} \tilde{w}_i^* B_{n+i}(z) \\
C(z) &= \delta_3^* Z_S(z) + C_0(z) + \sum_{i \in [n]} x_i C_i(z) + \sum_{i \in [N_w - n]} \tilde{w}_i^* C_{n+i}(z) \\
H(z) &= h_0^* + \sum_{i \in [N_g]} h_i^* z^i
\end{aligned}
$$

Let $\mathbf{Q}$ be the query matrix output by $\mathcal{Q}_{\mathsf{LPCP}}$, $\mathbf{a} \leftarrow \mathbf{Q}^{\mathsf{T}} \boldsymbol{\pi}^*$ and a_1', a_2', a_3' be the components computed by $\mathcal{V}_{\mathsf{LPCP}}$. By construction, $a_1' = A(\tau)$, $a_2' = B(\tau)$, $a_3' = C(\tau)$ and $a_4 = H(\tau)$. Define the polynomial $P \colon \mathbb{F} \to \mathbb{F}$ where $P(z) = A(z)B(z) - C(z) - H(z)Z_S(z)$. By construction, $\deg(P) \leq 2N_g$. Next, $\mathcal{V}_{\mathsf{LPCP}}$ accepts if $a_1' a_2' - a_3' - a_4 z = 0$, where $z = Z(\tau)$, or equivalently, if

$$0 = A(\tau)B(\tau) - C(\tau) - H(\tau)Z_S(\tau) = P(\tau). \tag{A.4}$$

Suppose Eq. (A.4) holds with probability $\varepsilon > 2N_g / (|\mathbb{F}| - N_g)$; that is, the verifier accepts with probability greater than ε. Since $\mathcal{Q}_{\mathsf{LPCP}}$ samples τ uniformly from $\mathbb{F} \setminus S$ and $\deg(P) \leq 2N_g$, we conclude by the Schwartz-Zippel lemma (Lemma 2.1) that $P \equiv 0$. In particular, this means that for all $j \in [N_g]$,

$$P(\alpha_j) = A(\alpha_j)B(\alpha_j) - C(\alpha_j) = 0,$$

since $Z_S(\alpha_j) = 0$ for all $j \in [N_g]$. Equivalently, this means that $A(\alpha_j)B(\alpha_j) = C(\alpha_j)$ for all $j \in [N_g]$. By construction of A, B, C, this means that

$$[1 \mid \tilde{\mathbf{u}}^{\mathsf{T}}]\mathbf{a}_j \cdot [1 \mid \tilde{\mathbf{u}}^{\mathsf{T}}]\mathbf{b}_j = [1 \mid \tilde{\mathbf{u}}^{\mathsf{T}}]\mathbf{c}_j,$$

where $\tilde{\mathbf{u}}^{\mathsf{T}} = [\mathbf{x}^{\mathsf{T}} \mid (\tilde{\mathbf{w}}^*)^{\mathsf{T}}]$. Since this holds for all $j \in [N_g]$, we have that $\mathcal{CS}_\kappa(\mathbf{x}, \tilde{\mathbf{w}}^*) = 1$, as required.

- **HVZK:** We first construct a simulator $\mathcal{S}_{\mathsf{LPCP}} = (\mathcal{S}_{\mathsf{LPCP},1}, \mathcal{S}_{\mathsf{LPCP},2})$:
  - $\mathcal{S}_{\mathsf{LPCP},1}(1^\kappa)$: The statement-independent algorithm samples $(\widetilde{\mathsf{st}}, \tilde{\mathbf{Q}}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}(1^\kappa)$. It outputs $\widetilde{\mathsf{st}}$, $\tilde{\mathbf{Q}}$, and $\mathsf{st}_{\mathcal{S}} = \widetilde{\mathsf{st}}$.
  - $\mathcal{S}_{\mathsf{LPCP},2}(\mathsf{st}_{\mathcal{S}}, \mathbf{x})$: On input the state $\mathsf{st}_{\mathcal{S}} = (\tilde{a}_0, \tilde{b}_0, \tilde{c}_0, \tilde{\mathbf{a}}, \tilde{\mathbf{b}}, \tilde{\mathbf{c}}, \tilde{z})$ and the statement $\mathbf{x}$, the statement-dependent algorithm samples $\tilde{a}_1, \tilde{a}_2, \tilde{a}_3 \xleftarrow{\text{R}} \mathbb{F}$. It computes $\tilde{a}_1' = \tilde{a}_1 + \tilde{a}_0 + \mathbf{x}^{\mathsf{T}}\tilde{\mathbf{a}}$, $\tilde{a}_2' = \tilde{a}_2 + \tilde{b}_0 + \mathbf{x}^{\mathsf{T}}\tilde{\mathbf{b}}$, and $\tilde{a}_3' = \tilde{a}_3 + \tilde{c}_0 + \mathbf{x}^{\mathsf{T}}\tilde{\mathbf{c}}$. Compute $\tilde{a}_4 = z^{-1}(\tilde{a}_1'\tilde{a}_2' - \tilde{a}_3')$. It outputs $\tilde{\mathbf{a}} = (\tilde{a}_1, \tilde{a}_2, \tilde{a}_3, \tilde{a}_4)$.

  To complete the proof, it suffices to show that the simulated distribution is identical to the real distribution for any $(\mathbf{x}, \mathbf{w})$ where $\mathcal{CS}_\kappa(\mathbf{x}, \mathbf{w}) = 1$. By construction, the verification state and query matrix (output by $\mathcal{S}_{\mathsf{LPCP},1}$) are identically distributed in the two cases, so it suffices to analyze the distribution of the responses. Let $(\mathsf{st}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}(1^\kappa)$, $\boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(1^\kappa, \mathbf{x}, \mathbf{w})$, and $\mathbf{a} \leftarrow \mathbf{Q}^{\mathsf{T}}\boldsymbol{\pi}$. Write $\mathsf{st} = (a_0, b_0, c_0, \mathbf{a}, \mathbf{b}, \mathbf{c}, z)$. First, $z = Z_S(\tau)$ for some $\tau \in \mathbb{F} \setminus S$. Since $Z_S(x) = \prod_{\alpha \in S}(x - \alpha)$ and $\tau \notin S$, we have that $z = Z_S(\tau) \neq 0$. Then the following holds:

  - In the real distribution, Eq. (A.3) holds (by completeness). Since $z \neq 0$, the value of a_4 is uniquely defined given a_1, a_2, a_3 and st. The value of a_4 that satisfies Eq. (A.3) precisely coincides with the value $\tilde{a}_4$ sampled by $\mathcal{S}_{\mathsf{LPCP},2}$ (for the choice of $\tilde{a}_1, \tilde{a}_2, \tilde{a}_3$ chosen by the simulator).
  - In the real distribution, $a_1 = \delta_1 Z_S(\tau) + \sum_{i \in [N_w - n]} w_{n+i} A_{n+i}(\tau)$, where δ_1 is uniform over $\mathbb{F}$ and independent of all other components. Since $Z_S(\tau) \neq 0$, this means a_1 is uniform over $\mathbb{F}$. A similar argument holds for a_2 and a_3 (by appealing to the randomness of δ_2 and δ_3, respectively). This is precisely the distribution of $\tilde{a}_1, \tilde{a}_2, \tilde{a}_3$ in the simulation.

Thus, the simulated response is identically distributed as the real response, and perfect HVZK holds. □

Remark A.3 (Knowledge against Affine Strategies)**.** While our compiler only requires a linear PCP with knowledge against linear strategies (Theorem 3.21), we can easily modify the linear PCP from Construction 3.1 to provide knowledge against affine prover strategies *without* increasing the query complexity. This means that we can base security on the *weaker* conjecture that Construction 3.11 is "affine-only." Using the modified linear PCP comes at a very slight increase in the concrete cost of the verifier, and has no effect on the prover complexity. Since we believe that Conjecture 3.15 holds, we do not use this modified linear PCP in our concrete implementation; we mainly present the observation below for completeness.

Previously, Bitansky et al. [BCI+13] provide a generic approach that compiles any linear PCP with soundness (resp., knowledge) against linear strategies into a 2-message linear interactive proof with soundness (resp., knowledge) against affine strategies. This compiler introduces an extra linearity check: namely, an additional query that is a random linear combination of the remaining queries. While we can apply this technique directly to Construction A.1, it increases the number of queries of the linear PCP. Here, we observe that we can remove the need for an extra query by exploiting the specific structure of the linear PCP in Construction A.1. We describe the approach generally for any linear PCP whose query matrix $\mathbf{Q}^\mathsf{T}$ contains as one of its columns an elementary basis vector (or a scaled version thereof) and whose verification procedure is a low-degree algebraic circuit. Both properties hold for Construction A.1. The argument proceeds as follows:

- Let m be the query length of the linear PCP, k be the number of queries, and $(\boldsymbol{\pi}, \mathbf{b})$ be an affine strategy where $\boldsymbol{\pi} \in \mathbb{F}^m$, $\mathbf{b} \in \mathbb{F}^k$, In this case, given the query matrix $\mathbf{Q} \in \mathbb{F}^{m \times k}$, the responses are computed as $\mathbf{a} \leftarrow \mathbf{Q}^\mathsf{T}\boldsymbol{\pi} + \mathbf{b}$.

- Suppose that the j^{th} column of $\mathbf{Q}^\mathsf{T}$ is a basis vector $\mathbf{e}_i \in \mathbb{F}^k$. In this case, an affine strategy of the form $(\boldsymbol{\pi}, \delta\mathbf{e}_i)$ is *equivalent* to a *linear* strategy $\boldsymbol{\pi}'$ where $\pi_\ell = \pi'_\ell$ for all $\ell \neq j$ and $\pi'_j = \pi_j + \delta$. Since the linear PCP has knowledge soundness against linear strategies, it must also have knowledge soundness against affine strategies of the form $(\boldsymbol{\pi}, \delta\mathbf{e}_i)$.

- To extend such a linear PCP to provide soundness (resp., knowledge) against arbitrary affine strategies, we embed a random linear combination of the other queries into the i^{th} query. In more detail, let $\mathbf{q}_1, \ldots, \mathbf{q}_k$ be the linear PCP queries sampled by $\mathcal{Q}_{\mathsf{LPCP}}$ (i.e., these are the rows of $\mathbf{Q}^\mathsf{T}$). The modified query-generation algorithm first samples $\gamma_\ell \xleftarrow{\text{R}} \mathbb{F}$ for each $\ell \in [k]$. It computes $\mathbf{q}'_i = \mathbf{q}_i + \sum_{\ell \neq i} \gamma_\ell \mathbf{q}_\ell$ and outputs $(\mathbf{q}_1, \ldots, \mathbf{q}_{i-1}, \mathbf{q}'_i, \mathbf{q}_{i+1}, \ldots, \mathbf{q}_k)$ as its updated set of queries. The coefficients $\gamma_1, \ldots, \gamma_k$ are included as part of the verification state st.

- Given a set of responses $\mathbf{a} \in \mathbb{F}^k$, the modified verification algorithm first computes $a'_i \leftarrow a_i - \sum_{\ell \neq i} \gamma_i a_i$. It then runs the original verification algorithm $\mathcal{V}_{\mathsf{LPCP}}$ on $(a_1, \ldots, a_{i-1}, a'_i, a_{i+1}, \ldots, a_k)$.

Completeness of the above construction is immediate. To see that it also provides soundness (resp., knowledge) against arbitrary affine strategies, we use the assumption that the linear PCP has an algebraic verifier. In this case, the verification procedure can be described by checking whether a multivariate polynomial p in the responses is nonzero or not.[16] Consider an arbitrary affine strategy $(\boldsymbol{\pi}, \mathbf{b})$, and let $a_1, \ldots, a_k$ be the responses for the modified linear PCP defined above. In particular, $a_\ell = \mathbf{q}_\ell^\mathsf{T}\boldsymbol{\pi} + b_\ell$ for each $\ell \neq i$ and $a_i = (\mathbf{q}'_i)^\mathsf{T}\boldsymbol{\pi} + b_i$. The verification relation first computes

$$a'_i = a_i - \sum_{\ell \neq i} \gamma_\ell a_\ell = \mathbf{q}_i^\mathsf{T}\boldsymbol{\pi} + b_i - \sum_{\ell \neq i} \gamma_\ell b_\ell.$$

It then checks the polynomial relation

$$p(a_1, \ldots, a_{i-1}, a'_i, a_{i+1}, \ldots, a_k) \stackrel{?}{=} 0.$$

[16] The verification procedure may check multiple such polynomial relations (and may more generally be modeled as an arithmetic circuit over $\mathbb{F}$). The analysis described here directly extends to these settings.

Fixing $\mathbf{q}_1, \ldots, \mathbf{q}_k$, $\boldsymbol{\pi}$, and $\mathbf{b}$ (all of which are independent of the γ_ℓ's), we can view p as a polynomial in the variables $\gamma_1, \ldots, \gamma_k$ for $\ell \neq i$. Since $\mathcal{Q}_{\mathsf{LPCP}}$ samples γ_i uniformly and independently over $\mathbb{F}$, we appeal to the Schwartz-Zippel lemma and conclude that the probability that the verifier accepts is at most $d/|\mathbb{F}|$, where d is the degree of a_i' in p. Thus, the only possible strategies $(\boldsymbol{\pi}, \mathbf{b})$ where the prover can have advantage better than $d/|\mathbb{F}|$ are those where $b_\ell = 0$ for all $\ell \neq i$. But by our second observation above, soundness (resp., knowledge) against strategies of the form $(\boldsymbol{\pi}, \delta\mathbf{e}_i)$ is implied by soundness (resp., knowledge) against linear strategies when the query matrix $\mathbf{Q}^\mathsf{T}$ contains a basis vector $\mathbf{e}_i$ as one of its columns.

In the particular case of Construction A.1, we can see by inspection of Eq. (A.1) that one of its columns of the query matrix $\mathbf{Q}^\mathsf{T}$ is an elementary basis vector $\mathbf{e}_i$ ($i = 4$ in this case). The degree d of a_i in the verification relation (Eq. (A.3)) is $d = 1$, so the soundness error (resp., knowledge error) of the modified construction described above is $\varepsilon + 1/|\mathbb{F}|$, where ε is the soundness (resp., knowledge) error of the original construction (see Theorem A.2).

B Linear PCP and zkSNARK Analysis

In this section, we provide the formal analysis of our linear-only vector encryption scheme and resulting zkSNARKs.

B.1 Analysis of Construction 3.11 (Vector Encryption)

In this section, we provide the formal analysis of our candidate linear-only vector encryption scheme (Construction 3.11) from Section 3.3.

B.1.1 Proof of Theorem 3.12 (Additive Homomorphism)

We start with the concrete statement. Take any collection of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k \in R_p^\ell$ and scalars $y_1, \ldots, y_k \in R_p$. Let $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{v}_i)$ for each $i \in [k]$. Define the ciphertext $\mathsf{ct}^* \leftarrow \mathsf{Add}(\mathsf{pp}, \{\mathsf{ct}_i\}_{i\in[k]}, \{y_i\}_{i\in[k]})$. Here, $\mathsf{pp} = (\mathbf{A}, \mathbf{D})$, $\mathsf{sk} = (\mathbf{S}, \mathbf{T})$, $\mathsf{ct}_i = (\mathbf{a}_i, \mathbf{c}_i)$ and $\mathsf{ct}^* = (\mathbf{a}^*, \mathbf{c}^*)$. By construction, $\mathbf{D} = \mathbf{S}^\mathsf{T}\mathbf{A} + p\mathbf{E}^\mathsf{T}$,

$$\mathbf{a}^* = \sum_{i\in[k]} y_i \mathbf{a}_i + \mathbf{A}\mathbf{r} + p\mathbf{e}_a, \text{ and } \mathbf{c}^* = \sum_{i\in[k]} y_i \mathbf{c}_i + \mathbf{D}\mathbf{r} + p\mathbf{e}_c$$

For each $i \in [k]$, we also have $\mathbf{c}_i = \mathbf{S}^\mathsf{T}\mathbf{a}_i + p\mathbf{e}_i + \mathbf{u}_i$, where $\mathbf{u}_i^\mathsf{T} = [\mathbf{v}_i^\mathsf{T} \mid (\mathbf{T}\mathbf{v}_i)^\mathsf{T}]$. Thus,

$$\mathbf{c}^* = \sum_{i\in[k]} \left(y_i \mathbf{S}^\mathsf{T}\mathbf{a}_i + y_i p\mathbf{e}_i + y_i \mathbf{u}_i \right) + \mathbf{S}^\mathsf{T}\mathbf{A}\mathbf{r} + p\mathbf{E}^\mathsf{T}\mathbf{r} + p\mathbf{e}_c. \tag{B.1}$$

Consider the output of $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}^*)$. The decryption algorithm starts by computing (over R_q)

$$\mathbf{z}^* = \mathbf{c}^* - \mathbf{S}^\mathsf{T}\mathbf{a}^* = \sum_{i\in[k]} y_i \mathbf{u}_i + p \left(\sum_{i\in[k]} y_i \mathbf{e}_i + \mathbf{E}^\mathsf{T}\mathbf{r} + \mathbf{e}_c - \mathbf{S}^\mathsf{T}\mathbf{e}_a \right).$$

We write $\sum_{i\in[k]} y_i \mathbf{u}_i = p\tilde{\mathbf{u}} + \tilde{\mathbf{u}}' \in R^{\ell'}$ where $\tilde{\mathbf{u}}, \tilde{\mathbf{u}}' \in R^{\ell'}$ and $\|\tilde{\mathbf{u}}'\|_\infty < p/2$. Then,

$$\mathbf{z}^* = \sum_{i\in[k]} y_i \mathbf{u}_i \bmod p + p \underbrace{\left(\tilde{\mathbf{u}} + \sum_{i\in[k]} y_i \mathbf{e}_i + \mathbf{E}^\mathsf{T}\mathbf{r} + \mathbf{e}_c - \mathbf{S}^\mathsf{T}\mathbf{e}_a \right)}_{\mathbf{e}'} \in R_q^{\ell'}. \tag{B.2}$$

If $\mathbf{e}'$ satisfies $\|\mathbf{e}'\|_\infty < q/(2p) - 1/2$, then Eq. (B.2) holds over R (and not just modulo q). Then, the decryption algorithm computes

$$\mathbf{u}^\mathsf{T} = \sum_{i\in[k]} y_i \mathbf{u}_i^\mathsf{T} = \left[\sum_{i\in[k]} y_i \mathbf{v}_i^\mathsf{T} \;\middle|\; \sum_{i\in[k]} y_i (\mathbf{T}\mathbf{v}_i)^\mathsf{T} \right] \in R_p^{\ell'},$$

and outputs $\sum_{i\in[k]} y_i \mathbf{v}_i$, as required. Thus, it suffices to argue that $\|\mathbf{e}'\|_\infty < q/(2p) - 1/2$. We analyze each term in $\mathbf{e}'$ separately.

- By definition, $\tilde{\mathbf{u}} = 1/p \cdot (\sum_{i\in[k]} y_i \mathbf{u}_i - \sum_{i \in k} y_i \mathbf{u}_i \bmod p)$. This means that $\|\tilde{\mathbf{u}}\|_\infty \le 1/p \cdot \left\| \sum_{i\in[k]} y_i \mathbf{u}_i \right\|_\infty$. Since $\mathbf{u}_i \in R_p^{\ell'}$, this means $\|\mathbf{u}_i\|_\infty \le p/2$, so $\|\tilde{\mathbf{u}}\|_\infty \le \gamma_R \|\mathbf{y}\|_1 /2$.
- The entries of $\mathbf{e}_i \in R_q^{\ell'}$ are sampled from χ, which by assumption, is the product of d independent subgaussian distributions over $\mathbb{Z}$, each with parameter at most s. Since $R = \mathbb{Z}[x]/(x^d + 1)$, each component of $\sum_{i\in[k]} y_i \mathbf{e}_i$ is subgaussian with parameter $\gamma_R \|\mathbf{y}\|_2 s$. Thus, the magnitude of each component of $\sum_{i\in[k]} y_i \mathbf{e}_i$ is bounded by $\gamma_R \|\mathbf{y}\|_2 Cs$ with probability at least $1 - 2\exp(-\pi C^2)$. By a union bound, $\left\| \sum_{i\in[k]} y_i \mathbf{e}_i \right\|_\infty \le \gamma_R \|\mathbf{y}\|_2 Cs$ with probability at least $1 - 2d\ell' \exp(-\pi C^2)$.
- Since the entries of $\mathbf{E} \in R_q^{n\times \ell'}$ and $\mathbf{r} \in R_q^n$ are sampled from χ, the magnitude of each entry in $\mathbf{E}$ and $\mathbf{r}$ is bounded by Cs with probability at least $1 - 2d\exp(-\pi C^2)$. By a union bound, $\left\|\mathbf{E}^\mathsf{T} \mathbf{r}\right\|_\infty \le \gamma_R n C^2 s^2$ with probability $1 - 2dn\ell' \exp(-\pi C^2)$.
- Since $\mathbf{e}_c$ is sampled from $[-B, B]^{d\ell'}$, $\|\mathbf{e}_c\|_\infty \le B$.
- Since the entries of $\mathbf{S} \in R_q^{n\times \ell'}$ and $\mathbf{e}_a \in R_q^n$ are sampled from χ, we have that $\left\|\mathbf{S}^\mathsf{T} \mathbf{e}_a\right\|_\infty \le \gamma_R n C^2 s^2$ with probability at least $1 - 2dn\ell' \exp(-\pi C^2)$.

Again by a union bound, with probability at least $1 - (4n+2)d\ell' \exp(-\pi C^2)$,

$$\|\mathbf{e}'\|_\infty \le B + \gamma_R \|\mathbf{y}\|_2 Cs + \gamma_R \|\mathbf{y}\|_1 /2 + 2\gamma_R n C^2 s^2. \tag{B.3}$$

Taking $q > 2p\|\mathbf{e}'\|_\infty + p$ thus suffices for correctness. For the asymptotic statement, it suffices to consider $C = \omega(\log \lambda)$. □

B.1.2 Proof of Theorem 3.13 (CPA Security)

By a standard hybrid argument, the $\mathsf{MLWE}_{n,m,d,q,\chi}$ assumption implies that the following two distributions are computationally indistinguishable:

$$\left(\mathbf{A}', \mathbf{S}^\mathsf{T}\mathbf{A}' + (\mathbf{E}')^\mathsf{T}\right) \text{ and } \left(\mathbf{A}', (\mathbf{U}')^\mathsf{T}\right), \tag{B.4}$$

where $\mathbf{A}' \xleftarrow{\text{R}} R_q^{n\times m}$, $\mathbf{S} \leftarrow \chi^{n\times \ell'}$, $\mathbf{E}' \leftarrow \chi^{m\times \ell'}$, and $\mathbf{U}' \xleftarrow{\text{R}} R_q^{m\times \ell'}$ (for any $\ell' = \mathsf{poly}(\lambda)$). Semantic security follows immediately from this assumption. Formally, we use a hybrid argument:

- Hyb_0: This is the real semantic security experiment. Namely, the challenger samples $\mathbf{A} \xleftarrow{\text{R}} R_q^{n\times n}$, $\mathbf{S} \leftarrow \chi^{n\times \ell'}$, $\mathbf{E} \leftarrow \chi^{n\times \ell'}$, $\mathbf{T} \xleftarrow{\text{R}} R_p^{\tau \times \ell}$, $b \xleftarrow{\text{R}} \{0,1\}$, and computes $\mathbf{D} \leftarrow \mathbf{S}^\mathsf{T}\mathbf{A} + p\mathbf{E}^\mathsf{T}$. The challenger gives the public parameters $\mathsf{pp} = (\mathbf{A}, \mathbf{D})$ to $\mathcal{A}$. On the i^{th} oracle query $(\mathbf{v}_{i,0}, \mathbf{v}_{i,1})$, the challenger samples $\mathbf{a}_i \xleftarrow{\text{R}} R_q^n$, $\mathbf{e}_i \xleftarrow{\text{R}} \chi^{\ell'}$, computes $\mathbf{u}_i^\mathsf{T} = [\mathbf{v}_{i,b}^\mathsf{T} \mid (\mathbf{T}\mathbf{v}_{i,b})^\mathsf{T}]$, $\mathbf{c}_i \leftarrow \mathbf{S}^\mathsf{T}\mathbf{a}_i + p\mathbf{e}_i + \mathbf{u}_i$, and replies with the ciphertext $\mathsf{ct}_i = (\mathbf{a}_i, \mathbf{c}_i)$. At the end of the experiment, the adversary outputs a bit b'. The output of the experiment is 1 if $b' = b$ and 0 otherwise.
- Hyb_1: Same as Hyb_0, except the challenger samples $\mathbf{D} \xleftarrow{\text{R}} R_q^{\ell' \times n}$ in the public parameters, and for each of the adversary's oracle queries, the challenger computes $\mathbf{c}_i \leftarrow p\mathbf{r}_i + \mathbf{u}_i$ where $\mathbf{r}_i \xleftarrow{\text{R}} R_q^{\ell'}$.

We first argue that the outputs of Hyb_0 and Hyb_1 are computationally indistinguishable under the $\mathsf{MLWE}_{n,m,d,q,\chi}$ assumption. Let $(\mathbf{A}', \mathbf{Z}')$ be the MLWE challenge and write $\mathbf{A}' = [\mathbf{A} \mid \mathbf{a}_1 \mid \cdots \mid \mathbf{a}_Q]$, where $\mathbf{A} \in R_q^{n\times n}$ and $\mathbf{a}_1, \ldots, \mathbf{a}_Q \in R_q^n$. Analogously, write $\mathbf{Z}' = [\mathbf{Z} \mid \mathbf{z}_1 \mid \cdots \mid \mathbf{z}_Q]$. Consider a CPA security experiment where we set $\mathsf{pp} = (p\mathbf{A}, p\mathbf{Z})$ and respond to the adversary's oracle queries $(\mathbf{v}_{i,0}, \mathbf{v}_{i,1})$ with the ciphertext

$\mathsf{ct}_i = (p\mathbf{a}_i, p\mathbf{z}_i + \mathbf{u}_i)$. In this case, if $\mathbf{Z}' = \mathbf{S}^\mathsf{T}\mathbf{A}' + (\mathbf{E}')^\mathsf{T}$, then this perfectly simulates Hyb_0. If $\mathbf{Z}' \xleftarrow{\text{R}} R_q^{\ell' \times m}$, this perfectly simulates Hyb_1. (In particular, if $\mathbf{A}$ is uniform over $R_q^{n \times n}$, then so is $p\mathbf{A}$ since $\gcd(p,q) = 1$, and likewise for $p\mathbf{a}_i$ for each $i \in [Q]$). Thus, under $\mathsf{MLWE}_{n,m,d,q,\chi}$, the outputs of Hyb_0 and Hyb_1 are computationally indistinguishable.

To complete the proof, observe that in Hyb_1, the distinguishing advantage of every adversary is exactly $1/2$ since the challenge ciphertexts perfectly hide the message $\mathbf{v}_{i,b}$ (since the $\mathbf{r}_i$'s are uniform and independent over $R_q^{\ell'}$ and $\gcd(p,q) = 1$). □

B.1.3 Proof of Theorem 3.14 (Circuit Privacy)

We first construct a simulator $\mathcal{S}$. On input the security parameter λ, the public parameters $\mathsf{pp} = (\mathbf{A}, \mathbf{D})$, the secret key $\mathsf{sk} = (\mathbf{S}, \mathbf{T})$, and a vector $\mathbf{v} \in R_p^\ell$, the simulator proceeds as follows:

1. Sample $\mathbf{a} \xleftarrow{\text{R}} R_q^n$ and $\tilde{\mathbf{e}} \leftarrow [-B, B]^{d\ell'}$.
2. Compute $\mathbf{u}^\mathsf{T} = [\mathbf{v}^\mathsf{T} \mid (\mathbf{T}\mathbf{v})^\mathsf{T}] \in R_p^{\ell'}$ and output the ciphertext $\mathsf{ct}^* = (\mathbf{a}, \mathbf{S}^\mathsf{T}\mathbf{a} + p\tilde{\mathbf{e}} + \mathbf{u})$.

We show that the output of the above simulator is computationally indistinguishable from the output of the Add algorithm. We proceed with a hybrid argument:

- Hyb_0: This is the real circuit privacy experiment. Namely, the challenger begins by sampling $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$ where $\mathsf{pp} = (\mathbf{A}, \mathbf{D})$ and $\mathsf{sk} = (\mathbf{S}, \mathbf{T})$. The challenger gives pp and sk to the adversary $\mathcal{A}$, which outputs a sequence of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$. The challenger computes $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{v}_i)$ for each $i \in [k]$, where each ct_i can be written as $\mathsf{ct}_i = (\mathbf{a}_i, \mathbf{c}_i)$, for $\mathbf{c}_i = \mathbf{S}^\mathsf{T}\mathbf{a}_i + p\mathbf{e}_i + \mathbf{u}_i$, $\mathbf{e}_i \in R_q^{\ell'}$ and $\mathbf{u}_i^\mathsf{T} = [\mathbf{v}_i^\mathsf{T} \mid (\mathbf{T}\mathbf{v}_i)^\mathsf{T}]$. The challenger gives $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$ to $\mathcal{A}$. Adversary $\mathcal{A}$ outputs a set of coefficients $y_1, \ldots, y_k \in R_p$. The challenger computes ct_0^* and ct_1^* as follows:
  - For ct_0^*, the challenger uses the Add algorithm. Namely, it samples $\mathbf{r} \leftarrow \chi^n$, $\mathbf{e}_a \leftarrow \chi^n$, $\mathbf{e}_c \leftarrow [-B, B]^{d\ell'}$, computes $\mathbf{a}_0^* \leftarrow \sum_{i \in [k]} y_i \mathbf{a}_i + \mathbf{A}\mathbf{r} + p\mathbf{e}_a$ and $\mathbf{c}_0^* \leftarrow \sum_{i \in [k]} y_i \mathbf{c}_i + \mathbf{D}\mathbf{r} + p\mathbf{e}_c$, and sets $\mathsf{ct}_0^* = (\mathbf{a}_0^*, \mathbf{c}_0^*)$.
  - For ct_1^*, the challenger uses the simulator $\mathcal{S}$. Namely, it samples $\mathbf{a}_1^* \xleftarrow{\text{R}} R_q^n$ and $\tilde{\mathbf{e}} \leftarrow [-B, B]^{d\ell'}$ and computes $\mathbf{c}_1^* \leftarrow \mathbf{S}^\mathsf{T}\mathbf{a}_1^* + p\tilde{\mathbf{e}} + \mathbf{u}$, where $\mathbf{u}^\mathsf{T} = [\sum_{i \in [k]} y_i \mathbf{v}_i^\mathsf{T} \mid \sum_{i \in [k]} y_i (\mathbf{T}\mathbf{v}_i)^\mathsf{T}] \in R_p^{\ell'}$. Finally, it sets $\mathsf{ct}_1^* = (\mathbf{a}_1^*, \mathbf{c}_1^*)$.

  The challenger samples $b \xleftarrow{\text{R}} \{0, 1\}$ and gives ct_b^* to the adversary. At the end of the experiment, the adversary outputs a bit $b' \in \{0, 1\}$, and the output of the experiment is 1 if $b' = b$.

- Hyb_1: Same as Hyb_0, except the challenger computes $\mathbf{c}_0^* \leftarrow \mathbf{S}^\mathsf{T}\mathbf{a}_0^* + p\mathbf{e}_c + \mathbf{u}$ where

$$\mathbf{u}^\mathsf{T} = \left[\sum_{i \in [k]} y_i \mathbf{v}_i^\mathsf{T} \,\middle|\, \sum_{i \in [k]} y_i (\mathbf{T}\mathbf{v}_i)^\mathsf{T} \right] \in R_p^{\ell'}.$$

- Hyb_2: Same as Hyb_1 except the challenger samples $\mathbf{a}_0^* \xleftarrow{\text{R}} R_q^n$.

For an adversary $\mathcal{A}$, we write $\mathsf{Hyb}_i(\mathcal{A})$ to denote the output distribution of Hyb_i with adversary $\mathcal{A}$. We argue that the output distribution of each pair of adjacent hybrids are computationally (or statistically) indistinguishable. Finally, we show that for all adversaries $\mathcal{A}$, $\Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1] = 1/2$.

Lemma B.1. *For all adversaries $\mathcal{A}$ (restricted to strategies in S), the statistical distance between $\mathsf{Hyb}_0(\mathcal{A})$ and $\mathsf{Hyb}_1(\mathcal{A})$ is ε (see Eq. (3.6)).*

Proof. The only difference between Hyb_0 and Hyb_1 is how $\mathbf{c}_0^*$ is constructed. As in the proof of Theorem 3.12 (see Eq. (B.1)), in Hyb_0,

$$\begin{aligned}\mathbf{c}_0^* &= \sum_{i\in[k]} \left(y_i\mathbf{S}^\mathsf{T}\mathbf{a}_i + y_i p\mathbf{e}_i + y_i\mathbf{u}_i\right) + \mathbf{S}^\mathsf{T}\mathbf{A}\mathbf{r} + p\mathbf{E}^\mathsf{T}\mathbf{r} + p\mathbf{e}_c \\ &= \mathbf{S}^\mathsf{T}\mathbf{a}_0^* + p\left[\tilde{\mathbf{u}} + \sum_{i\in[k]} y_i\mathbf{e}_i + \mathbf{E}^\mathsf{T}\mathbf{r} - \mathbf{S}^\mathsf{T}\mathbf{e}_a + \mathbf{e}_c\right] + \mathbf{u},\end{aligned}$$

where $\sum_{i\in[k]} y_i\mathbf{u}_i = p\tilde{\mathbf{u}} + \tilde{\mathbf{u}}' \in R^{\ell'}$, $\|\tilde{\mathbf{u}}'\|_\infty < p/2$ and $\tilde{\mathbf{u}}' = \mathbf{u} \bmod p$. Let $\mathbf{e}' = \tilde{\mathbf{u}} + \sum_{i\in[k]} y_i\mathbf{e}_i + \mathbf{E}^\mathsf{T}\mathbf{r} - \mathbf{S}^\mathsf{T}\mathbf{e}_a$. From the proof of Theorem 3.12, with probability at least $1-(4n+2)d\ell'\exp(-\pi C^2)$,

$$\|\mathbf{e}'\|_\infty \le \gamma_R \|\mathbf{y}\|_2 Cs + \gamma_R\|\mathbf{y}\|_1/2 + 2\gamma_R nC^2s^2.$$

In Hyb_1, $\mathbf{c}_0^* = \mathbf{S}^\mathsf{T}\mathbf{a}_0^* + p\mathbf{e}_c + \mathbf{u}$. By Lemma 2.2 and a union bound, the statistical distance between the distributions of $\mathbf{e}_c$ and $\mathbf{e}'+\mathbf{e}_c$ is at most $d\ell'\|\mathbf{e}'\|_\infty/B$. The claim follows. □

Lemma B.2. *Suppose that p, q are coprime. Under the $\mathsf{MLWE}_{n,m,d,q,\chi}$ assumption with $m = n$, for all efficient adversaries $\mathcal{A}$, $\mathsf{Hyb}_1(\mathcal{A})$ and $\mathsf{Hyb}_2(\mathcal{A})$ are computationally indistinguishable.*

Proof. The only difference between Hyb_1 and Hyb_2 is that in Hyb_1, $\mathbf{a}_0^* \leftarrow \sum_{i\in[k]} y_i\mathbf{a}_i + \mathbf{A}\mathbf{r} + p\mathbf{e}_a$ while in Hyb_2, $\mathbf{a}_0^* \xleftarrow{\text{R}} R_q^n$. This follows from the MLWE assumption. To see this, suppose there is an adversary $\mathcal{A}$ where $|\Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1]| \ge \varepsilon'$. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ for MLWE. Let $(\mathbf{A}, \mathbf{z})$ be the MLWE challenge where $\mathbf{A} \in R_q^{n\times n}$ and $\mathbf{z} \in R_q^n$. Algorithm $\mathcal{B}$ simulates an execution of Hyb_1 or Hyb_2 as follows:

- Sample $\mathbf{S} \leftarrow \chi^{n\times\ell'}$, $\mathbf{E} \leftarrow \chi^{n\times\ell'}$ and $\mathbf{T} \xleftarrow{\text{R}} R_p^{\tau\times\ell}$. Let $\mathbf{D} \leftarrow \mathbf{S}^\mathsf{T}(p\mathbf{A}^\mathsf{T}) + p\mathbf{E}^\mathsf{T}$, and set $\mathsf{pp} = (p\mathbf{A}^\mathsf{T}, \mathbf{D})$ and $\mathsf{sk} = (\mathbf{S}, \mathbf{T})$. Since $\mathbf{A}$ is uniform and $\gcd(p,q) = 1$, the matrix $p\mathbf{A}^\mathsf{T}$ is also uniform.
- Let $\mathbf{a}_0^* \leftarrow \sum_{i\in[k]} y_i\mathbf{a}_i + p\mathbf{z}$. Construct the remaining components $\mathbf{c}_0^*$, $\mathbf{a}_1^*$, and $\mathbf{c}_1^*$ as in Hyb_1 and Hyb_2.
- Sample $b \xleftarrow{\text{R}} \{0,1\}$ and give ct_b^* to $\mathcal{A}$. Let $b' \in \{0,1\}$ be the adversary's output. Output 1 if $b' = b$ and 0 otherwise.

If $\mathbf{z}^\mathsf{T} = \mathbf{s}^\mathsf{T}\mathbf{A} + \mathbf{e}^\mathsf{T}$, then $\mathbf{a}_0^* = \sum_{i\in[k]} y_i\mathbf{a}_i + p\mathbf{A}^\mathsf{T}\mathbf{s} + p\mathbf{e}$ where $\mathbf{s} \leftarrow \chi^n$ and $\mathbf{e} \leftarrow \chi^n$. This is precisely the distribution in Hyb_1. Conversely, if $\mathbf{z} \xleftarrow{\text{R}} R_q^n$, then $\mathbf{a}_0^*$ is uniform (since $\gcd(p,q) = 1$ and $\mathbf{z}$ is uniform and independent of the y_i and $\mathbf{a}_i$). This is precisely the distribution in Hyb_2. Thus, $\mathcal{B}$ breaks MLWE with advantage ε'. □

To conclude the proof, observe that ct_0^* and ct_1^* are identically distributed in Hyb_2. Thus, for all adversaries $\mathcal{A}$, $\Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1] = 1/2$. Together with Lemmas B.1 and B.2, this means that for all efficient adversaries $\mathcal{A}$, $\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1] \le 1/2 + \varepsilon + \mathsf{negl}(\lambda)$. For the asymptotic statement, we set $C = \omega(\log\lambda)$ and $B_1, B_2 = kp$. □

B.1.4 Proof of Theorem 3.19 (Modulus Switching)

Since $\mathbf{z} = \mathbf{c} - \mathbf{S}^\mathsf{T}\mathbf{a} \pmod q$, we can write $\mathbf{z} = \mathbf{c} - \mathbf{S}^\mathsf{T}\mathbf{a} + q\tilde{\mathbf{v}} \in R^{\ell'}$, where $\tilde{\mathbf{v}} \in R^{\ell'}$. Define the vector $\tilde{\mathbf{z}} = \mathbf{c}' - \mathbf{S}^\mathsf{T}\mathbf{a}' + q'\tilde{\mathbf{v}} \in R^{\ell'}$. Then,

$$\begin{aligned}\|\tilde{\mathbf{z}}\|_\infty &= \left\|\mathbf{c}' - \mathbf{S}^\mathsf{T}\mathbf{a}' + q'\tilde{\mathbf{v}}\right\|_\infty \\ &= \left\|\mathbf{c}' - \mathbf{S}^\mathsf{T}\mathbf{a}' + q'\tilde{\mathbf{v}} + (q'/q)(\mathbf{c} - \mathbf{S}^\mathsf{T}\mathbf{a} + q\tilde{\mathbf{v}} - \mathbf{c} + \mathbf{S}^\mathsf{T}\mathbf{a} - q\tilde{\mathbf{v}})\right\|_\infty \\ &\le \|\mathbf{c}' - (q'/q)\cdot\mathbf{c}\|_\infty + \left\|\mathbf{S}^\mathsf{T}\left(\mathbf{a}' - (q'/q)\cdot\mathbf{a}\right)\right\|_\infty + (q'/q)\cdot\|\mathbf{z}\|_\infty.\end{aligned}$$

We analyze each term separately:

- Since $\mathbf{c}' \leftarrow \mathsf{Scale}(\mathbf{c}, q, q', p)$, by definition of the Scale operation, $\|\mathbf{c}' - (q'/q) \cdot \mathbf{c}\|_\infty \le p/2$.
- Similarly, since $\mathbf{a}' = \mathsf{Scale}(\mathbf{a}, q', q, p)$, we have that $\|\mathbf{a}' - (q'/q) \cdot \mathbf{a}\|_\infty \le p/2$. The entries of $\mathbf{S} \in R_q^{n \times \ell'}$ are sampled from χ. Since χ is subgaussian with parameter s, the magnitude of each entry in $\mathbf{S}$ is bounded by Cs with probability at least $1 - 2d\exp(-\pi C^2)$. By a union bound over the components of $\mathbf{S}$, we have that $\left\|\mathbf{S}^\mathsf{T}\left(\mathbf{a}' - (q'/q) \cdot \mathbf{a}\right)\right\|_\infty \le \gamma_R n C s (p/2)$ with probability $1 - 2dn\ell' \exp(-\pi C^2)$.
- By assumption, $\|\mathbf{z}\|_\infty < q/2 - (1 + n\gamma_R Cs) \cdot (p/2) \cdot (q/q')$. Thus,
$$(q'/q) \cdot \|\mathbf{z}\|_\infty < q'/2 - (1 + n\gamma_R Cs) \cdot (p/2).$$

Thus, with probability $1 - 2dn\ell' \exp(-\pi C^2)$, $\|\tilde{\mathbf{z}}\|_\infty < q'/2$. Now, $\mathbf{z}' = \mathbf{c}' - \mathbf{S}^\mathsf{T}\mathbf{a}' = \tilde{\mathbf{z}} \pmod{q'}$. But if the entries of $\tilde{\mathbf{z}}$ are all bounded by $q'/2$, then it must be the case that $\mathbf{z}' = \mathbf{c}' - \mathbf{S}^\mathsf{T}\mathbf{a}' + q'\tilde{\mathbf{v}} = \tilde{\mathbf{z}} \in R$. Here, the relation is taken over the *ring* and not modulo q'. Working now modulo p, we have the following:

$$\mathbf{z}' = \mathbf{c}' - \mathbf{S}^\mathsf{T}\mathbf{a}' + q'\tilde{\mathbf{v}} = \mathbf{c} - \mathbf{S}^\mathsf{T}\mathbf{a} + q\tilde{\mathbf{v}} = \mathbf{z} \pmod{p},$$

since $\mathbf{c} = \mathbf{c}' \pmod{p}$, $\mathbf{a} = \mathbf{a}' \pmod{p}$, and $q' = q \pmod{p}$ by construction or by assumption. □

B.2 Proof of Theorem 3.23 (Zero Knowledge)

Let $\mathcal{S}_{\mathsf{LPCP}} = (\mathcal{S}_{\mathsf{LPCP},1}, \mathcal{S}_{\mathsf{LPCP},2})$ be the simulator for Π_{LPCP}, and $\mathcal{S}_{\mathsf{Enc}}$ be the circuit privacy simulator for Π_{Enc}. We use $\mathcal{S}_{\mathsf{LPCP}}$ and $\mathcal{S}_{\mathsf{Enc}}$ to construct a simulator $\mathcal{S}_{\mathsf{SNARK}} = (\mathcal{S}_{\mathsf{SNARK},1}, \mathcal{S}_{\mathsf{SNARK},2})$ for Π_{SNARK}:

- $\mathcal{S}_{\mathsf{SNARK},1}(1^\lambda, 1^\kappa)$: On input the security parameter λ and the system index κ, run $(\widetilde{\mathsf{st}}_{\mathsf{LPCP}}, \tilde{\mathbf{Q}}, \mathsf{st}_{\mathsf{LPCP},\mathcal{S}}) \leftarrow \mathcal{S}_{\mathsf{LPCP},1}(1^\kappa)$, where $\tilde{\mathbf{Q}} \in \mathbb{F}^{m \times k}$. Sample $(\widetilde{\mathsf{pp}}, \widetilde{\mathsf{sk}}) \leftarrow \mathsf{Setup}_{\mathsf{Enc}}(1^\lambda, 1^k)$ and compute $\widetilde{\mathsf{ct}}_i \leftarrow \mathsf{Encrypt}_{\mathsf{Enc}}(\widetilde{\mathsf{sk}}, \tilde{\mathbf{q}}_i^\mathsf{T})$ for each $i \in [m]$. Output the common reference string $\widetilde{\mathsf{crs}} = (\kappa, \widetilde{\mathsf{pp}}, \widetilde{\mathsf{ct}}_1, \ldots, \widetilde{\mathsf{ct}}_m)$, verification state $\widetilde{\mathsf{st}} = (\widetilde{\mathsf{st}}_{\mathsf{LPCP}}, \widetilde{\mathsf{sk}})$, and the simulation state $\mathsf{st}_{\mathcal{S}} = (\mathsf{st}_{\mathsf{LPCP},\mathcal{S}}, \widetilde{\mathsf{pp}}, \widetilde{\mathsf{sk}})$.
- $\mathcal{S}_{\mathsf{SNARK},2}(\mathsf{st}_{\mathcal{S}}, \mathbf{x})$: On input $\mathsf{st}_{\mathcal{S}} = (\mathsf{st}_{\mathsf{LPCP},\mathcal{S}}, \widetilde{\mathsf{pp}}, \widetilde{\mathsf{sk}})$ and the statement $\mathbf{x}$, compute $\tilde{\mathbf{a}} \leftarrow \mathcal{S}_{\mathsf{LPCP},2}(\mathsf{st}_{\mathsf{LPCP},\mathcal{S}}, \mathbf{x})$ and output the simulated proof $\tilde{\pi} = \mathcal{S}_{\mathsf{Enc}}(1^\lambda, \widetilde{\mathsf{pp}}, \widetilde{\mathsf{sk}}, \tilde{\mathbf{a}})$.

To complete the proof, we argue that the real distribution and the simulated distributions are computationally indistinguishable. This follows by a simple hybrid argument:

- Hyb_0: This is the real game. Namely, the challenger first samples a bit $b \xleftarrow{\text{R}} \{0,1\}$, $(\mathsf{crs}, \mathsf{st}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa)$, and $(\widetilde{\mathsf{crs}}, \widetilde{\mathsf{st}}, \mathsf{st}_{\mathsf{LPCP},\mathcal{S}}) \leftarrow \mathcal{S}_1(1^\lambda, 1^\kappa)$. If $b = 0$, it gives $(\mathsf{crs}, \mathsf{st})$ to the adversary, and otherwise, it gives $(\widetilde{\mathsf{crs}}, \widetilde{\mathsf{st}})$ to the adversary. After the adversary outputs a statement $\mathbf{x}$ and witness $\mathbf{w}$, the challenger first checks that $\mathcal{CS}_\kappa(\mathbf{x}, \mathbf{w}) = 1$ (aborting the experiment otherwise), and then computes $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathbf{x}, \mathbf{w})$ and $\tilde{\pi} \leftarrow \mathcal{S}_{\mathsf{SNARK},2}(\mathsf{st}_{\mathcal{S}}, \mathbf{x})$. It gives π to $\mathcal{A}$ if $b = 0$ and $\tilde{\pi}$ if $b = 1$. At the end of the experiment, the adversary outputs a bit $b' \in \{0,1\}$, and the output of the experiment is 1 if $b' = b$.
- Hyb_1: Same as Hyb_0 except the challenger constructs π using the circuit privacy simulator (in place of $\mathsf{Prove}(\mathsf{crs}, \mathbf{x}, \mathbf{w})$). Let $\mathbf{Q} \in \mathbb{F}^{m \times k}$ be the query matrix the challenger sampled to construct $\mathsf{crs} = (\kappa, \mathsf{pp}, \mathsf{ct}_1, \ldots, \mathsf{ct}_m)$ and let $\mathsf{st} = (\mathsf{st}_{\mathsf{LPCP}}, \mathsf{sk})$ be the corresponding verification state. To construct π, the challenger now computes $\boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(1^\kappa, \mathbf{x}, \mathbf{w})$ and sets $\pi \leftarrow \mathcal{S}_{\mathsf{Enc}}(1^\lambda, \mathsf{pp}, \mathsf{sk}, \mathbf{a})$, where $\mathbf{a} = \sum_{i \in [m]} \pi_i \mathbf{q}_i^\mathsf{T} = \mathbf{Q}^\mathsf{T}\boldsymbol{\pi}$ and $\mathbf{q}_i^\mathsf{T}$ denotes the i^{th} row of $\mathbf{Q}$.
- Hyb_2: Same as Hyb_1, except the challenger uses the linear PCP simulator to construct the CRS and the proof. Specifically, instead of running $\mathcal{Q}_{\mathsf{LPCP}}$ to obtain $\mathsf{st}_{\mathsf{LPCP}}$ and $\mathbf{Q}$, the challenger instead samples $(\mathsf{st}_{\mathsf{LPCP}}, \mathbf{Q}, \mathsf{st}_{\mathsf{LPCP},\mathcal{S}}) \leftarrow \mathcal{S}_{\mathsf{LPCP},1}(1^\kappa)$. When constructing the proof, the challenger substitutes the simulated response $\mathbf{a} \leftarrow \mathcal{S}_{\mathsf{LPCP},2}(\mathsf{st}_{\mathsf{LPCP},\mathcal{S}}, \mathbf{x})$ in place of the value $\mathbf{a} = \mathbf{Q}^\mathsf{T}\boldsymbol{\pi}$ from Hyb_1.

Let $\mathsf{Hyb}_i(\mathcal{A})$ be the output of an execution of experiment Hyb_i. We now analyze the distribution of each pair of adjacent hybrid distributions as well as the distribution in Hyb_2:

- By design, the only difference between Hyb_0 and Hyb_1 is the challenger computes π using the simulator $\mathcal{S}_{\mathsf{Enc}}$ (with target value $\mathbf{a} = \mathbf{Q}^{\mathsf{T}}\boldsymbol{\pi}$) instead of using $\mathsf{Add}_{\mathsf{Enc}}(\mathsf{pp}, \{\mathsf{ct}_1, \dots, \mathsf{ct}_m\}, \{\pi_1, \dots, \pi_m\})$, where each ct_i is an encryption of $\mathbf{q}_i^{\mathsf{T}}$ (in which case $\sum_{i \in [m]} \pi_i \mathbf{q}_i^{\mathsf{T}} = \mathbf{Q}^{\mathsf{T}}\boldsymbol{\pi}$). If Π_{Enc} is ε-circuit private, then
$$|\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]| \le 2\varepsilon.$$
- The only difference between Hyb_1 and Hyb_2 is the challenger uses the linear PCP simulator $\mathcal{S}_{\mathsf{LPCP}}$to sample the queries $\mathbf{Q}$, the verification state st, and the responses $\mathbf{a}$ instead of using $\mathcal{Q}_{\mathsf{LPCP}}$ and $\mathcal{P}_{\mathsf{LPCP}}$. If Π_{LPCP} is perfect HVZK, then
$$\Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1] = \Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1].$$
- Finally, in Hyb_2, the behavior of the challenger is the same regardless of the bit b (i.e., the challenger computes crs, st, and π according to the specification of $\mathcal{S}_{\mathsf{SNARK}}$). This means that for all adversaries $\mathcal{A}$, $\Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1] = 1/2$, and the claim follows. □

C Linear PCP Implementation Details

In this section, we provide a more detailed description of our multipoint evaluation and interpolation approach outlined in Section 4.1. Following [BCG+13], for a polynomial $A(z)$ of degree less than $|D|$, we write $\mathrm{FFT}_D(A(z))$ to denote the vector of evaluations $(A(\alpha))_{\alpha \in D}$. Similarly, we write $\mathrm{FFT}_D^{-1}((A'(\alpha))_{\alpha \in D})$ to denote the coefficients of the polynomial A (of degree less than $|D|$) where $A(\alpha) = A'(\alpha)$ for all $\alpha \in D$.

Let $\omega \in \mathbb{F}$ be a primitive 2^d-th root of unity and $H = H_1 = \langle \omega \rangle \subset \mathbb{F}$ be the subgroup of order 2^d generated by ω (consisting of the 2^d-th roots of unity). Let $\xi_1 = 1$ and take $\xi_2, \dots, \xi_i \in \mathbb{F}^* \setminus H_1$ such that the cosets $H_i = \xi_i H_1$ are all pairwise disjoint. We define the domain to be $D = \bigcup_{i \in [k]} H_i$. For a set $S \subset \mathbb{F}$, let $\mathbf{V}_S \in \mathbb{F}^{|D| \times |D|}$ be the Vandermonde matrix associated with evaluating a polynomial of degree up to $|D| - 1$ on the points in D. Let $\hat{\mathbf{V}}_H \in \mathbb{F}^{2^d \times 2^d}$ be the Vandermonde matrix associated with evaluation a polynomial of degree up to 2^d on H (i.e., the roots of unity). Then, we have that

$$\mathbf{V}_S = \begin{bmatrix} \hat{\mathbf{V}}_H & \hat{\mathbf{V}}_H & \cdots & \hat{\mathbf{V}}_H \\ \hat{\mathbf{V}}_H \cdot \Xi_2 & \hat{\mathbf{V}}_H \cdot \xi_2^{2^d} \Xi_2 & \cdots & \hat{\mathbf{V}}_H \cdot \xi_2^{(k-1)2^d} \Xi_2 \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\mathbf{V}}_H \cdot \Xi_k & \hat{\mathbf{V}}_H \cdot \xi_k^{2^d} \Xi_k & \cdots & \hat{\mathbf{V}}_H \cdot \xi_k^{(k-1)2^d} \Xi_k \end{bmatrix},$$

where $\Xi_i = \mathrm{diag}(1, \xi_i, \xi_i^2, \dots, \xi_i^{2^d - 1})$. Take any input $\mathbf{a} \in \mathbb{F}^{k \cdot 2^d}$, and for $i \in [k]$, let $\hat{\mathbf{a}}_i = (a_{(i-1)2^d+1}, \dots, a_{i \cdot 2^d}) \in \mathbb{F}^{2^d}$. We describe an algorithm to compute $\mathbf{a}' = \mathbf{V}_S \mathbf{a}$:

- Let $\hat{\mathbf{a}}_i' = \left(a'_{(i-1)2^d+1}, \dots, a'_{i \cdot 2^d}\right)$. By construction,
$$\hat{\mathbf{a}}_i' = \hat{\mathbf{V}}_H \cdot \left(\sum_{j \in [k]} \xi_i^{(j-1)2^d} \Xi_i \hat{\mathbf{a}}_j \right).$$
Let $\hat{\mathbf{b}}_i = \sum_{j \in [k]} \xi_i^{(j-1)2^d} \Xi_i \hat{\mathbf{a}}_j \in \mathbb{F}^{2^d}$. Given $\hat{\mathbf{b}}_i$, computing $\hat{\mathbf{a}}_i = \hat{\mathbf{V}}_H \hat{\mathbf{b}}_i$ can be done using a standard radix-2 FFT in $O(d \cdot 2^d)$ time.
- Naïvely, we can compute $\hat{\mathbf{b}}_i$ in $O(k \cdot 2^d)$ time, so computing all of the entries in $\mathbf{b} = \left[\hat{\mathbf{b}}_1^{\mathsf{T}} \mid \cdots \mid \hat{\mathbf{b}}_k^{\mathsf{T}}\right]^{\mathsf{T}} \in \mathbb{F}^{k \cdot 2^d}$ requires $O(2^d k^2)$ time. However we can do so more efficiently as follows. By definition,
$$\hat{b}_{i,j} = \sum_{\ell \in [k]} \xi_i^{(\ell-1)2^d} \xi_i^{j-1} \hat{a}_{\ell,j}.$$

Now, define $\tilde{\mathbf{b}}_j = (\hat{b}_{1,j}, \ldots, \hat{b}_{k,j}) \in \mathbb{F}^k$, and similarly, let $\tilde{\mathbf{a}}_j = (\hat{a}_{1,j}, \ldots, \hat{a}_{k,j}) \in \mathbb{F}^k$. Then,

$$\tilde{\mathbf{b}}_j = \operatorname{diag}(\xi_1^{j-1}, \ldots, \xi_k^{j-1}) \underbrace{\begin{bmatrix} 1 & \xi_1^{2^d} & \cdots & \xi_1^{2^d(k-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \xi_k^{2^d} & \cdots & \xi_k^{2^d(k-1)} \end{bmatrix}}_{\Xi'} \tilde{\mathbf{a}}_j.$$

Observe now that $\Xi' \in \mathbb{F}^{k \times k}$ is itself a Vandermonde matrix corresponding to evaluating a degree $(k-1)$ polynomial on the points $\xi_1^{2^d}, \ldots, \xi_k^{2^d}$. While $\xi_1^{2^d}, \ldots, \xi_k^{2^d}$ are *not* roots of unity (so standard FFTs cannot be used here), we can still solve this problem efficiently if $\xi_1, \ldots, \xi_k$ form a geometric sequence (i.e., $\xi_i = \alpha\xi_{i-1}$ for some fixed $\alpha \in \mathbb{F}$) [BS05]. In particular, using the Bostan-Schost algorithms, multipoint evaluation on k values in a geometric sequence requires computing 2 degree-k polynomial multiplications and $O(k)$ additional work. In the case where $k < 2^{d-1}$, we can use standard radix-2 FFTs to implement the degree-k polynomial multiplications in $O(k \log k)$ time. Thus, computing each $\tilde{\mathbf{b}}_j$ can be done in just $O(k \log k)$ time. Repeating this for all $j \in [2^d]$ yields an algorithm to compute $\mathbf{b}$ in $O(2^d k \log k)$ time.

The overall running time of this algorithm is $O(2^d k(d + \log k))$, which matches the running time of a standard FFT over a domain of size $k \cdot 2^d$. While the concrete efficiency of the algorithm is worse than a standard radix-2 FFT, in fields where there are insufficient roots of unity (such as the ones we consider), this provides an efficient algorithm to implement the linear PCP prover. In all of our experiments, $k \le 64$.

In our implementation, we set $\xi_i = g^{2(i-1)\omega}$ for $i \in [k]$, where g is a multiplicative generator of $\mathbb{F}^*$. This enables efficient implementation of multipoint evaluation over the set D as well as the set $gD = \{gh \mid h \in D\}$ (needed for efficient implementation of the linear PCP prover algorithm; see [BCG+13] for further details).

Lagrange interpolation and inverse FFTs. In addition to computing FFT_D, the linear PCP prover needs to compute the inverse operation FFT_D^{-1}. This follows immediately from our algorithm above by inverting each of the steps (i.e., replace both sets of FFTs with their corresponding inverse FFTs).

The query-generation algorithm $\mathcal{Q}_{\mathsf{LPCP}}$ in Claim 2.6 (Construction A.1) essentially reduces to multiple Lagrange polynomial evaluations (with basis D) at a random field element. Ben-Sasson et al. [BCG+13] described an efficient implementation of this when the domain D is the roots of unity. In our setting (of working over a field with insufficient roots of unity), we augment D with cosets of the roots of unity. The Ben-Sasson et al. algorithm directly generalizes to this setting and we refer to [BCG+13, Appendix E] for the details.

D The Power Diffie-Hellman Assumption over Small Fields

In this section, we briefly recall the q-power Diffie-Hellman assumption introduced by Groth [Gro10] and subsequently used as the basis for both pairing-based SNARKs [GGPR13, PHGR13] as well as lattice-based SNARKs [GMNO18]. Following [GMNO18], we formulate the assumption with respect to a linear encoding scheme, which captures both the pairing-based instantiation as well as the lattice-based instantiation.

Definition D.1 (Linear Encoding Scheme)**.** A (secret-key) linear encoding scheme Π_{Enc} over a finite field $\mathbb{F}$ is a tuple of algorithms $\Pi_{\mathsf{Enc}} = (\mathsf{Setup}, \mathsf{Encode}, \mathsf{Add})$ with the following properties:

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: On input the security parameter λ, the setup algorithm outputs a public evaluation key pk and a secret encoding key sk.
- $\mathsf{Encode}(\mathsf{sk}, x) \to \mathsf{enc}_x$: On input the secret key sk and an element $x \in \mathbb{F}$, the encoding algorithm outputs an encoding enc_x of x.

- $\mathsf{Add}(\mathsf{pk}, (\mathsf{enc}_1, \ldots, \mathsf{enc}_d), (\alpha_1, \ldots, \alpha_d)) \to \mathsf{enc}'$: On input the public key pk, encodings $\mathsf{enc}_1, \ldots, \mathsf{enc}_d$ and coefficients $\alpha_1, \ldots, \alpha_d \in \mathbb{F}$, the add algorithm outputs a new encoding enc'.

The encoding scheme is *d-linear* if for all values $k \leq d$, values $x_1, \ldots, x_k \in \mathbb{F}$, scalars $\alpha_1, \ldots, \alpha_k \in \mathbb{F}^d$, and sampling $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{enc}_i \leftarrow \mathsf{Encode}(\mathsf{sk}, x_i)$ for all $i \in [k]$, we have that

$$\Pr[\mathsf{Add}(\mathsf{pk}, (\mathsf{enc}_1, \ldots, \mathsf{enc}_k), (\alpha_1, \ldots, \alpha_k)) \in S] = 1 - \mathsf{negl}(\lambda),$$

where S denotes the support of $\mathsf{Encode}(\mathsf{sk}, \sum_{i \in [k]} \alpha_i x_i)$.

Definition D.2 (q-Power Diffie-Hellman Assumption [Gro10, GMNO18])**.** Fix a parameter $q \in \mathbb{N}$. A linear encoding scheme $\Pi_{\mathsf{Enc}} = (\mathsf{Setup}, \mathsf{Encode}, \mathsf{Add})$ over a field $\mathbb{F}$ satisfies the q-power Diffie-Hellman assumption (q-PDH) if for all efficient adversaries $\mathcal{A}$, and sampling $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $s \xleftarrow{\text{R}} \mathbb{F}$, $\mathsf{enc}_i \leftarrow \mathsf{Encode}(\mathsf{sk}, s^i)$ for all $i \in \{0, \ldots, 2q\}$, $\sigma \leftarrow (\mathsf{pk}, \mathsf{enc}_0, \ldots, \mathsf{enc}_q, \mathsf{enc}_{q+2}, \ldots, \mathsf{enc}_{2q})$, we have that

$$\Pr[\mathcal{A}(1^\lambda, \sigma) \in S] = \mathsf{negl}(\lambda),$$

where S is the set of encodings in the support of $\mathsf{Encode}(\mathsf{sk}, s^{q+1})$.

Lemma D.3 (q-PDH Assumption over Small $\mathbb{F}$)**.** *Let $\Pi_{\mathsf{Enc}} = (\mathsf{Setup}, \mathsf{Encode}, \mathsf{Add})$ be a d-linear encoding scheme over a finite field $\mathbb{F}$. If $d \geq 2q$, there exists an adversary that runs in time $\mathsf{poly}(q, \log |\mathbb{F}|)$ and wins the q-PDH security game for Π_{Enc} with advantage $2q/|\mathbb{F}|$.*

Proof. The adversary $\mathcal{A}$ starts by choosing $2q$ distinct points $z_1, \ldots, z_{2q} \in \mathbb{F}$, and forms the polynomial $f(x) = \prod_{i \in [2q]} (x - z_i)$. Write this as $f(x) = \sum_{i=0}^{2q} \alpha_i x^i$. Then, for all $i \in [2q]$, $z_i^{q+1} = -\alpha_{q+1}^{-1} \sum_{j \neq q+1} \alpha_j z_i^j$. Let $(\mathsf{pk}, \mathsf{enc}_0, \ldots, \mathsf{enc}_q, \mathsf{enc}_{q+2}, \ldots, \mathsf{enc}_{2q})$ be the q-PDH challenge. Here, enc_i is an encoding of s^i, where $s \in \mathbb{F}$ is sampled by the q-PDH challenger at the beginning of the experiment. Since $d \geq 2q$, the adversary can homomorphically compute an encoding of $-\alpha_{q+1}^{-1} \sum_{i \neq q+1} \alpha_i s^i$. By the above analysis, if $s \in \{z_1, \ldots, z_{2q}\}$, then this quantity is exactly s^{q+1}. Since s is uniform and independent of $z_1, \ldots, z_{2q}$, the probability that $s \in \{z_1, \ldots, z_{2q}\}$ is exactly $2q/|\mathbb{F}|$, which proves the claim. □

Remark D.4 (q-Power Diffie-Hellman Assumption over Small $\mathbb{F}$)**.** When the q-PDH assumption is used for constructing pairing-based zkSNARKs [Gro10, PHGR13, GGPR13], the size of the underlying field $\mathbb{F}$ is super-polynomial (i.e., $|\mathbb{F}| = 2^{\Omega(\lambda)}$). In this case, the attack in Lemma D.3 has negligible advantage. Indeed, the q-PDH assumption plausibly holds over standard pairing-based groups, and holds unconditionally in the generic (bilinear) group model [Gro10].

In the lattice-based zkSNARK of Gennaro et al. [GMNO18], they consider fields of polynomial size. Unfortunately, Lemma D.3 shows that the q-PDH assumption does not hold for encoding schemes over fields of polynomial size. For the specific instantiation proposed by Gennaro et al., $q \approx 2^{16}$ and $|\mathbb{F}| \approx 2^{32}$, so Lemma D.3 gives an attack on q-PDH with advantage $2q/|\mathbb{F}| = 2^{-15}$. Since their zkSNARK relies on hardness of the q-PDH assumption for soundness, this means that their suggested parameters provide at best 15 bits of provable soundness. To obtain 128-bits of soundness, it would be necessary to either apply soundness amplification (which increases all parameters by a factor of $128/15 \approx 8.5$) or instantiate the Regev-based encoding scheme over a super-polynomial size field (which would also incur additional overhead).

In this work, we work over small (polynomial-size) fields and use parallel repetition (at the linear PCP level) for soundness amplification (see Remark 2.7). This increases the number of linear PCP queries, but since we encrypt *vectors* of queries, the overhead for parallel amplification is *additive* rather than multiplicative in the number of repetitions. This yields a much more efficient construction over *small* fields compared to the Gennaro et al. construction (see Table 1).