SHORTER AND FASTER POST-QUANTUM DESIGNATED-VERIFIER ZKSNARKS FROM LATTICES

A THESIS

PRESENTED TO
THE FACULTY OF THE SCHOOL OF ENGINEERING AND APPLIED SCIENCE
UNIVERSITY OF VIRGINIA

IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE
MASTER OF SCIENCE

Hang Su

April 2021

APPROVAL SHEET

This thesis is submitted in partial fulfillment of the requirements for the degree of Masters of Science

Hang Su

This thesis has been read and approved by the Examining Committee:

(David J. Wu) Principal Advisor

(David Evans)

(Mohammad Mahmoody)

Accepted for the School of Engineering and Applied Science:

Craig H. Benson, Dean, School of Engineering and Applied Science

April 2021

Abstract

Zero-knowledge succinct argument of knowledge (zkSNARK), a *non-interactive* zero-knowledge proof of knowledge protocol, enables efficient privacy-preserving proofs of memberships for general NP languages. Nowadays, zkSNARKs have become important build blocks in numerous applications, and a growing number of works have focused on optimizing different properties of the proof system.

This work is focusing on minimizing concrete proof size for *post-quantum* zkSNARKs. At a high-level, our construction follows the blueprint of Bitansky et al. [BCI+13] and Boneh et al. [BISW17] of combining a linear probabilistically checkable proof (linear PCP) together with a linear-only vector encryption scheme. We developed a concretely-efficient *lattice-based* instantiation by considering quadratic extension fields $\mathbb{F}_{p^2}$ with moderate characteristic and using linear-only vector encryption over rank-2 module lattice. To our knowledge, it is plausibly the first system that uses *linear PCPs* over extension fields. Moreover, applying linear-only encryption over *extension* fields reduces the lattice parameters, and thus improves the concrete efficiency of our lattice-based zkSNARK significantly.

Before our work, 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. After an initial preprocessing step, verifying an NP relation of size 2^{20} requires a 16 KB proof, 68 s of prover time, and 1.2 ms of verifier time. Our proofs are 10.3× shorter than existing post-quantum candidates. Compared to previous lattice-based zkSNARKs (also in the designated-verifier preprocessing model), we obtain a 39.4× reduction in proof size and a 60.2× reduction in prover time, all while achieving a much higher level of soundness.

To my family, my teachers, and my friends.

青青陵上柏，磊磊涧中石。
人生天地间，忽如远行客。

—《青青陵上柏》

街灯的光穿窗而入，屋子里显出微明。
我大略一看，熟识的墙壁，壁端的棱线，熟识的书堆，堆边的未订的画集，
外面的进行着的夜，无穷的远方，无数的人们，都和我有关。
我存在着，我在生活，我将生活下去。

—鲁迅《这也是生活》

当人们无法选择自己的未来时，就会珍惜自己选择过去的权利。

时间呈现为透明的灰暗，所有一切都包孕在这隐藏的灰暗之中。
我们并不是生活在土地上，事实上我们生活在时间里。
田野、街道、河流、房屋是我们置身时间之中的伙伴。
时间将我们推移向前或者向后，并且改变着我们的模样。

我是那样崇拜生命在我体内流淌的声音。
除了生命本身，我再也找不出活下去的另外理由了。

我们都是忘记了对方的模样以后，在路上相遇。

—余华《在细雨中呼喊》

I like the peace in the backseat
I don't have to drive, I don't have to speak
I can watch the countryside, and I can fall asleep

My family tree's losing all it's leaves
Crashing towards the driver's seat
The lightning bolt made enough heat
To melt the street beneath your feet

I've been learning to drive, my whole life

— Arcade Fire, In the Backseat

Acknowledgements

I would like to begin by thanking my advisor, David Wu, for his advice and mentorship throughout the past two years. This thesis would not have been possible without his guidance. Two years ago, when I was young and ignorant, I had the good fortune of meeting him and then taking his Introduction to Cryptography class. David drew me into this field with his great passion towards both cryptography and mentorship. I can not be more grateful for David's kindness and patience, and I am always impressed by his helpful and insightful advice. Working and researching with David has been (somewhat) annoying and (extremely) fruitful. David is always a few steps ahead, pushing me to think clearly and write precisely. It might be arrogant, but I still want to say: David, you are a standard in research that I hope I can reach some day in the future.

Before I entered UVA, I was fortunate to know and work with many incredibly talented friends in my undergraduate years. I would like to especially thank Yue "Tripack" Yao and Yifan "Evan" Zhao. Yue, you were the one who led me on the road to programming and computer science. Without you, I would not be who I am today. I really miss the hours we spent talking about computer science research and all the insightful ideas you provided. Yifan, I am very fortunate to have had you as a friend for these years. It is always fun to work with you on random programming language related projects. Our discussions on research, gaming, and everything else were a highlight of my undergraduate years. You two are both amazing friends, and I look forward to seeing what you accomplish in your research career.

Finally, I want to thank my family for all their unconditional love and support.

2020 was a year of great challenge, "Une Année Sans Lumière". I can not be more grateful for my uncle and aunt taking me in after learning of my father's serious illness. Most of this work, including coding, was finished in their place. They are the toughest people I have ever met. They have all been greatly supportive and have carried me through the darkest hours.

My cousins, Jessica and Sarah Lao, have always been uplifting, talented, and unfailingly loyal. Both of you got into Harvard, and I can never be more proud of that. You guys have set a level of excellence that I will always look up to. Jessica, thank you for the recipe book and for making sure I always eat on schedule. Your (too-sugary) desserts pair so well with my triple-strong coffee. Sarah, thank you for sitting beside me, going through my writing (including this acknowledgement) to ensure everything is grammatically correct, and tolerating my lame pink guy jokes. It is calming to talk with you during these dark days.

It would be impossible to adequately appreciate my parents, Lihong Pan and Zhilong Su, for the role they played in my life. Mom, thank you for tolerating my stubbornness, laziness, and arrogance during my youth, spending countless hours on my education. Dad, thank you for believing in me, encouraging me to make the most out of my talent, and reminding me to be on schedule. I know it has always been your long-time dream to study in the US, but due to a tight budget, it did not work out. Now I am stepping in your old boots to realize your dream. I hope you are proud of me. I don't know how best to commemorate your life. I guess the only way is for me, as your legacy, to make the most out of my own life, thus extending yours. We all miss you so much. This thesis is for you.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

On a high level, a proof system for a language $\mathcal{L} \subseteq \{0,1\}^*$ is a two-party protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. The goal of the prover is to convince the verifier of some statement $\mathbf{x} \in \{0,1\}^*$ is a member of the language $\mathcal{L}$ (namely $\mathbf{x} \in \mathcal{L}$). This give rise to two basic properties: completeness, which roughly indicates an honest prover will be able to convince an honest verifier of any true statement $\mathbf{x} \in \mathcal{L}$, and soundness, which roughly says a plausibly honest prover will not be able to prove a fales statement $\mathbf{x} \notin \mathcal{L}$ to an honest verifier.

In the case of the NP languages, we notice that the NP class of languages can be efficiently checked in polynomial time. Namely, if a language $\mathcal{L} \subseteq \{0,1\}^*$ is in NP, there exists a polynomial time algorithm $\mathcal{R}$ (referred as an NP relation) such that for every $\mathbf{x} \in \{0,1\}^*$,

$$\mathbf{x} \in \mathcal{L} \Longleftrightarrow \exists \mathbf{w} \in \{0,1\}^{\mathsf{poly}(|\mathbf{x}|)} : \mathcal{R}(\mathbf{x}, \mathbf{w}) = 1.$$

In this way, all NP languages have an efficient *non-interactive* proof system: to prove a statement $\mathbf{x} \in \{0,1\}^*$ is in $\mathcal{L}$, the prover sends the witness the NP witness $\mathbf{w}$, and the verifier computes if $\mathcal{R}(\mathbf{x}, \mathbf{w}) = 1$.

For an NP relation $\mathcal{R}$, a zero-knowledge proof of knowledge [GMR85] enables a prover to convince a verifier of $\mathbf{x} \in \mathcal{L}$ without revealing anything about the NP witness $\mathbf{w}$. In this work, we are focusing on zero-knowledge succinct argument of knowledge (zkSNARK) [Kil92, Mic00, GW11]. In a zkSNARK, we additionally require the proof containing only a single message π from prover to verifier, while the length of π and the verification complexity are 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]).

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]. They 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 sampling 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). 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 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 applies the linear PCP verification. 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).

Post-quantum zkSNARKs. Many existing constructions of zkSNARKs rely on group-based and pairing-based assumptions [Gro10, PHGR13, GGPR13, BCI+13, Gro16, BCC+16, BBB+18, MBKM19, CHM+20, Set20, SL20] with only pre-quantum security. 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 the pre-quantum ones, current post-quantum constructions have substantially longer proofs: proofs in the most succinct pre-quantum construction (Groth [Gro16]) are just 128 bytes, while those in the most succinct post-quantum constructions [BCR+19, COS20, BFH+20] are over 1000× longer (i.e., at least 130 KB for verifying circuits of size 2^{20}). The increase in parameter sizes is caused by the gap 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].

Lattice-based instantiations of Bitansky et al. Gennaro et al. [GMNO18] and 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/p, 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 instantiation is significantly less efficient. For an instance of size 2^{16}, the proof size is 640 KB, over $5000\times$ larger than the pairing-based construction of Groth [Gro16]; similarly, the prover time for a similar-sized instance is roughly $40\times$ slower than the pairing-based analog. In fact, as we discuss in Appendix A, 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 $q > p^2 N$. Zero-knowledge adds a further multiplicative factor of 2^κ where κ is a statistical security parameter.

For 128 bits of soundness then, we need $p > 2^{128} N$. 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 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 soundness (see Appendix A and Remark A.4). But even with this limited level of security, 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

Two things primarily enable our lattice-based zkSNARKto be concretely-efficient: (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]. 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 encrypts rows of $\mathbf{Q}$. The proof then consist of a single ciphertext encrypting the vector of linear PCP responses. The advantage of this approach is that we can reduce the cost of ciphertext expansion for lattice-based vector encryption by amortization. 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 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:** Using vector encryption based on the construction of Peikert et al., we can encrypt a vector of plaintext values in a *single* lattice ciphertext. We note though that for an encryption scheme to plausibly satisfy the necessary "linear-only" property, the ciphertext space must be sparse, or an adversary can obliviously sample a valid ciphertext *without* the knowledge of the corresponding plaintext. 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). Directly applying "double encryption" would incur a doubled ciphertext size. In Section 3.3, we describe an alternative route where we increase the vector dimension (essentially by concatenating a tag to each message) that achieves the same effect. The advantage of our approach is that it incurs a small *additive* overhead on ciphertext size or proof size rather than a multiplicative one.

- **Modulus switching:** We can decrease the proof size of the zkSNARK using modulus switching technique in [BGV12]. The slack between the modulus q and lower bound of the modulus allows us to apply this trick. With modulus switching, after prover homomorphically computes its response ciphertext (a vector of a large ring elements), the prover rescales each component of the ciphertext to a much smaller modulus. The actual decryption is then applied over the rescaled ciphertext. In our construction, modulus switching decreases the modulus q by more than a factor of 2, which translate to a same factor of decrease in concrete proof size. We describe this in Theorem 3.21.

Instantiating our vector encryption scheme over $\mathbb{F}_p$ using a 23-bit p yields a zkSNARK where the proof size is 27 KB (and a 19 GB CRS); with a 28-bit prime, the proof size increases to 29 KB with a CRS of size 5.5 GB (for R1CS instances with 2^{20} constraints). This is already an improvement over previous post-quantum zkSNARKs, but comes at the cost of having a large CRS.

Extension fields of moderate characteristic. Considering linear PCPs over extension fields *of moderate characteristic* is a way to reduce the lattice parameters. 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 the linear PCPs based on QAPs is agnostic to the choice of the field, and works 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). 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 the efficiently-computable isomorphism between $\mathbb{F}_{p^k}$-operations and 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, where $k = 2$ (see Remark 3.19). We obtain a construction from one instantiation of this approach 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 16 KB (over 40% reduction from working over a prime field).

- **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. 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 $R_p = R/pR$, and $R_p \cong \mathbb{F}_{p^2}$ when $p = 3 \bmod 4$. This gives a direct compilation from a linear PCP over a quadratic extension $\mathbb{F}_{p^2}$ to a zkSNARK over $\mathbb{F}_{p^2}$, under the conjecture that the vector encryption scheme is linear-only over R_p. 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 $2\times$ 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 1.1 and in Section 4.3. Leveraging encryption

[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 less parallel repetitions, and *reduces* the overall size of the CRS.

schemes over extension fields and higher-rank modules has also been useful for improving the asymptotic and concrete efficiency of other lattice-based constructions [GHS12a, GHS12b, Gen09].

Parameter selection. We consider quadratic extension fields with two different characteristics in this work: (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 small (e.g., $p = 2^{13} - 1$), the extension field does not contain a sufficiently-large subgroup of roots of unity to directly apply 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.

Another consideration that underlies our choise of prime modulus in this work is the magnitude of the lattice modulus q needed to instantiate the lattice-based lattice-based encryption scheme over $\mathbb{F}_p$ or $\mathbb{F}_{p^2}$. When $q < 2^{128}$, we can use compiler intrinsic types to implement the 128-bit arithmetic, which 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 4.2.

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. Due to the noise accumulation through homomorphic operations, the lattice-based schemes are not directly re-randomizable. In this work, we show that a weaker notion of circuit privacy [Gen09] (i.e., the result of computing a linear combination of ciphertexts hide the coefficients of the linear combination) suffices to argue zero-knowledge for the SNARK. 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 describe the details in Section 3.3. Moreover, we note in Remark 3.27 that even without circuit privacy, a direct compilation can still *heuristically* provide zero-knowledge.

Implementation and evaluation. In Chapter 4, we describe our implementation of our lattice-based zkSNARK. We provide a comprehensive evaluation of the different trade-offs in parameter sizes and compute time 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 Table 1.2 and Section 1.4.

1.3 Contributions

Our main contribution is a new designated-verifier zkSNARK from lattice-based assumptions where the proof size (for verifying an R1CS instance of size 2^{20}) is under 17 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 39.4× shorter. However, there remains a large gap (128×) compared to the shortest *pre-quantum* zkSNARK by Groth [Gro16]. We refer to Table 1.1 for the full comparison and describe our experimental setup in detail in Section 4.3.

The prover and verifier complexities of our new zkSNARK also compare favorably against other post-quantum schemes. Our construction is 4.6–6.3× faster for the prover compared to Aurora and Fractal on R1CS instances of similar size. Compared to the Gennaro et al. [GMNO18] lattice-based candidate, our construction is 60.2× faster for the prover and 5.1× faster for the verifier. Moreover, our main constructions (with *provable* zero-knowledge) are 1.1–1.6× *faster* for the prover and about 2.7–7.9× faster for the verifier compared to pre-quantum pairing-based construction of Groth [Gro16]. In fact, the prover in one of our instantiations (see Section 4.3) is 2.3× *faster* than the construction of Groth. The caveat is that this instantiation does not provide *provable* zero-knowledge, but may still do so in a heuristic sense (see Remark 3.27). This makes it more suitable for use in verifiable computation scenarios that do not require zero-knowledge.

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. Compared to Ligero [AHIV17], our prover is about 1.8× more expensive, but our proof sizes and verification times are over 874× better (see Table 1.1). In general, these alternative approaches typically enjoy lower prover costs, but either have longer proofs or higher verification costs for verifying general, *unstructured* computations. We provide more details and comparisons with other zkSNARKs in Section 1.4.

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 using a different base 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 1.1).

Scheme	**Structure**	**PQ**	**TP**	**PV**	**R1CS Size**	**Size**		**Time**		
						CRS	**Proof**	**Setup**	**Prover**	**Verifier**
[Gro16]	Pairings	○	○	●	2^{16}	12.4 MB	128 B	5.6 s	5.5 s	3.3 ms
					2^{20}	199 MB	128 B	72.0 s	78.6 s	3.4 ms
[GMNO18] *	Lattices	●	○	○	2^{16}	17.3 MB†	640 KB	167 s	235 s	3.46 ms
Ligero [AHIV17]	Random Oracle	●	●	●	2^{16}	—	4.3 MB	—	2.5 s	1.3 s
					2^{20}	—	14 MB	—	37.6 s	21.6 s
Aurora [BCR+19]	Random Oracle	●	●	●	2^{16}	—	121 KB	—	17.8 s	380 ms
					2^{20}	—	169 KB	—	304 s	6.3 s
Fractal [COS20]‡	Random Oracle	●	●	●	2^{16}	1.4 GB	178 KB	11.6 s	21.3 s	8.3 ms
					2^{19}	11 GB	215 KB	115.8 s	184.3 s	9.5 ms
This work (Shorter Proofs)	Lattices	●	○	○	2^{16}	191 MB	15.2 KB	88.3 s	3.9 s	0.69 ms
					2^{20}	5.3 GB	16.4 KB	2240 s	68 s	1.23 ms
This work (Shorter CRS)	Lattices	●	○	○	2^{16}	104 MB	19.9 KB	53.0 s	3.4 s	0.37 ms
					2^{20}	1.9 GB	20.8 KB	877 s	56 s	0.43 ms

*As we discuss in Appendix A (Remark A.4), the parameter instantiation proposed in Gennaro et al. [GMNO18] only provides 15 bits of 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 9×. 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 1.1: 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" setting operates over the field $\mathbb{F}_{p^2}$ where $p = 2^{13} - 1$) and the "Shorter CRS" setting operates over the field $\mathbb{F}_{p^2}$ where $p = 2^{19} - 1$. 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 (○).

1.4 Related Work

There has recently been a flurry of works studying the asymptotic and concrete efficiency of succinct argument systems. We survey several families of constructions here and also include a comparison with several representative schemes in Table 1.2. 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$^+$13] described general frameworks for constructing *constant-size* zkSNARKs from linear PCPs (specifically, QAPs). Several works have built upon and extended these frameworks [DFGK14, Gro16, BISW17, GMNO18, BISW18, BIOW20]. These constructions are the basis of numerous systems and implementations [PHGR13, BCG$^+$13, BFR$^+$13, BCTV14b, BCTV14a, FGP14, WSR$^+$15, BBFR15, CTV15, DFKP16, FFG$^+$16, BCG$^+$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) [BSCS16, 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.

1.5 Works Contained in this Thesis

The results and description in this thesis are based on the material that originally appeared in the following publication:

- *Shorter and Faster Post-Quantum Designated-Verifier zkSNARKs from Lattices* with Yuval Ishai, and David J. Wu (ACM CCS, 2021) [ISW21].

	PQ	**TP**	**PV**	**Proof Size**		**Runtime**		**Crypto.**
				Asymptotic	**Concrete**	**Prover**	**Verifier**	**Structure**
Groth [Gro16]	○	○	●	1	128 B	$N \log N$	$\|x\|$	Pairings
Marlin [CHM$^+$20]	○	●	●	1	704 B	$N \log N$	$\|x\| + \log N$	Pairings
Sonic [MBKM19]	○	◐	●	1	1.1 KB	$N \log N$	$\|x\| + \log N$	Pairings
Xiphos [SL20]	○	●	●	$\log N$	61 KB	N	$\|x\| + \log N$	Pairings
Spartan [Set20]	○	●	●	$\sqrt{N}$	142 KB	N	$\|x\| + \sqrt{N}$	Groups
Fractal [COS20]	●	●	●	$\log^2 N$	215 KB†	$N \log N$	$\|x\| + \log^2 N$	RO
[GMNO18]*	●	○	○	1	640 KB‡	$N \log N$	$\|x\|$	Lattices
STARK [BBHR18b]	●	●	●	$\log^2 N$	3.2 MB	$N \mathsf{polylog}(N)$	$\|x\| + \log^2 N$	RO
This work*	●	○	○	1	16 KB	$N \log N$	$\|x\|$	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` [Lab21b] 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 1.2: Comparison with recent zkSNARKs for verifying an NP statement of size N constraints 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.

Chapter 2

Preliminaries

We begin by introducing the basic notation that we use throughout this thesis. For a positive integer $n \in \mathbb{N}$, we write $[n]$ to denote the set of integers $\{1, \ldots, n\}$. We write $\{x_i\}_{i \in [n]}$ to denote the ordered multi-set of variables $x_1, \ldots, x_n$. For a finite set $\mathcal{S}$, we write $x \overset{\text{R}}{\leftarrow} \mathcal{S}$ to denote that x is sampled uniformly at random from $\mathcal{S}$. For a distribution $\mathcal{D}$, we write $x \leftarrow \mathcal{D}$ to denote that x is sampled from $\mathcal{D}$.

Throughout this work, we use λ to denote a computational security parameter and κ to denote a statistical security parameter. We say a function f is negligible if $f(\lambda) = o(1/\lambda^c)$ for all $c \in \mathbb{N}$; we denote this $f(\lambda) = \mathsf{negl}(\lambda)$. We use $\mathsf{poly}(\lambda)$ to denote a quantity that is upper-bounded by a fixed polynomial on input λ.

We say an event happens with negligible probability if the probability that the event occurs is negligible, and an event happens with overwhelming probability if the probability of its complement occurs is negligible. We say a function $\mathcal{A}$ is efficient if it runs in probabilistic polynomial time at the length of the input. We denote $\mathcal{A}(x; r)$ the output of $\mathcal{A}$ on the input of x and randomness r. In typical cases where we do not specify the randomness explicitly, we use $\mathcal{A}(x)$ to denote the output distribution of $\mathcal{A}$ on the input of x, where the randomness is drawn from the uniform distribution.

We say 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 $\mathcal{D}_1$ and $\mathcal{D}_2$ are statistically indistinguishable if the statistical distance between $\mathcal{D}_1$ and $\mathcal{D}_2$ are upper-bounded by a negligible function $\mathsf{negl}(\kappa)$.

Vectors and Matrices. We will typically use bold lowercase letters (e.g., $\mathbf{v}, \mathbf{c}$) to denote vectors and use bold uppercase letters (e.g., $\mathbf{S}, \mathbf{T}$) to denote matrices. For a vector $\mathbf{v} \in \mathbb{Z}_p^n$, we use non-boldface letters to refer to its components, namely we write $\mathbf{v} = (v_1, \ldots, v_n)$. For a vector $\mathbf{v} \in \mathbb{R}^n$, we write $\|\mathbf{v}\|_p$ to denote the ℓ_p norm of $\mathbf{v}$. For a matrix $\mathbf{A} \in \mathbb{Z}_p^{h \times w}$, we write $\mathbf{a}_i$ where $i \in [h]$ to denote the i^{th} row of matrix $\mathbf{A}$. For $i \in [h], j \in [w]$, we use $a_{i,j}$ to denote the entry in i^{th} row and j^{th} column.

Field and Field Extensions. We will typically use $\mathbb{F}$ to denote finite fields. We write $\mathbb{F}_p$ to denote a finite field of order p, and we write the degree-d field extension of $\mathbb{F}_p$ as $\mathbb{F}_{p^d}$. Recall that $\mathbb{F}_{p^d}$ 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 $\mathbb{F}_p^d$, and there is an efficiently-computable isomorphism between $s \in \mathbb{F}_{p^d}$ and $\mathbf{v}_s \in \mathbb{F}_p^d$. For all $s, t \in \mathbb{F}_{p^d}$, $\mathbf{v}_s + \mathbf{v}_t = \mathbf{v}_{s+t} \in \mathbb{F}_p^d$. Moreover, we refer $\mathbf{M}_s \in \mathbb{F}_p^{d\times d}$ to the linear transformation over $\mathbb{F}_p^d$ by s: $\forall s, t \in \mathbb{F}_{p^d}$, $\mathbf{M}_s\mathbf{v}_t = \mathbf{v}_{st}$.

Polynomial Ring. We will typically use R to denote rings. In this work, we work over $R = \mathbb{Z}[x]/(x^d + 1)$, where d is a power of 2. We specifically consider the case 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 present an element $r \in R$ as an array of coefficients $\mathbf{v}_r \in \mathbb{Z}^d$. For an element $r \in R$, we denote $\|r\|_\infty$ to be the ℓ_∞-norm of the vector of coefficient corresponding to r. For a vector $\mathbf{r} \in R^n$, we write $\|\mathbf{r}\|_p$ to denote the ℓ_p-norm of the vector of coefficient $\mathbf{v}_\mathbf{r} \in \mathbb{Z}^{dn}$ formed by concatenating $\mathbf{v}_{r_i} \in \mathbb{Z}^d$ for each $i \in [n]$. We write γ_R to denote the expansion constant where any $r, s \in R$, we have $\|rs\|_\infty \leq \gamma_R \|r\|_\infty \|s\|_\infty$. For $d = 1$, $\gamma_R = 1$ and for $d = 2$, $\gamma_R = 2$.

Discrete Gaussian and Tail Bounds. We recall some preliminaries on the Discrete Gaussian distribution. For a real number $s > 0$, the Gaussian function $\rho_s : \mathbb{R} \to \mathbb{R}^+$ with width s is $\rho_s(x) := \exp(-\pi x^2/s^2)$. The discrete Gaussian function $D_{\mathbb{Z},s}$ over $\mathbb{Z}$ with mean 0 and width s is the distribution that

$$\Pr[X = x : X \leftarrow D_{\mathbb{Z},s}] = \frac{\rho_s(x)}{\sum_{i\in\mathbb{Z}} \rho_s(i)}. \tag{2.0.1}$$

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 also subgaussian with parameter $\|\mathbf{s}\|_2$ where $\mathbf{s} = (s_1, \ldots, s_m)$.

Smudging Lemma. We recall the smudging lemma, which "smudge out" any small values with large noise.

Lemma 2.1 (Smudging Lemma [AJLA+12, MW16])**.** *Let B, B' be integers. Fix any value $e_1 \leq |B|$ and sample $e_2 \overset{\mathrm{R}}{\leftarrow} [-B', B']$. The statistical distance between the distributions of $e_1 + e_2$ and e_2 is at most B'/B.*

Schwartz-Zippel Lemma. We recall the Schwartz-Zippel Lemma [Zip79, Sch80], which we use throughout the thesis.

Lemma 2.2 (Schwartz-Zippel Lemma [Zip79, Sch80])**.** *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a multivariate polynomial of total degree d, not identically zero. Then for any subset $\mathcal{S} \subset \mathbb{F}$,*

$$\Pr[f(\alpha_1, \ldots, \alpha_n) = 0 \mid \alpha_1, \ldots, \alpha_n \overset{\mathrm{R}}{\leftarrow} \mathcal{S}] \leq \frac{d}{|\mathcal{S}|}.$$

2.1 Circuit Satisfiability Problems

The circuits that we discuss in this thesis are not boolean but *arithmetic*. An arithmetic circuit defined over a finite field $\mathbb{F}$ takes elements in $\mathbb{F}$ as inputs, outputs elements in $\mathbb{F}$, and contains only *bilinear gates*.

Rank-1 Constraint Satisfiability. We recall the definition of the NP-complete language rank-1 constraint satisfiability (R1CS) that was introduced implicitly by Gennaro et al. [GGPR13] and formalized more explicitly in [SBV$^+$13, BCG$^+$13, BCR$^+$19].

Definition 2.3 (Rank-1 Constraint Satisfiability [SBV$^+$13, BCG$^+$13, GGPR13, BCR$^+$19])**.** A rank-1 constraint satisfiability (R1CS) system over a finite field $\mathbb{F}$ is specified by a tuple $\mathcal{S} = (n, N_g, N_w, \{\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i\}_{i\in[N_g]})$ 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 $\mathcal{S}$ 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 $\mathcal{S}(\mathbf{x}, \mathbf{w}) = 1$ 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 $\mathcal{S}$, we define the corresponding relation $\mathcal{R}_\mathcal{S} = \{(\mathbf{x}, \mathbf{w}) \in \mathbb{F}^n \times \mathbb{F}^{N_w} : \mathcal{S}(\mathbf{x}, \mathbf{w}) = 1\}$.

Remark 2.4 (Boolean and Arithmetic Circuit Satisfiability)**.** As is shown in [BCI$^+$13, GGPR13, BCTV14b], the language R1CS capture Boolean and arithmetic circuit satisfiability as special cases. Let n, h, l denote the input, witness and output size respectively. A Boolean circuit satisfiability instance for a Boolean circuit $C : \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ with α wires and β bilinear gates yields an R1CS instance with $N_w = \alpha$ and $N_g = \beta + h + 1$ [BCI$^+$13]. An arithmetic circuit satisfiability instance for an arithmetic circuit $C : \mathbb{F}^n \times \mathbb{F}^h \to \mathbb{F}^\ell$, where ℓ denotes the output size, with α wires and β bilinear gates yields an R1CS instance with $N_w = \alpha$ and $N_g = \beta + \ell$ [BCTV14b].

2.2 Succinct Non-Interactive Arguments

We review the formal definition of zkSNARKs for R1CS.

Definition 2.5 (Succinct Non-Interactive Argument of Knowledge [BCI$^+$13, adapted])**.** Let $\mathcal{S} = (n, N_g, N_w, \{\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i\}_{i\in[N_g]})$ be a family of R1CS over a finite field $\mathbb{F}$, let $\mathcal{R}_\mathcal{S}$ be the associated relation, and let $\mathcal{L}_\mathcal{S}$ be the associated language. A succinct non-interactive argument of knowledge (SNARK) in preprocessing model [1] for $\mathcal{R}_\mathcal{S}$ is a tuple $\Pi_{\mathsf{SNARK}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ with following properties:

[1]In the preprocessing model, we allow for a *statement-independent* setup algorithm that runs in polynomial time in the size of $\mathcal{S}$. In "fully-succinct" SNARK, the setup and prover are required to run in sublinear (or polylogarithmic) in the size of $\mathcal{S}$ [BCI$^+$13]. Using recursive composition [BCCT13], it is possible to obtain fully-succinct SNARKs from preprocessing SNARKs.

- $(\mathsf{crs}, \mathsf{st}) \leftarrow \mathsf{Setup}(1^\lambda)$: On input the security parameter λ, the setup algorithm outputs a common reference string crs and verification state st.
- $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathbf{x}, \mathbf{w})$: On input a common reference string crs, a statement $\mathbf{x}$, and a witness $\mathbf{w}$, the prove algorithm outputs a proof π.
- $b \leftarrow \mathsf{Verify}(\mathsf{st}, \mathbf{x}, \pi)$: 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 all the following properties:

- **Completeness:** For all $\lambda \in \mathbb{N}$ and all $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}_{\mathcal{S}}$,

$$\Pr[\mathsf{Verify}(\mathsf{st}, \mathbf{x}, \mathbf{w}) = 1] = 1,$$

  where $(\mathsf{crs}, \mathsf{st}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathbf{x}, \mathbf{w})$.

- **Knowledge:** For all polynomial-sized prover $\mathcal{P}^*$, there exists a polynomial-sized extractor $\mathcal{E}$ such that for all security parameters $\lambda \in \mathbb{N}$ and for all auxiliary inputs $z \in \{0, 1\}^{\mathsf{poly}(\lambda)}$,

$$\Pr[\mathsf{Verify}(\mathsf{st}, \mathbf{x}, \pi) = 1 \wedge (\mathbf{x}, \mathbf{w}) \notin \mathcal{R}_{\mathcal{S}}] = \mathsf{negl}(\lambda),$$

  where $(\mathsf{crs}, \mathsf{st}) \leftarrow \mathsf{Setup}(1^\lambda), (\mathbf{x}, \pi) \leftarrow \mathcal{P}^*(1^\lambda, \mathsf{crs}; z), \mathbf{w} \leftarrow \mathcal{E}(1^\lambda, \mathsf{crs}, \mathsf{st}, \mathbf{x}; z)$.

- **Efficiency:** There exists an universal polynomial p (independent of $\mathcal{S}$) such that the Setup and Prove run in time $p(\lambda + |\mathcal{S}|)$, Verify runs in time $p(\lambda + |\mathbf{x}| + \log |\mathcal{S}|)$, and the proof size $|\pi|$ is $p(\lambda + \log |\mathcal{S}|)$.

Remark 2.6 (Public Verification vs. Designated Verifier)**.** We say a SNARK is *publicly-verifiable* if the verification state st is allowed to be public. Alternatively, a *designated-verifier* SNARK holds security only when the verification state st is kept *secret*, and only the holder of st can check proof.

Definition 2.7 (Zero-Knowledge [BCI+13, adapted])**.** A SNARK $\Pi_{\mathsf{SNARK}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ is computational zero-knowledge (i.e., zkSNARK) if there exists an efficient simulator $\mathcal{S}_{\mathsf{SNARK}} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for all efficient and stateful adversaries $\mathcal{A}$ and auxiliary input $z \in \{0, 1\}^{\mathsf{poly}(\lambda)}$ such that

$$\Pr[\mathsf{ExptZK}_{\Pi_{\mathsf{SNARK}}, \mathcal{A}, \mathcal{S}_{\mathsf{SNARK}}}(1^\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda), \tag{2.2.1}$$

where the experiment $\mathsf{ExptZK}_{\Pi_{\mathsf{SNARK}}, \mathcal{A}, \mathcal{S}_{\mathsf{SNARK}}}$ is defined as follows:

1. The challenger samples $b \xleftarrow{\text{R}} \{0, 1\}$.
   (a) If $b = 0$, the challenger computes $(\mathsf{crs}, \mathsf{st}) \leftarrow \mathsf{Setup}(1^\lambda)$ and sends $(\mathsf{crs}, \mathsf{st})$ to $\mathcal{A}$.
   (b) If $b = 1$, the challengers computes $(\widetilde{\mathsf{crs}}, \widetilde{\mathsf{st}}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}_1(1^\lambda)$ and sends $(\widetilde{\mathsf{crs}}, \widetilde{\mathsf{st}})$ to $\mathcal{A}$.

2. The adversary $\mathcal{A}$ outputs a statement $\mathbf{x}$ and a witness $\mathbf{w}$.
3. If $(\mathbf{x}, \mathbf{w}) \notin \mathcal{R}$, then the experiment halts with output 0. Otherwise, the experiment proceeds as follows:
   (a) If $b = 0$, the challenger replies with $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathbf{x}, \mathbf{w})$.
   (b) If $b = 1$, the challenger replies with $\pi \leftarrow \mathcal{S}_2(\mathsf{st}_{\mathcal{S}}, \mathbf{x})$.
4. 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. (2.2.1) is bounded by $1/2+\varepsilon$, we say the scheme is ε-computational zero-knowledge.

2.3 Linear PCPs

We now recall the notion of a linear PCP (LPCP) from Bitansky et al. [BCI$^+$13].

Definition 2.8 (Linear PCP [BCI$^+$13, adapted])**.** Let $\mathcal{R} : \mathbb{F}^n \times \mathbb{F}^h \to \{0,1\}$ be a binary relation [2] (with associated language $\mathcal{L}$) over a finite field $\mathbb{F}$. A k-query input-independent linear PCP for $\mathcal{R}$ with query length ℓ and soundness error ε is a tuple of algorithms $\Pi_{\mathsf{LPCP}} = (\mathcal{Q}_{\mathsf{LPCP}}, \mathcal{P}_{\mathsf{LPCP}}, \mathcal{V}_{\mathsf{LPCP}})$ with the following properties:

- $(\mathsf{st}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}()$: Query-generator outputs a query matrix $\mathbf{Q} \in \mathbb{F}^{\ell \times k}$ and a verification state st.
- $\boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(\mathbf{x}, \mathbf{w})$: On the input of a statement $\mathbf{x} \in \mathbb{F}^n$ and a witness $\mathbf{w} \in \mathbb{F}^h$, the prove algorithm outputs a proof $\boldsymbol{\pi} \in \mathbb{F}^\ell$.
- $b \leftarrow \mathcal{V}_{\mathsf{LPCP}}(\mathsf{st}, \mathbf{x}, \mathbf{a})$: On input the verification state st, the statement $\mathbf{x}$, and a vector of responses $\mathbf{a} \in \mathbb{F}^k$, the verification algorithm outputs a bit $b \in \{0,1\}$.

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

- **Completeness:** For all $\mathbf{x} \in \mathbb{F}^n$ and $\mathbf{w} \in \mathbb{F}^h$ where $\mathcal{R}(\mathbf{x}, \mathbf{w}) = 1$,

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

  where $(\mathsf{st}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}()$ and $\boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(\mathbf{x}, \mathbf{w})$.

- **Knowledge:** There exists an efficient extractor $\mathcal{E}_{\mathsf{LPCP}}$ such that for every $\boldsymbol{\pi}^* \in \mathbb{F}^\ell$, if

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

  then

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

[2] We can also define linear PCPs for infinite family of relations $\mathcal{R} = \bigcup_{\kappa \in \mathbb{N}} \mathcal{R}_\kappa$. In this case, the inputs to the query-generation and proving algorithms would take the relation index 1^κ as input, and the parameters $n, h, k, \ell, \varepsilon$ can all be functions of κ.

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 $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}$,
$$\{(\mathsf{st}, \mathbf{Q}, \mathbf{Q}^{\mathsf{T}}\boldsymbol{\pi})\} \equiv \{(\widetilde{\mathsf{st}}, \tilde{\mathbf{Q}}, \tilde{\mathbf{a}})\},$$
where $(\mathsf{st}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}()$, $\boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(\mathbf{x}, \mathbf{w})$, $(\widetilde{\mathsf{st}}, \tilde{\mathbf{Q}}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}_1()$, and $\tilde{\mathbf{a}} \leftarrow \mathcal{S}_2(\mathsf{st}_{\mathcal{S}}, \mathbf{x})$. When the statistical distance between these two distributions is δ, we say that Π_{LPCP} is δ-statistical HVZK. [3]

Linear PCP for R1CS. The quadratic arithmetic programs (QAPs) introduced by Gennaro et al. [GGPR13] 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.

Construction 2.9 (Linear PCP for R1CS [GGPR13, BCG+13, adapted])**.** Let $\mathcal{S} = (n, N_g, N_w, \{\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i\}_{i\in[N_g]})$ be an R1CS system over a finite field $\mathbb{F}$, where $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i \in \mathbb{F}^{N_w+1}$ (indexed from 0 to N_w). We additionally define the following components:

- $S = \{\alpha_1, \ldots, \alpha_{N_g}\} \subset \mathbb{F}$ be an arbitrary subset of $\mathbb{F}$.
- For each $i \in \{0, \ldots, N_w\}$, let $A_i, B_i, C_i : \mathbb{F} \to \mathbb{F}$ be the unique polynomials of degree $N_g - 1$, where $j \in [N_g]$,
$$A_i(\alpha_j) = a_{j,i}, \quad B_i(\alpha_j) = b_{j,i}, \quad C_i(\alpha_j) = c_{j,i}.$$
- Let $Z_{\mathcal{S}} : \mathbb{F} \to \mathbb{F}$ be the polynomial $Z_{\mathcal{S}}(z) := \prod_{j\in[N_g]}(z - \alpha_j)$, namely Z_S is the polynomial whose roots are the elements of S.

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

- $\mathcal{Q}_{\mathsf{LPCP}}()$: Sample $\tau \xleftarrow{\text{R}} \mathbb{F} \setminus S$. 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 $\mathbf{Q}$ shown as follows:

$$\mathbf{Q} = \begin{bmatrix} Z_S(\tau) & 0 & 0 & A_{n+1}(\tau) & \cdots & A_{N_w}(\tau) & 0 & 0 & \cdots & 0 \\ 0 & Z_S(\tau) & 0 & B_{n+1}(\tau) & \cdots & B_{N_w}(\tau) & 0 & 0 & \cdots & 0 \\ 0 & 0 & Z_S(\tau) & C_{n+1}(\tau) & \cdots & C_{N_w}(\tau) & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & \tau & \cdots & \tau^{N_g} \end{bmatrix}^{\mathsf{T}} \in \mathbb{F}^{(4+N_w+N_g-n)\times 4}. \tag{2.3.1}$$

[3] In particular, this definition separates the simulator $\mathcal{S}_{\mathsf{LPCP}}$ into a *statement-independent* algorithm $\mathcal{S}_1$ and a *statement-dependent* algorithm $\mathcal{S}_2$. This separation is important for arguing *adaptive* zero-knowledge of our zkSNARKs (where the adversary's statement can depend on the common reference string).

- $\mathcal{P}_{\mathsf{LPCP}}(\mathbf{x}, \mathbf{w})$: On the input $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}_{\mathcal{S}}$, sample $\delta_1, \delta_2, \delta_3 \xleftarrow{\text{R}} \mathbb{F}$. Construct polynomials $A, B, C : \mathbb{F} \to \mathbb{F}$, each with degree N_g, where

$$\begin{aligned} A(z) &:= \delta_1 Z_S(z) + A_0(z) + \sum_{i \in [N_w]} w_i A_i(z) \\ B(z) &:= \delta_2 Z_S(z) + B_0(z) + \sum_{i \in [N_w]} w_i B_i(z) \\ C(z) &:= \delta_3 Z_S(z) + C_0(z) + \sum_{i \in [N_w]} w_i C_i(z) \end{aligned} \tag{2.3.2}$$

  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 the 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 $a_1' = a_0 + a_1 + \mathbf{x}^{\mathsf{T}}\mathbf{a}$, $a_2' = b_0 + a_2 + \mathbf{x}^{\mathsf{T}}\mathbf{b}$, and $a_3' = c_0 + a_3 + \mathbf{x}^{\mathsf{T}}\mathbf{c}$. It accepts if

$$a_1' a_2' - a_3' - a_4 z = 0. \tag{2.3.3}$$

Theorem 2.10 (Linear PCP for QAP)**.** *Construction 2.9 is complete, has knowledge error $2N_g/(|\mathbb{F}| - N_g)$, and is perfect HVZK.*

Proof. Let $\mathcal{S} = (n, N_g, N_w, \{\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i\}_{i \in [N_g]})$ be an R1CS system over $\mathbb{F}$. We prove the properties separately:

- **Completeness:** Take any $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}_{\mathcal{S}}$, and let $(\mathsf{st}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}()$, $\boldsymbol{\pi} \xleftarrow{\text{R}} \mathcal{P}_{\mathsf{LPCP}}(\mathbf{x}, \mathbf{w})$, $\mathbf{a} \leftarrow \mathbf{Q}^{\mathsf{T}}\boldsymbol{\pi}$. 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 all $i \in [n]$, A is the polynomial in Eq. (2.3.2), 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 the extractor works, take any $\boldsymbol{\pi}^* = (\delta_1^*, \delta_2^*, \delta_3^*, \tilde{\mathbf{w}}^*, \mathbf{h}^*)$ where

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

We use $\boldsymbol{\pi}^*$ and $\mathcal{S}$ 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, we have $a_1' = A(\tau)$, $a_2' = B(\tau)$, $a_3' = C(\tau)$ 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, $\deg(P) \leq 2N_g$. $\mathcal{V}_{\mathsf{LPCP}}$ accepts if $a_1' a_2' - a_3' - a_4 z = 0$, where $z = Z(\tau)$, or equivalently, if

$$0 = A(\tau)B(\tau) - C(\tau) - H(\tau)Z_S(\tau) = P(\tau). \tag{2.3.4}$$

Suppose Eq. (2.3.4) holds with probability $\varepsilon > 2N_g/(|\mathbb{F}| - N_g)$; namely, the verifier accepts $\boldsymbol{\pi}^*$ 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 2.2) that $P \equiv 0$. Noticing $Z_S(\alpha_j) = 0$ for all $j \in [N_g]$, this means $A(\alpha_j)B(\alpha_j) = C(\alpha_j)$. By construction of A, B, C, let $\mathbf{u}^{\mathsf{T}} = [1 \mid \mathbf{x}^{\mathsf{T}} \mid (\tilde{\mathbf{w}}^*)^{\mathsf{T}}]$, we have

$$\mathbf{u}^{\mathsf{T}} \mathbf{a}_j \cdot \mathbf{u}^{\mathsf{T}} \mathbf{b}_j = \mathbf{u}^{\mathsf{T}} \mathbf{c}_j$$

for all $j \in [N_g]$. Since this holds for all $j \in [N_g]$, we have that $(\mathbf{x}, \tilde{\mathbf{w}}^*) \in \mathcal{R}_{\mathcal{S}}$, thus knowledge holds.

- **HVZK:** We first construct the simulator $\mathcal{S}_{\mathsf{LPCP}} = (\mathcal{S}_1, \mathcal{S}_2)$:
  - $\mathcal{S}_1()$: The statement-independent algorithm samples $(\widetilde{\mathsf{st}}, \tilde{\mathbf{Q}}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}()$, outputs $\widetilde{\mathsf{st}}$, $\tilde{\mathbf{Q}}$, $\mathsf{st}_{\mathcal{S}} = \widetilde{\mathsf{st}}$.
  - $\mathcal{S}_2(\mathsf{st}_{\mathcal{S}}, \mathbf{x})$: 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 $\tilde{a}_1, \tilde{a}_2, \tilde{a}_3 \xleftarrow{R} \mathbb{F}$, and 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 prove HVZK, it suffices to show that the distribution of the simulation and the real one are identical, for any $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}_\mathcal{S}$. By construction, the verification state and query matrix (output by $\mathcal{S}_1$) are identically distributed in the two cases. On the distribution of the responses, let $(\mathsf{st}, \mathbf{Q}) \xleftarrow{\text{R}} \mathcal{Q}_{\mathsf{LPCP}}()$, $\boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(\mathbf{x}, \mathbf{w})$, 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. (2.3.3) 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. (2.3.3) precisely coincides with the value $\tilde{a}_4$ sampled by $\mathcal{S}_{\mathsf{LPCP}}$ (based on 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 $\mathbb{F}$. Similarly, such 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 2.11 (Knowledge against Affine Srategies)**.** While our compiler only requires a linear PCP with knowledge against linear strategies (Theorem 3.23), 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 3.17 holds, we do not use this modified linear PCP in our concrete implementation; we mainly present the observation below for completeness.

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, i.e., an additional query that is a random linear combination of the remaining queries. While this technique can be applied directly to Construction 2.9, the number of queries of the linear PCP will increase. By expoliting the specific structure of the linear PCP in Construction 2.9, we can remove the need for an extra query. We describe the approach generally for any linear PCP whose query matrix $\mathbf{Q}^\mathsf{T}$ contains one of its columns as an elementary basis vector (or a scaled version thereof) and whose verification procedure is a low-degree algebraic circuit. Both properties hold for Construction 2.9.

- 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 $\mathbf{Q}^{\mathsf{T}} \in \mathbb{F}^{k \times m}$ is a basis vector $\mathbf{e}_j \in \mathbb{F}^k$. In this case, an affine strategy of the form $(\boldsymbol{\pi}, \delta\mathbf{e}_j)$ 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 $(\boldsymbol{\pi}, \delta\mathbf{e}_j)$.

- To extend such a linear PCP to provide soundness (resp., knowledge) against arbitrary affine strategies, we embed a random linear combination of other queries into the i^{th} query. In detail, let $\mathbf{q}_1, \ldots, \mathbf{q}_k$ be the linear PCP queries sampled by $\mathcal{Q}_{\mathsf{LPCP}}$ (i.e., the rows of $\mathbf{Q}^{\mathsf{T}} \in \mathbb{F}^{k \times m}$). The modified query-generation algorithm first samples $\gamma_\ell \xleftarrow{\text{R}} \mathbb{F}$ for $\ell \in [k]$ and $\ell \neq i$, then 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. Random 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 $a'_i \leftarrow a_i - \sum_{\ell \neq i} \gamma_i a_i$, 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 the modified linear PCP 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. [4] Consider an arbitrary affine strategy $(\boldsymbol{\pi}, \mathbf{b})$, and let $a_1, \ldots, a_\ell$ be the responses for the modified linear PCP defined above, we have $a_\ell = \mathbf{q}_\ell^{\mathsf{T}} \boldsymbol{\pi} + b_\ell$ for $\ell \neq i$ and $a_i = (\mathbf{q}'_i)^{\mathsf{T}} \boldsymbol{\pi} + b_i$. The verification relation first computes

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

then checks the polynomial relation

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

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 γ_i's are sampled uniformly and independently over $\mathbb{F}$ by $\mathcal{Q}_{\mathsf{LPCP}}$, by Lemma 2.2 (Schwartz-Zippel Lemma) we 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 $\ell \neq i$.

Moreover, by our previous observation that soundness (resp., knowledge) against strategies of affine form $(\boldsymbol{\pi}, \delta\mathbf{e}_j)$ is implied by soundness (resp., knowledge) against linear strategies when the query matrix $\mathbf{Q}^{\mathsf{T}} \in \mathbb{F}^{k \times m}$ contains one of its columns as a basis vector $\mathbf{e}_j$ (or a scaled version thereof).

[4] 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 this setting.

In the case of Construction 2.9, by observing $\mathbf{Q}^{\mathsf{T}} \in \mathbb{F}^{k \times m}$ in Eq. (2.3.1) ($i = 4$ in this case) that one of its columns is an elementary basis vector. The degree d of a_i in the verification relation (Eq. (2.3.3)) is $d = 1$, so the soundness error (resp., knowledge error) of the modified construction described above is $\max(\varepsilon, 1/|\mathbb{F}|)$, where ε is the soundness (resp., knowledge) error of the original construction (see Theorem 2.10).

Remark 2.12 (Soundness Amplification for Linear PCPs)**.** Construction 2.9 gives a 4-query linear PCP for any R1CS system with N_g constraints that has knowledge error $2N_g/(|\mathbb{F}| - N_g)$. To achieve statistical soundness, the straightforward method is to work over a field of superpolynomial size, while it is more efficient to work over smaller fields (and of small characteristic). We can use standard parallel amplification to amplify soundness, namely, for a k-query LPCP with query length m and knowledge error ε, we can obtain a $k\rho$-query LPCP with same query length and knowledge error ε^{ρ}. The setup algorithm samples ρ sets of query matrices $\mathbf{Q}_1, \ldots, \mathbf{Q}_\rho \in \mathbb{F}^{m \times k}$ and construct a query matrix $\mathbf{Q} = [\mathbf{Q}_1 \mid \ldots \mid \mathbf{Q}_\rho] \in \mathbb{F}^{m \times k\rho}$. The verifier accepts a response $\mathbf{a} = [\mathbf{a}_1 \mid \ldots \mid \mathbf{a}_\rho]$ only if all the responses are valid.

Chapter 3

Lattice-Based Succint Arguments

We introduce the underlying components of zkSNARKS in this chapter: the information-theoretic compiler (linear PCPs over extension fields) and the cryptographic compiler (linear-only vector encryption). We show how to combine these components together 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

Construction 2.9 gives a linear PCP for R1CS over any sufficient large field $\mathbb{F}$. In this work, we consider linear PCPs over quadratic extensions $\mathbb{F}_{p^2}$. As mentioned 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 one, we need to transform a linear PCP from $\mathbb{F}_{p^2}$ to a linear PCP over $\mathbb{F}_p$. We describe this transformation here.

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 relation $\mathcal{R}$ 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 $\mathcal{R}$ with query length $d\ell$ over the base field $\mathbb{F}_p$ with following properties:

- $\mathcal{Q}_{\mathsf{LPCP}}$: The query algorithm runs $(\mathsf{st}, \mathbf{Q}') \leftarrow \mathcal{Q}'_{\mathsf{LPCP}}()$, 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 constructed by replacing each entry $q'_{i,j} \in \mathbb{F}_{p^d}$ in $\mathbf{Q}'$ with the transpose of "multiply-by-$q'_{i,j}$" matrix $\mathbf{M}^{\mathsf{T}}_{q'_{i,j}} \in \mathbb{F}_p^{d \times d}$. Namely,

$$\mathbf{Q}' = \begin{bmatrix} q'_{1,1} & \cdots & q'_{1,k} \\ \vdots & \ddots & \vdots \\ q'_{\ell,1} & \cdots & q'_{\ell,k} \end{bmatrix} \text{ and } \mathbf{Q} = \begin{bmatrix} \mathbf{M}^{\mathsf{T}}_{q'_{1,1}} & \cdots & \mathbf{M}^{\mathsf{T}}_{q'_{1,k}} \\ \vdots & \ddots & \vdots \\ \mathbf{M}^{\mathsf{T}}_{q'_{\ell,1}} & \cdots & \mathbf{M}^{\mathsf{T}}_{q'_{\ell,k}} \end{bmatrix}.$$

Output the query matrix $\mathbf{Q} \in \mathbb{F}_p^{d\ell \times dk}$ and the verification state st.

- $\mathcal{P}_{\mathsf{LPCP}}$: The prover algorithm computes $\boldsymbol{\pi}' \leftarrow \mathcal{P}_{\mathsf{LPCP}}(\mathbf{x}, \mathbf{w})$. Let $\boldsymbol{\pi} \in \mathbb{F}_p^{d\ell}$ be the vector constructed by replacing each $\pi_i' \in \mathbb{F}_{p^d}$ with the vector representation $\mathbf{v}_{\pi_i'} \in \mathbb{F}_p^d$ of π_i'. Out the proof vector $\boldsymbol{\pi}$.
- $\mathcal{V}_{\mathsf{LPCP}}$: The verifier algorithm parses $\mathbf{a} = [\mathbf{v}_{a_1'} \mid \dots \mid \mathbf{v}_{a_k'}]$ and constructs $\mathbf{a}'$ by replacing each $\mathbf{v}_{a_i'} \in \mathbb{F}_p^d$ with $a_i' \in \mathbb{F}_{p^d}$. 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, has knowledge error ε and is prefect* HVZK*, then the same holds for Π_{LPCP} from Construction 3.1.*

Proof. We prove the properties separately:

- **Completeness:** Take any $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}$, compute $(\mathsf{st}, \mathbf{Q}') \in \mathcal{Q}'_{\mathsf{LPCP}}()$, $\boldsymbol{\pi}' \leftarrow \mathcal{P}'_{\mathsf{LPCP}}(\mathbf{x}, \mathbf{w})$. 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 $\mathbf{a} \leftarrow \mathbf{Q}^\mathsf{T}\boldsymbol{\pi}$. We write $\mathbf{a} = [\mathbf{a}_1, \dots, \mathbf{a}_k]$. By construction, for all $i \in [k]$,

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

  where $\mathbf{q}'_{\cdot,i} \in \mathbb{F}_{p^d}^\ell$ corresponds to the ith column of $\mathbf{Q}'$. This means that the vector $\mathbf{a}'$ computed by $\mathcal{V}_{\mathsf{LPCP}}$ satisfies $\mathbf{a}' = (\mathbf{Q}')^\mathsf{T}\boldsymbol{\pi}'$. Thus, completeness of Π_{LPCP} holds by following the completeness of Π'_{LPCP}.
- **Knowledge:** Compute $(\mathsf{st}, \mathbf{Q}') \leftarrow \mathcal{Q}'_{\mathsf{LPCP}}()$ and let $\mathbf{Q}$ be the matrix $\mathcal{Q}_{\mathsf{LPCP}}$ constructs from $\mathbf{Q}'$. Take any $\boldsymbol{\pi} \in \mathbb{F}_p^{d\ell}$ and let $\boldsymbol{\pi}' \in \mathbb{F}_{p^d}^\ell$ be the vector obtained by viewing $\boldsymbol{\pi}$ as ℓ blocks of d elements. 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 extractor $\mathcal{E}_{\mathsf{LPCP}}$ for Π_{LPCP} can invoke the extractor $\mathcal{E}'_{\mathsf{LPCP}}$ for Π'_{LPCP}, since any linear query $\mathbf{q}'$ that $\mathcal{E}'_{\mathsf{LPCP}}$ makes to $\langle \boldsymbol{\pi}', \cdot \rangle$ can be simulated via a linear query to $(\cdot)^\mathsf{T}\boldsymbol{\pi}$ by expanding each component in $\mathbf{q}'$ into a matrix of $\mathbb{F}_p^{d \times d}$. The knowledge of Π_{LPCP} now follows the knowledge soundness of Π'_{LPCP}.
- **Perfect HVZK:** On the input $\mathbf{x}$, the $\mathcal{S}_{\mathsf{LPCP}}$ runs the simulator $\mathcal{S}'_{\mathsf{LPCP}}(\mathbf{x})$ for Π'_{LPCP} by outputs $(\mathsf{st}, \tilde{\mathbf{Q}}', \tilde{\mathbf{a}}')$ where $\tilde{\mathbf{Q}}' \in \mathbb{F}_{p^d}^{\ell \times k}$ and $\tilde{\mathbf{a}}' \in \mathbb{F}_{p^d}^\ell$. The simulator $\mathcal{S}_{\mathsf{LPCP}}$ constructs $\tilde{\mathbf{Q}} \in \mathbb{F}_p^{d\ell \times dk}$ by replacing each entry $\tilde{q}'_{i,j} \in \mathbb{F}_{p^d}$ in $\tilde{\mathbf{Q}}'$ with $\mathbf{M}^\mathsf{T}_{\tilde{a}'_{i,j}} \in \mathbb{F}_p^{d \times d}$ and constructs $\tilde{\mathbf{a}} \in \mathbb{F}_p^{dk}$ by replacing each element $\tilde{a}'_i \in \mathbb{F}_{p^d}$ in $\tilde{\mathbf{a}}'$ with $\mathbf{v}_{\tilde{a}'_i} \in \mathbb{F}_p^d$. $\mathcal{S}_{\mathsf{LPCP}}$ outputs $(\mathsf{st}, \tilde{\mathbf{Q}}, \tilde{\mathbf{a}})$. Perfect HVZK of Π_{LPCP} follows by perfect HVZK of Π'_{LPCP}. □

3.2 Linear-Only Vector Encryption

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

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 following properties:

- $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$: On the input of security parameter λ and plaintext dimension ℓ, the setup algorithm outputs public parameters pp and a secret key sk.
- $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{v})$: On the input of the secret key sk and a plaintext vector $\mathbf{v} \in \mathbb{F}^\ell$, the encryption algorithm outputs a ciphertext ct.
- $\mathbf{v}/\perp \leftarrow \mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct})$: On the input of 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 $\perp$.
- $\mathsf{ct}^* \leftarrow \mathsf{Add}(\mathsf{pp}, \{\mathsf{ct}_i\}_{i\in[n]}, \{c_i\}_{i\in[n]})$: 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 parameter $\lambda \in \mathbb{N}$, a collection of plaintext vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathbb{F}^\ell$, and scalars $y_1, \ldots, y_k \in \mathbb{F}$, where $k = k(\lambda)$,

$$\Pr[\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}^*) = \sum_{i\in[k]} y_i \mathbf{v}_i] = 1 - \mathsf{negl}(\lambda), \tag{3.2.1}$$

  where $(\mathsf{sk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{v}_i)$ 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 Π_{Enc} is additively-homomorphic with respect to $S \subseteq R_p^k$ if Eq. (3.2.1) holds for all $(y_1, \ldots, y_k) \in S$. Note that additive-homomorphism implies the correctness in decryption.

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

$$\Pr[\mathcal{A}^{\mathcal{O}_b(\mathsf{sk},\cdot,\cdot)}(1^\lambda, \mathsf{pp}) = b] = \frac{1}{2} + \mathsf{negl}(\lambda), \tag{3.2.2}$$

  where $(\mathsf{sk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, $\mathcal{O}_b$ takes sk and $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{F}^\ell$ as input, and output $\mathsf{ct}_b \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{v}_b)$. We say Π_{Enc} is Q-query CPA secure if Eq. (3.2.2) holds against at most Q queries from $\mathcal{O}_b$,

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 any polynomial-size adversary $\mathcal{A}$, there is a polynomial-size extractor $\mathcal{E}$ such that, for all security parameter $\lambda \in \mathbb{N}$, any auxiliary input $\mathbf{z} \in \{0,1\}^{\mathsf{poly}(\lambda)}$, 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:

- The challenger begin by computing $(\mathsf{sk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$ and sampling $(\mathbf{v}_1, \ldots, \mathbf{v}_m) \leftarrow \mathcal{M}(1^\lambda, \mathsf{pp})$. It computes $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{v}_i)$ for $i \in [m]$ and runs $\mathcal{A}(\mathsf{pp}, \mathsf{ct}_1, \ldots, \mathsf{ct}_m; z)$ to obtain a tuple of new ciphertexts $(\mathsf{ct}'_1, \ldots, \mathsf{ct}'_k)$.
- The challenger computes $\mathbf{\Pi} \leftarrow \mathcal{E}(\mathsf{pp}, \{\mathsf{ct}_i\}_{i\in[m]}; z)$, $\mathbf{V}' \leftarrow \mathbf{\Pi} \cdot (\mathbf{v}_1, \ldots, \mathbf{v}_m)^\mathsf{T}$, where $\mathbf{\Pi} \in \mathbb{F}^{k\times m}$ and $\mathbf{V}' \in \mathbb{F}^{k\times \ell}$. The output of the experiment is 1 if for any $i \in [k]$, $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}'_i) \neq \perp$ and $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}'_i) \neq \mathbf{v}'_i$. Otherwise, the output of the experiment is 0.

Circuit Privacy. In additional to the previous properties, we additionally require a *circuit privacy* property [Gen09]. Circuit privacy says that the ciphertext output Add can be simulated given only the underlying plaintext value, *without* knowledge of the linear combination used to construct the ciphertext. This is important to argue zero knowledge (see Section 3.4 and Theorem 3.26).

Definition 3.5 (Circuit Privacy [Gen09, adapted])**.** 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 Π_{Enc} satisfy circuit privacy if for all efficient and stateful adversaries $\mathcal{A}$, there exists an efficient simulator $\mathcal{S}$ such that for all security parameters $\lambda \in \mathbb{N}$,

$$\Pr[\mathsf{ExptCircuitPriv}_{\Pi_{\mathsf{Enc}},\mathcal{A},\mathcal{S}}(1^\lambda) = 1] = \frac{1}{2} + \mathsf{negl}(\lambda), \tag{3.2.3}$$

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

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

We also consider a weaker form of circuit privacy in this thesis where we additionally constrain the adversary to choose the coefficients from a *priori* specified $S \subseteq \mathbb{F}$. Int this case, we say Π_{Enc} satisfies circuit privacy with respect to S. Moreover, when the probability of Eq. (3.2.3) is bounded by $1/2 + \varepsilon$, we say Π_{Enc} is ε-circuit privacy.

Remark 3.6 (Multi-Query Circuit Privacy)**.** We can define a multi-query variant of Definition 3.5 where the adversary can adaptively choose *multiple* collections of coefficients $y_1, \ldots, y_k \in R_p$ and on each query, learn either homomorphically-evaluated ciphertext from Add or simulated ciphertext from $\mathcal{S}$. The multi-query notion is useful to argue *multi-theorem* zero-knowledge when compiling a linear PCP into a preprocessing SNARG [BCI+13]. Definition 3.5 implies the multi-query variant by a standard hybrid argument.

Remark 3.7 (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 (linear targeted malleability) 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 (Remark 2.11 shows how to construct linear PCPs with knowledge against affine strategies).

Remark 3.8 (Auxiliary Input Distributions.)**.** Definition 3.4 requires the extractor succeed for an *arbitrary* auxiliary input z. While the formulation is convenient for the purpose of definition, it might 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 case, it suffices to consider a relaxiation 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 uniformly random). We refer to [BCTV14b, BCI$^+$13] for more discussion.

3.3 Candidate Linear-Only Vector Encryption

(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 [BGV12, LS15])**.** Fix a security number n, sample number $m = m(\lambda)$, integers $n = n(\lambda), q = q(\lambda), d = d(\lambda)$ where d is a power of 2. 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 $\mathbf{A} \xleftarrow{\mathrm{R}} R_q^{n\times m}$, $\mathbf{s} \leftarrow \chi^n$, $\mathbf{e} \leftarrow \chi^m$ and $\mathbf{u} \xleftarrow{\mathrm{R}} R_q^m$, the following two distributions are computationally indistinguishable:

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

Remark 3.10 (Relation to LWE and RLWE)**.** The MLWE assumption generalizes both the classic learning with errors (LWE) assumption [Reg05] and the ring learning with errors (RLWE) assumption [LPR10]: 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,[1] 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 used for *re-randomization* of homomorphically-evaluated ciphertexts. These parameters 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 and efficient since we avoid the need to compute $\lfloor k \cdot q/p \rceil \bmod q$ (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 2.1) 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, [2] 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

[1] Note that 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.

[2] 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).

sparsification parameter. During setup, we sample a random matrix $\mathbf{T} \xleftarrow{\text{R}} \mathbb{F}^{\ell \times \tau}$ 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 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 2 and let $R = \mathbb{Z}[x]/(x^d + 1)$. Fix lattice parameter $p = p(\lambda), q = q(\lambda), n = n(\lambda)$ and an error distribution $\chi = \chi(\lambda)$ over R_q. Moreover, we define:

- 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:

- $(\mathsf{sk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(1^{\lambda}, 1^{\ell})$: Sample $\mathbf{A} \xleftarrow{\text{R}} R_q^{n \times n}$, $\mathbf{S} \leftarrow \chi^{n \times \ell'}, \mathbf{T} \xleftarrow{\text{R}} R_q^{\tau \times \ell}$, 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 $\mathsf{pp} = (\mathbf{A}, \mathbf{D})$.
- $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{v})$: On the input of secret key $\mathsf{sk} = (\mathbf{S}, \mathbf{T})$ and plaintext 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 $\mathbf{a} \leftarrow R_q^n, \mathbf{e} \leftarrow \chi^{\ell'}$ and compute $\mathbf{c} \leftarrow \mathbf{S}^{\mathsf{T}}\mathbf{a} + p\mathbf{e} + \mathbf{u} \in R_q^{\ell'}$. Output the ciphertext $\mathsf{ct} = (\mathbf{a}, \mathbf{c})$.
- $\mathsf{ct}^* \leftarrow \mathsf{Add}(\mathsf{pp}, \{\mathsf{ct}_i\}_{i \in [k]}, \{y_i\}_{i \in [k]})$: On the input of 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 $\mathbf{r} \leftarrow \chi^n, \mathbf{e}_a \leftarrow \chi^n, \mathbf{e}_c \xleftarrow{\text{R}} [-B, B]^{\ell'}$ and outputs

$$\mathsf{ct}^* = \Big(\sum_{i \in [k]} y_i \mathbf{a}_i + \mathbf{A}\mathbf{r} + p\mathbf{e}_a, \sum_{i \in [k]} y_i \mathbf{c}_i + \mathbf{D}\mathbf{r} + p\mathbf{e}_c \Big). \tag{3.3.1}$$

- $\mathbf{v}/\perp \leftarrow \mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct})$: On the input of the secret key $\mathsf{sk} = (\mathbf{S}, \mathbf{T})$ and ciphertext $\mathsf{ct} = (\mathbf{a}, \mathbf{c})$, compute $\mathbf{z} \leftarrow \mathbf{c} - \mathbf{S}^{\mathsf{T}}\mathbf{a} \in R_q^{\ell'}$. Compute $\mathbf{u} = \mathbf{z} \mod p$, 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 $\mathbf{v}_2 = \mathbf{T}\mathbf{v}_1$, otherwise $\perp$.

Correctness and Security Analysis. Below, we describe the main theorems and formal analysis on the correctness and security of Construction 3.11.

Theorem 3.12 (Additive Homomorphism)**.** *Let λ be a security parameter, and p, q, n, χ, ℓ', B be as defined in Construction 3.11. Suppose χ is subgaussian* [3] *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, B_1, B_2 be bounds, $S = \{\mathbf{y} \in R_p^k : \|\mathbf{y}\|_1 \le B_1$ and $\|\mathbf{y}\|_2 \le B_2\}$. If $\ell', n > 8$, and

$$q > 2p(B + \gamma_R B_2 C s + \gamma_R B_1/2 + 2\gamma_R n C^2 s^2) + p, \tag{3.3.2}$$

then Eq. (3.2.1) *holds with probability $1 - (4n+2)d\ell' \exp(-\pi C^2)$ for all $(y_1, \ldots, y_k) \in S$.*

Proof. We begin with the concrete statement. Take any collection of plaintext 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 $i \in [k]$. Define 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 $i \in [k]$, 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,

$$\mathbf{c}^* = \sum_{i\in[k]} \left(y_i \mathbf{S}^\mathsf{T}\mathbf{a}_i + y_i p\mathbf{e}_i + y_i \mathbf{u}_i \right) + \mathbf{S}^\mathsf{T}\mathbf{A}\mathbf{r} + p\mathbf{E}^\mathsf{T}\mathbf{r} + p\mathbf{e}_c. \tag{3.3.3}$$

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,

$$\mathbf{z}^* = \sum_{i\in[k]} y_i \mathbf{u}_i \bmod p + p \underbrace{\left(\tilde{\mathbf{u}} + \sum_{i\in[k]} y_i \mathbf{e}_i + \mathbf{E}^\mathsf{T}\mathbf{r} + \mathbf{e}_c - \mathbf{S}^\mathsf{T}\mathbf{e}_a \right)}_{\mathbf{e}'} \in R_q^{\ell'}. \tag{3.3.4}$$

If $\mathbf{e}'$ satisfies $\|\mathbf{e}'\|_\infty < q/(2p) - 1/2$, then Eq. (3.3.4) holds over R (and not just modulo q). Then, the

[3] 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.

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 element in $\mathbf{E}$ and $\mathbf{r}$ is bounded by Cs with probability at least $1-2d\exp(-\pi C^2)$. By a union bound, $\|\mathbf{E}^{\mathsf{T}}\mathbf{r}\|_\infty \le \gamma_R n C^2 s^2$ 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 $\|\mathbf{S}^{\mathsf{T}}\mathbf{e}_a\|_\infty \le \gamma_R n C^2 s^2$ with probability at least $1 - 2dn\ell' \exp(-\pi C^2)$.

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

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

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

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.*

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

$$\left(\mathbf{A}', \mathbf{S}^{\mathsf{T}}\mathbf{A}' + (\mathbf{E}')^{\mathsf{T}}\right) \text{ and } \left(\mathbf{A}', (\mathbf{U}')^{\mathsf{T}}\right), \tag{3.3.5}$$

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.
  (a) 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 sends $\mathsf{pp} = (\mathbf{A}, \mathbf{D})$ to $\mathcal{A}$.
  (b) 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 sends $\mathsf{ct}_i = (\mathbf{a}_i, \mathbf{c}_i)$ to $\mathcal{A}$.
  (c) After Q queries, $\mathcal{A}$ outputs a bit b'.

  The output of the experiment is 1 if $b' = b$, otherwise 0.
- 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 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 $i \in [Q]$). Thus, the outputs of Hyb_0 and Hyb_1 are computationally indistinguishable by $\mathsf{MLWE}_{n,m,d,q,\chi}$ assumption.

Noticing that in Hyb_1, the advantage of distinguishing ciphertext is exactly 1/2 since each $\mathbf{r}_i$ for $i \in [Q]$ is uniform and independent over $R_q^{\ell'}$ and $\gcd(p, q) = 1$. Thus, the challenge ciphertexts perfectly hide $\mathbf{v}_i$'s. □

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, B_1, B_2 be bounds, $S = \{\mathbf{y} \in R_p^k : \|\mathbf{y}\|_1 \le B_1$ 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

$$\varepsilon = (4n+2)d\ell' \exp(-\pi C^2) + \frac{d\ell'(\gamma_R B_2 C s + \gamma_R B_1/2 + 2\gamma_R n C^2 s^2)}{B}. \tag{3.3.6}$$

Proof. 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 $\mathbf{a} \overset{\text{R}}{\leftarrow} R_q^n$ 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.
  - (a) The challenger begins by sampling $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^{\lambda}, 1^{\ell})$ where $\mathsf{pp} = (\mathbf{A}, \mathbf{D})$ and $\mathsf{sk} = (\mathbf{S}, \mathbf{T})$, then sends pp and sk to $\mathcal{A}$.
  - (b) $\mathcal{A}$ outputs a collection of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$.
  - (c) The challenger computes $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{v}_i)$ for $i \in [k]$, where 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 $\mathbf{u}_i^{\mathsf{T}} = [\mathbf{v}_i^{\mathsf{T}} \mid (\mathbf{T}\mathbf{v}_i)^{\mathsf{T}}]$. Challenger sends $\mathsf{ct}_1, \ldots, \mathsf{ct}_k$ to $\mathcal{A}$.
  - (d) Adversary $\mathcal{A}$ outputs a set of coefficients $y_1, \ldots, y_k \in R_p$.
  - (e) The challenger computes ct_0^* and ct_1^* as follows:
    - For ct_0^*, the challenger uses the Add algorithm: 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 $\mathcal{S}$: It samples $\mathbf{a}_1^* \overset{\text{R}}{\leftarrow} R_q^n$, $\tilde{\mathbf{e}} \leftarrow [-B, B]^{d\ell'}$, computes $\mathbf{c}_1^* \leftarrow \mathbf{S}^{\mathsf{T}}\mathbf{a}_1^* + p\tilde{\mathbf{e}} + \mathbf{u}$, where $\mathbf{u}^{\mathsf{T}} = [\sum_{i\in[k]} y_i\mathbf{v}_i^{\mathsf{T}} \mid \sum_{i\in[k]} y_i(\mathbf{T}\mathbf{v}_i)^{\mathsf{T}}] \in R_p^{\ell'}$, and sets $\mathsf{ct}_1^* = (\mathbf{a}_1^*, \mathbf{c}_1^*)$.
  - (f) The challenger samples $b \overset{\text{R}}{\leftarrow} \{0, 1\}$ and sends ct_b^* to $\mathcal{A}$.
  - (g) $\mathcal{A}$ outputs $b' \in \{0, 1\}$.

  The output of the experiment is 1 if $b' = b$, otherwise 0.
- Hyb_1: Same as Hyb_0, except the challenger computes $\mathbf{u}^{\mathsf{T}} = [\sum_{i\in[k]} y_i\mathbf{v}_i^{\mathsf{T}} \mid \sum_{i\in[k]} y_i(\mathbf{T}\mathbf{v}_i)^{\mathsf{T}}] \in R_p^{\ell'}$ and $\mathbf{c}_0^* \leftarrow \mathbf{S}^{\mathsf{T}}\mathbf{a}_0^* + p\mathbf{e}_c + \mathbf{u}$.
- Hyb_2: Same as Hyb_1 except the challenger samples $\mathbf{a}_0^* \overset{\text{R}}{\leftarrow} R_q^n$.

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 3.15. *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. (3.3.6)).*

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. (3.3.3)), in Hyb_0,

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

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 $1-(4n+2)d\ell'\exp(-\pi C^2)$,

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

In Hyb_1, $\mathbf{c}_0^* = \mathbf{S}^\mathsf{T}\mathbf{a}_0^* + p\mathbf{e}_c + \mathbf{u}$. By Lemma 2.1 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. $\square$

Lemma 3.16. *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 distribution of y_i and $\mathbf{a}_i$). This is precisely the distribution in Hyb_2. Thus, $\mathcal{B}$ breaks MLWE with advantage ε'. $\square$

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 3.15 and 3.16, 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 $B_1, B_2 = pk$. $\square$

Conjecture 3.17 (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 linearly-only (Definition 3.4).*

Remark 3.18 (Plausibility of Linear-Only Conjecture)**.** We make a few remarks on the plausibility of Conjecture 3.17.

- **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 on existing ciphertexts. The approach we take in Construction 3.11 to implement this 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, and it is not known that *vanilla* Regev-style 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.

Extensions and Variants. We will describe an extension of Construction 3.11 to higher-degree extensions (Remark 3.19) and an alternative approach based on bit decomposition to decrease noise growth (Remark 3.20).

Remark 3.19 (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 $k \geq 1$. This allows us to directly compile linear PCPs over a higher-degree extension field into preprocessing zkSNARKs.

In addition, 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 $F_1(x), \ldots, F_\ell(x) \bmod p$), 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 encrypted responses to all ℓ R1CS systems. In this way, a *single* ciphertext contains 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 (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 (based on FFT) to implement the homomorphic operations. [4] In this thesis, we focus on modules of rank 1 and 2 where the additional cost of the polynomial (when $d = 2$) is small. Extending to modules of higher rank and taking advantage of batching in a *concretely-efficient* manner is an interesting direction for further exploration.

Remark 3.20 (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 number of ciphertexts (and the length of the linear combination) by a factor of m. In particular, an encryption of a vector $\mathbf{v} \in R_p^\ell$ now consists of the 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 $\alpha = \alpha_{m-1} \cdots \alpha_1\alpha_0$ is the value 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 $mkp^{1/m}$; 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 requisite 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. More precisely, 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 Bitansky et al. [BGV12]:

- $\mathbf{x}' \leftarrow \mathsf{Scale}(\mathbf{x}, q, q', p)$: 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 $\mathbf{x}' = \mathbf{x} \pmod{p}$.

Theorem 3.21 (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*

$$\|\mathbf{z}\|_\infty < q/2 - (1 + n\gamma_R Cs) \cdot (p/2) \cdot (q/q'). \tag{3.3.7}$$

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

[4] 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.

Proof. 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,

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

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 $\left\|\mathbf{S}^\mathsf{T}\Big(\mathbf{a}' - (q'/q) \cdot \mathbf{a}\Big)\right\|_\infty \le \gamma_R n C s (p/2)$ 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. □

3.4 zkSNARKs from Linear-Only Encryption

We state the result of Bitansky et al. [BCI+13] for construction zkSNARKs from linear PCPs and linear-only vector encryption. We specifically state the variant by Boneh et al. [BISW17] based on linear-only vector encryption.

Construction 3.22 (SNARK from Linear-Only Vector Encryption)**.** Let $\mathbb{F}$ be a finite field, $\mathcal{S}$ be an R1CS system over $\mathbb{F}$, and let $\mathcal{R}_\mathcal{S}$ be the associated relation. The construction has 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 $\mathcal{R}_\mathcal{S}$.

- Let $\Pi_{\mathsf{Enc}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Add})$ be a secret-key additively-homomorphic vector encryption scheme for $\mathbb{F}^k$.

We construct a single-theorem designated-verifier zkSNARK $\Pi_{\mathsf{SNARK}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ for $\mathcal{R}_{\mathcal{S}}$ as follows:

- $\mathsf{Setup}(1^\lambda, 1^\ell)$: On input the security parameter λ and the circuit family parameter ℓ, run $(\mathsf{st}_{\mathsf{LPCP}}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}()$ where $\mathbf{Q} \in \mathbb{F}^{m \times \ell}$. Write $\mathbf{q}_i^{\mathsf{T}} \in \mathbb{F}^\ell$ to denote the i^{th} row of $\mathbf{Q}$ for $i \in [m]$. Then compute $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{Setup}_{\mathsf{Enc}}(1^\lambda, 1^k)$ and $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}_{\mathsf{Enc}}(\mathsf{sk}, \mathbf{q}_i^{\mathsf{T}})$ for $i \in [m]$. Output the common reference string $\mathsf{crs} = (\mathsf{pp}, \mathsf{ct}_1, \ldots, \mathsf{ct}_m)$ and the verification state $\mathsf{st} = (\mathsf{st}_{\mathsf{LPCP}}, \mathsf{sk})$.

- $\mathsf{Prove}(\mathsf{crs}, \mathbf{x}, \mathbf{w})$: On input the common reference string $\mathsf{crs} = (\mathsf{pp}, \mathsf{ct}_1, \ldots, \mathsf{ct}_m)$, a statement $\mathbf{x}$, and a witness $\mathbf{w}$, the prover first construct an LPCP proof $\boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(\mathbf{x}, \mathbf{w})$. The prover then homomorphically computes the linear PCP response $\mathsf{ct}^* \leftarrow \mathsf{Add}_{\mathsf{Enc}}(\mathsf{pp}, \{\mathsf{ct}_1, \ldots, \mathsf{ct}_m\}, \{\pi_1, \ldots, \pi_m\})$, and outputs $\pi = \mathsf{ct}^*$.

- $\mathsf{Verify}(\mathsf{st}, \mathbf{x}, \pi)$: 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 decrypts $\mathbf{a} \leftarrow \mathsf{Decrypt}_{\mathsf{Enc}}(\mathsf{sk}, \mathsf{ct}^*)$. Then it outputs 0 if $\mathbf{a} = \perp$, otherwise $\mathcal{V}_{\mathsf{LPCP}}(\mathsf{st}_{\mathsf{LPCP}}, \mathbf{x}, \mathbf{a})$.

Completeness and Knowledge. Completeness of Construction 3.22 follows immediately from the completeness of the underlying linear PCP and the completeness of the additively-homomorphic encryption scheme. Knowledge follows from the linear-only property and the knowledge of the underlying linear PCP. Our analysis follows closely from the corresponding analysis from Bitansky et al. [BCI+13] and Boneh et al. [BISW17], we simply state the theorem here.

Theorem 3.23 (SNARKs from Linear-Only Vector Encryption [BCI+13, BISW17])**.** *If Π_{LPCP} is statistical sound against linear prover and Π_{Enc} is CPA-secure (for up to m messages) and strictly linear-only, then Π_{SNARK} from Construction 3.22 is a designated-verifier SNARK for $\mathcal{R}_{\mathcal{S}}$ in the preprocessing model.*

Remark 3.24. The only difference between Theorem 3.23 and the corresponding statements in previous works [BCI+13, BISW17] is that the previous works consider linear-only encryption with support for affine strategies, and thus, they require a linear PCP (or linear interactive proof) that provide soundness 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 soundness against affine strategies with small overhead (see Remark 2.11).

Remark 3.25 (Reusability)**.** A limitation of Construction 3.22 is that it only provides one-time soundness in designated-verifier model. *Reusable soundness* is a much stronger notion, where soundness holds even a malicious prover has access to the verification oracle. Bitansky et al. [BCI+13] showed that combining linear PCP satisfying reusable soundness (also known as "strong soundness") with an encryption scheme satisfying an *interactive* linear-only assumption yields a designated-verifier SNARK with reusable soundness. Under this stronger security notion, the adversary has access to an oracle that can check the well-formedness of ciphertexts. [5] While it is straightforward to realize strong soundness for the linear PCP, our vector encryption scheme (Construction 3.11) does not satisfy the stronger linear-only assumption. Moreover, 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 soundness (as we consider here) does imply soundness for logarithmically-many proofs.

In cases where the 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 soundness is always detectable; namely, to break soundness, the malicious prover has to first submit multiple proofs that cause the verifier to reject (i.e., a super-constant number of rejections). 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 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. By HVZK of the linear PCP, we can simulate the linear PCP responses given only the statement. By Circuit Privacy, we can say the encrypted linear PCP responses can be simulated given only the simulated linear PCP response.

Theorem 3.26 (Zero-Knowledge from Circuit Privacy). *Let Π_{LPCP} and Π_{Enc} be as defined in Theorem 3.23. If Π_{LPCP} satisfies perfect honest-verifier zero-knowledge and Π_{Enc} is CPA-secure (for up to m message) and computationally (resp., statistically) circuit private, then Π_{SNARK} from Construction 3.22 is computationally (resp., statistically) zero-knowledge. More precisely, if Π_{LPCP} is δ-HVZK and Π_{Enc} is ε-circuit private, then Π_{SNARK} from Construction 3.22 is $2(\delta+\varepsilon)$-zero-knowledge.*

Proof. 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 $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ for Π_{SNARK}:

- $\mathcal{S}_1(1^\lambda, 1^\ell)$: On the input of λ and ℓ, run $(\widetilde{\mathsf{st}}_{\mathsf{LPCP}}, \tilde{\mathbf{Q}}, \mathsf{st}_{\mathsf{LPCP},\mathcal{S}}) \leftarrow \mathcal{S}_{\mathsf{LPCP},1}()$, where $\tilde{\mathbf{Q}} \in \mathbb{F}^{m\times k}$. Compute $(\widetilde{\mathsf{pp}}, \widetilde{\mathsf{sk}}) \leftarrow \mathsf{Setup}_{\mathsf{Enc}}(1^\lambda, 1^k)$ and $\widetilde{\mathsf{ct}}_i \leftarrow \mathsf{Encrypt}_{\mathsf{Enc}}(\widetilde{\mathsf{sk}}, \tilde{\mathbf{q}}_i^{\mathsf{T}})$ for $i \in [m]$. Output $\widetilde{\mathsf{crs}} = (\widetilde{\mathsf{pp}}, \widetilde{\mathsf{ct}}_1, \ldots, \widetilde{\mathsf{ct}}_m)$,

[5]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.

verification state $\widetilde{\mathsf{st}} = (\widetilde{\mathsf{st}}_{\mathsf{LPCP}}, \widetilde{\mathsf{sk}})$, and the simulation state $\mathsf{st}_{\mathcal{S}} = (\mathsf{st}_{\mathsf{LPCP},\mathcal{S}}, \mathsf{pp}, \mathsf{sk})$.

- $\mathcal{S}_2(\mathsf{st}_{\mathcal{S}}, \mathbf{x})$: On the input of $\mathsf{st}_{\mathcal{S}} = (\mathsf{st}_{\mathsf{LPCP},\mathcal{S}}, \mathsf{pp}, \mathsf{sk})$ 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 $\tilde{\pi} = \mathcal{S}_{\mathsf{Enc}}(1^\lambda, \mathsf{pp}, \mathsf{sk}, \tilde{\mathbf{a}})$.

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

- Hyb_0: This is the real game. The challenger first samples a bit $b \xleftarrow{\mathrm{R}} \{0,1\}$, $(\mathsf{crs}, \mathsf{st}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, and $(\widetilde{\mathsf{crs}}, \widetilde{\mathsf{st}}, \mathsf{st}_{\mathsf{LPCP},\mathcal{S}}) \leftarrow \mathcal{S}_1(1^\lambda, 1^\ell)$. 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 $\mathcal{R}(\mathbf{x}, \mathbf{w}) = 1$ (aborting the experiment otherwise), then computes $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathbf{x}, \mathbf{w})$ and $\tilde{\pi} \leftarrow \mathcal{S}_1(\mathsf{st}_{\mathcal{S}}, \mathbf{x})$. 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 $\mathsf{crs} = (\mathsf{pp}, \mathsf{ct}_1, \ldots, \mathsf{ct}_m)$ and let $\mathsf{st} = (\mathsf{st}_{\mathsf{LPCP}}, \mathsf{sk})$ be the corresponding verification state. To construct π, the challenger now computes $\boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(\mathbf{x}, \mathbf{w})$ and sets $\pi \leftarrow \mathcal{S}_{\mathsf{Enc}}(1^\lambda, \mathsf{pp}, \mathsf{sk}, \mathbf{a})$, where $\mathbf{a} = \mathbf{Q}^\mathsf{T}\boldsymbol{\pi}$.

- 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 $(\mathsf{st}_{\mathsf{LPCP}}, \mathbf{Q}, \mathsf{st}_{\mathsf{LPCP},\mathcal{S}}) \leftarrow \mathcal{S}_{\mathsf{LPCP},1}()$. 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:

- 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 $\mathsf{Add}_{\mathsf{Enc}}(\mathsf{pp}, \{\mathsf{ct}_1, \ldots, \mathsf{ct}_m\}, \{\pi_1, \ldots, \pi_m\})$, 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 δ-circuit private, then

$$|\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]| \leq 2\delta.$$

- 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 ε-HVZK, then

$$|\Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1]| \leq 2\varepsilon.$$

- 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 $\mathcal{S}$). This means that for all adversaries $\mathcal{A}$, $\Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1] = 1/2$, and the claim holds. $\square$

Remark 3.27 (Leakage-Resilient Linear PCPs)**.** While circuit privacy played a central role in the analysis of Theorem 3.26, augmenting our linear-only vector encryption scheme (Construction 3.11) with circuit privacy incurs a non-trivial concrete cost. As shown in Fig. 4.4, instantiating the encryption scheme in a setting without circuit privacy (corresponds to the setting where $\kappa = 0$), we can achieve a 40% reduction in prover time and a 52% reduction in proof size for the resulting zkSNARK. In fact, for this parameter setting, the prover time of our lattice-based instantiation is 2.3× faster than the pairing-based construction of Groth [Gro16]. 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. (3.3.1) to instead output

$$\mathsf{ct}^* = (\mathbf{a}^*, \mathbf{c}^*) = \Big(\sum_{i \in [k]} y_i \mathbf{a}_i,\ \sum_{i \in [k]} y_i \mathbf{c}_i \Big). \tag{3.4.1}$$

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, in this case, we have that

$$\mathbf{c}^* = \mathbf{S}^\mathsf{T} \mathbf{a}^* + \sum_{i \in [k]} y_i \mathbf{u}_i \bmod p + p \left(\tilde{\mathbf{u}} + \sum_{i \in [k]} y_i \mathbf{e}_i \right),$$

where $\tilde{\mathbf{u}}, \tilde{\mathbf{u}}' \in R^{\ell'}$, $\|\tilde{\mathbf{u}}'\|_\infty < p/2$, and $\sum_{i \in [k]} y_i \mathbf{u}_i = p\tilde{\mathbf{u}} + \tilde{\mathbf{u}}' \in R^{\ell'}$. In the standard circuit-privacy setting, the simulator is only given $\sum_{i \in [k]} y_i \mathbf{u}_i \bmod p$ (i.e., the value of the linear combination modulo p) and must be able to simulate the ciphertext ct^* given only this information. However, if we consider a relaxed version of circuit privacy where we additionally give the simulator some additional "hints" about the coefficients y_i: namely, the value of $\sum_{i \in [k]} y_i \mathbf{a}_i \in R_q^n$ and the value of $\tilde{\mathbf{u}} + \sum_{i \in [k]} y_i \mathbf{e}_i \in R^{\ell'}$ (technically, the second relation is also over $R_q^{\ell'}$, but for our parameter settings, all of the terms in the sum are small that no reduction modulo q occur).

If we now consider a "leakage-resilient" linear PCP where HVZK holds with respect to this partial leakage on the linear PCP coefficients, then we can still apply the Bitansky et al. [BCI+13] compiler to obtain a zkSNARK. It is interesting to analyze whether the QAP-based linear PCPs we consider in this work remain statistical HVZK in the presence of this leakage. A positive result would yield an improvement to the concrete efficiency of our zkSNARK. The main source of leakage is the verifier learns a linear combination (over R) of (Gaussian-distributed) vectors $\mathbf{e}_i$ with coefficients drawn from the linear PCP. In this case, the length k of the linear PCP is significantly larger than the dimension ℓ' of the error vectors, so this is a severely under-determined linear system.

In the zkSNARK construction, the underlying linear PCP is randomized, and it is not clear that leaking a small number of linear combinations will allow a malicious verifier to efficiently extract even a single coefficient from the linear PCP. Based on this analysis, it seems plausible that instantiating the zkSNARK with this variant of the linear-only vector encryption scheme still provides some degree of zero-knowledge in a *heuristic* sense. We emphasize though that all of our evaluation and benchmarks we consider in this work use the circuit-private version of Construction 3.11 and ensures provable zero-knowledge. But understanding leakage-resilience for linear PCPs may provide further improvements to the concrete efficiency of lattice-based zkSNARKs.

Chapter 4

Implementation and Evaluation

In this section, we provide an overview of our lattice-based zkSNARK implementation constructed by combining Construction 2.9 with Construction 3.11.

4.1 Linear PCP Implementation

The prover's computation in zkSNARK from Construction 3.22 consists of two main components:

- Computing the linear PCP proof (in Construction 2.9 and Theorem 2.10).
- Homomorphically compute 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 Section 2.3 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 will refer to as "radix-2 FFTs"), we require that $\mathbb{F}$ contains a multiplicative subgroup of order 2^d where $2^d > N_g$. The construction of Ben-Sasson et al. use a specially-chosen elliptic curve group whose order is divisible by a large power of 2, while 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, $\mathbb{F}_{p^2}$ has a subgroup of order 2^{d+1} if $p = 2^d \pm 1$. However, if $N_g > 2p$, $\mathbb{F}_{p^2}$ will *never* have a sufficiently large subgroup for a direct usage of radix-2 FFTs. [1]

[1] 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.

Our Approach. When the field $\mathbb{F}$ contains a multiplicative subgroup whose order is moderately large power of two (e.g., 2^d), we can still leverage multiple radix-2 FFTs to eficiently implement multipoint polynomial evaluation and interpolation over a domain $D \subset \mathbb{F}$, where $|D| = k \cdot 2^d$ for some small positive integer $k > 1$. 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.

We follow noatations from [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 $A(z)$ of degree less than $|D|$, such that for all $\alpha \in D$, $A(\alpha) = A'(\alpha)$.

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. Let $\xi_1 = 1$ and take $\xi_i \in \mathbb{F}^* \setminus H_1$ for $i \in \{2, \ldots, k\}$, such that $H_i = \xi_i H$ are pairwise disjoint *cosets* of H_1. We define the domain $D = \cup_{i \in [k]} H_i$ for multipoint evaluation and interpolation. 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

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

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

- Let $\mathbf{b} = \mathbf{B} \cdot \mathbf{a} \in \mathbb{F}^{k \cdot 2^d}$ and $\hat{\mathbf{a}}'_i = (a'_{(i-1)2^d+1}, \ldots, a'_{i \cdot 2^d}) \in \mathbb{F}^{2^d}$. By construction, $\hat{\mathbf{b}}_i = \sum_{j \in [k]} \xi_j^{(j-1)2^d} \Xi_j \cdot \hat{\mathbf{a}}_j$ and

  $$\hat{\mathbf{a}}'_i = \hat{\mathbf{V}}_H \cdot \underbrace{\left(\sum_{j \in [k]} \xi_j^{(j-1)2^d} \Xi_j \cdot \hat{\mathbf{a}}_j \right)}_{\hat{\mathbf{b}}_i}.$$

  Since $\hat{\mathbf{V}}_H$ is a Vandermonde matrix, on input $\hat{\mathbf{b}}_i \in \mathbb{F}^{2^d}$, computing $\hat{\mathbf{a}}'_i = \hat{\mathbf{V}}_H \cdot \hat{\mathbf{b}}_i$ with standard radix-2 FFT takes $O(d \cdot 2^d)$ time.

- Naïvely, we can compute $\hat{\mathbf{b}}_i$ in $O(k \cdot 2^d)$ time with brute force since Ξ_i is a diagonal matrix, so computing all of the entries $\mathbf{b} = \left[\hat{\mathbf{b}}_1^\mathsf{T} \mid \ldots \mid \hat{\mathbf{b}}_k^\mathsf{T} \right]^\mathsf{T} \in \mathbb{F}^{k \cdot 2^d}$ takes $O(k^2 \cdot 2^d)$ time. However, we can compute more efficiently as follows. By construction,

  $$\hat{b}_{i,j} = \sum_{\ell \in [k]} \xi_i^{(\ell-1)2^d} \xi_i^{j-1} \hat{a}_{\ell,j}.$$

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

$$\tilde{\mathbf{b}}_j = \mathrm{diag}(\xi_1^{j-1}, \ldots, \xi_k^{j-1}) \cdot \underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \xi_2^{2^d} & \cdots & \xi_2^{2^d(k-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \xi_k^{2^d} & \cdots & \xi_k^{2^d(k-1)} \end{bmatrix}}_{\Xi'} \tilde{\mathbf{a}}_j.$$

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, $k \leq 64$. We show in Fig. 4.3 that the computational cost of the linear PCP prover implemented using this approach represents a small fraction of the overall prover running time in the zkSNARK (i.e., less than 10% of the cost for the parameter settings in Fig. 4.3, and at most 16% across *all* of our experiments).

LPCP Query Computation. The complexity of computing $\mathcal{Q}_{\mathsf{LPCP}}$ is dominated by the complexity of evaluating each A_i, B_i, C_i at the random element τ. We show how to compute A_i efficiently in a general way; a similar approach holds for the B_i and C_i for $i \in [N_w]$.

Recall the previous definition of domain $S \subset \mathbb{F}_{p^2}$ in our approach, where we let $\omega \in \mathbb{F}_{p^2}$ be a primitive 2^{d+1}-th root of unity and let $H = H_1 = \langle \omega \rangle \subset \mathbb{F}_{p^2}$ be the subgroup of order 2^{d+1} generated by ω. Additionally, we let $g \in \mathbb{F}_{p^2}^*$ be the multiplicative generator of the $\mathbb{F}_{p^2}$. We take $\{\xi_1, \ldots, \xi_k\} \subset \mathbb{F}_{p^2}^* \setminus H$, where $\xi_i = g^{2i-2} \in \mathbb{F}_{p^2}^* \setminus H$ for $i \in [k]$, such that $H_i = \xi_i H$ are pairwise disjoint cosets of H. Let $S = \{\alpha_i\}_{i \in [N_g]} = \cup_{i \in [k]} H_i \subset \mathbb{F}_{p^2}$ be the domain for multipoint evaluation interpolation.

Recall the formula for Lagrange interpolation, on the input $z \in \mathbb{F}_{p^2} \setminus S$:

$$A_i(z) := \sum_{j \in [N_g]} a_{j,i} \cdot L_j(z),$$

where

$$L_j(z) := \prod_{\substack{m \in [N_g] \\ j \neq m}} \frac{z - \alpha_m}{\alpha_j - \alpha_m} = \prod_{r \in [k]} L_{j,r}(z),$$

$$L_{j,r}(z) := \prod_{\substack{t \in [2^{d+1}] \\ j \neq (r-1)\cdot 2^{d+1}+t}} \frac{z - \xi_r \cdot \alpha_t}{\alpha_j - \xi_r \cdot \alpha_t} = \begin{cases} \dfrac{z^{2^{d+1}} - \xi_r^{2^{d+1}}}{\xi_{\lceil j/2^{d+1} \rceil}^{2^{d+1}} - \xi_r^{2^{d+1}}} & j \notin [(r-1)\cdot 2^{d+1}+1, r \cdot 2^{d+1}] \\ L'_t(\xi_r^{-1} z) & j \in [(r-1)\cdot 2^{d+1}+1, r \cdot 2^{d+1}] \end{cases},$$

$$L'_t(z) := \prod_{\substack{h \in [2^{d+1}] \\ t \neq h}} \frac{z - \alpha_t}{\alpha_h - \alpha_t}.$$

Since L'_t is the standard radix-2 FFT polynomial interpolation over the subgroup of order 2^{d+1}, we can efficiently evaluate $\{L_j(z)\}_{j \in [N_g]}$ as follows.

1. For $i \in [k]$,

   i. We compute $\{\theta_i\}_{i \in [k]}$, where

   $$\theta_i = \prod_{\substack{r \in [k] \\ r \neq i}} L_{j,r}(z) = \prod_{\substack{r \in [k] \\ r \neq i}} \frac{z^{2^{d+1}} - \xi_r^{2^{d+1}}}{\xi_i^{2^{d+1}} - \xi_r^{2^{d+1}}}.$$

   ii. Evaluate all the radix-2 FFT Lagrange interpolation polynomials $\{L'_t(\cdot)\}_{t \in [2^{d+1}]}$ on $\xi_i^{-1} z$.

   iii. For $t \in [2^{d+1}]$, let $k = (i-1)\cdot 2^{d+1} + t$, and compute $L_k(z) = \theta_i \cdot L'_t(\xi_i^{-1} z)$.

2. Output $\{L_i(z)\}_{i \in [N_g]}$.

Then, after computing $\{L_i(z)\}_{i \in [N_g]}$, computing $\{A_i(z)\}_{i \in [N_g]}$ only requires taking linear combination over $\{L_i(z)\}_{i \in [N_g]}$, which is determined by the coefficients vectors $\{\mathbf{a}_i\}_{i \in [N_g]}$.

LPCP Prover Computation. The complexity of computing $\mathcal{P}_{\mathsf{LPCP}}$ is partly dominated by the computation of the coefficients of the N_g-degree polynomial $H(z)$. We generally follow the footsteps described by [BCG+13]. Since we cannot compute the coefficients of $H(z)$ via a direct use of radix-2 FFTs, we show how to apply our approach as follows.

We begin by introducing the notation for multipoint interpolation and evaluation. Given a domain $D \subseteq \mathbb{F}_{p^2}$ and a polynomial $A(z)$ of degree less than $|D|$, we use $\text{extFFT}_D(A(z))$ to denote the evauated results $\{A(\alpha)\}_{\alpha \in D}$ from the extended radix-2 FFTs over cosets. Similarly, we use $\text{extFFT}_D^{-1}(\{A(\alpha)\}_{\alpha \in D})$ to denote the inversion operation, namely, given $|D|$ points, return the polynomial of degree less than $|D|$ that interpolates between these points on D.

We now decribe how to perform the computation for $\mathcal{P}_{\mathsf{LPCP}}$ in terms of the notations above. Below, we let T be a subset of $\mathbb{F}_{p^2}$ with $|T| = N_g$ and $S \cap T = \varnothing$. Let $T = gS$ in our setting, and we denote $T = \{\beta_i\}_{i\in[N_w]}$, so that $T = \cup_{i\in[k]}(g \cdot H_i) \subset \mathbb{F}_{p^2}$, where $g \cdot H_i$ and H_i are pairwise disjoint.

1. Sample $\delta_1, \delta_2, \delta_3 \xleftarrow{\text{R}} \mathbb{F}_{p^2}$.

2. For $j \in [N_g]$, compute

$$\begin{aligned} A'(\alpha_j) &:= A_0(\alpha_j) + \sum_{i\in[N_w]} w_i A_i(\alpha_j) = a_{j,0} + \sum_{i\in[N_w]} w_i a_{j,i} \\ B'(\alpha_j) &:= B_0(\alpha_j) + \sum_{i\in[N_w]} w_i B_i(\alpha_j) = b_{j,0} + \sum_{i\in[N_w]} w_i b_{j,i} \\ C'(\alpha_j) &:= C_0(\alpha_j) + \sum_{i\in[N_w]} w_i C_i(\alpha_j) = c_{j,0} + \sum_{i\in[N_w]} w_i c_{j,i} \end{aligned}$$

3. Compute the coefficients for $A'(z)$ by invoking $\text{extFFT}_S^{-1}(\{A'(\alpha_j)\}_{j\in[N_g]})$.
   Compute the coefficients for $B'(z)$ by invoking $\text{extFFT}_S^{-1}(\{B'(\alpha_j)\}_{j\in[N_g]})$.
   Compute the coefficients for $C'(z)$ by invoking $\text{extFFT}_S^{-1}(\{C'(\alpha_j)\}_{j\in[N_g]})$.

4. Compute the pointwise evaluation $\{A'(\beta_i)\}_{i\in[N_g]}$ over T by invoking $\text{extFFT}_T(A'(z))$.
   Compute the pointwise evaluation $\{B'(\beta_i)\}_{i\in[N_g]}$ over T by invoking $\text{extFFT}_T(B'(z))$.
   Compute the pointwise evaluation $\{C'(\beta_i)\}_{i\in[N_g]}$ over T by invoking $\text{extFFT}_T(C'(z))$.

5. Compute the pointwise evaluation $\{H'(\beta_i)\}_{i\in[N_g]}$ over T for $H'(z) = (A'(z)B'(z) - C'(z))/Z_S(z)$.

6. Compute the coefficients for $H'(z)$ by invoking $\text{extFFT}_T^{-1}(\{H'(\beta_i)\}_{i\in[N_g]})$.

7. Compute the N_g coefficients of polynomial $H(z) = H'(z) + \delta_1 B'(z) + \delta_2 A'(z) + \delta_1\delta_2 Z_S(z) - \delta_3$.

8. Output the N_g coefficients of polynomial $H(z)$.

Choosing T to be the multiplicative coset of S simplifies the computation in Step 3, 4, 5, 6 as follows:

- Since $Z_S(z) = \prod_{i\in[k]}\left((g\xi_j)^{2^{d+1}} - \xi_i^{2^{d+1}}\right)$ everywhere on gH_j for $j \in [k]$, evaluating $Z_S(z)$ over T for $\{Z_S(\beta_i)\}_{i\in[N_g]}$ in step 5 requires $O(dk^2)$ field operations.

- For extFFT_T in step 4, extFFT_T^{-1} in step 6, and extFFT_S^{-1} in step 2, all of them require $O(N_g \log N_g)$ field operations. Moreover, letting $\Xi_g = \text{diag}(1, g, g^2, \ldots, g^{N_g-1})$, it holds that

$$\begin{aligned} \text{extFFT}_T(\cdot) &= (\text{extFFT}_S \circ \Xi_g)(\cdot) \\ \text{extFFT}_T^{-1}(\cdot) &= (\Xi_g^{-1} \circ \text{extFFT}_S^{-1})(\cdot). \end{aligned}$$

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 the 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 $\mathbb{F}_{p^2}$ $(d = 2)$ of 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 (Theorem 2.10 and Remark 2.12). The query length of the linear PCP is chosen to achieve knowledge error at most 2^{-128}. For a linear PCP over $\mathbb{F}_{p^2}$, the linear PCP from Theorem 2.10 has knowledge error $2N_g/(p^2 - N_g)$. This is repeated ρ times to amplify soundness (ρ is chosen such that $(2N_g/(p^2 - N_g))^\rho \le 2^{-128}$). The 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 3.17: namely, we choose τ to be the smallest value where $p^{-\tau d} \le 2^{-128}$.
- The noise smudging bound B is chosen so that $\varepsilon = 2^{-\kappa}$ in Eq. (3.3.6) 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 module dimension $n < 2^{12}$ (the module dimension will be set (later) based on the security parameter and the modulus q, and in all of our cases, $n < 2^{12}$).

  In Construction 3.22, the number of homomorphic operations the prover performs is equal to the length k of the linear PCP query. From Theorem 2.10, 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 take 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 set the bounds $B_1 = dkp$ and $B_2 = \sqrt{dk}p$. In Remark 3.20, 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. (3.3.6) is essentially determined by $\gamma_R B_2 Cs + \gamma_R B_1/2 + 2\gamma_R n C^2 s^2$. 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 allow 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. (3.3.2). In all the cases we consider, $q \leq 2^{128}$, 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 critical to reducing the prover cost.

- Given the modulus q and noise rate s for the discrete Gaussian distribution, we use the LWE Estimator[2] by Albrecht et al. [APS15] to determine the smallest module 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 $2n$.

- As is discussed previously, applying modulus switching (Theorem 3.21) to Construction 3.11 reduces the size of the modulus q. In Table 4.1, we show that this technique reduces q by a facor of $2.5\times$ to $2.7\times$, which translates to a same factor of reduction in the concrete proof size of our zkSNARK construction.

We provide some example parameters we use in Table 4.1. 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].

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 typical parameter settings, the module 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 $\mathsf{ct}_1 = (\mathbf{a}_1, \mathbf{c}_1), \ldots, \mathbf{c}_N = (\mathbf{a}_N, \mathbf{c}_N)$ 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

[2] 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	$2^{13} - 1$	$(3585, 1)$	97	35	66	208	10
$\mathbb{F}_{p^2}$, $\mathbb{F}_{p^2}$	128	138	$2^{13} - 1$	$(1815, 2)$	98	35	64	104	5

[*]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 4.1: Lattice parameters for zkSNARK instantiations obtained by combining the linear PCP Π_{LPCP} from Theorem 2.10 and Remark 2.12 with the linear-only vector encryption scheme Π_{Enc} from Construction 3.11. Here, λ_q is the estimated bits of quantum security, λ_c is the estimated bits of classical security, p is the plaintext modulus, d is the module rank, n is the module dimension, q is the ciphertext modulus, 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 and for the smallest plaintext fields we consider in our evaluation.

components of lattice-based cryptosystems has been used for both lattice-based key-exchange [BCD+16] as well as previous lattice-based zkSNARKs [GMNO18].

In our implementation, $q \approx 2^{100}$ and each R_q^n element can be represented by roughly 50 KB (see Table 4.1 for one specific set 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 for concrete efficiency.

Noise distribution. We take our noise distribution χ to be a discrete Gaussian distribution $\chi = \chi_s$ with mean 0 and width s (Eq. (2.0.1)). 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 just compute over $\mathbb{Z}_{2^{128}}$ and defer the modular reduction to the end of the computation. Moreover, the modular reduction just corresponds to dropping the $(128 - \log q)$ most significant bits.

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., 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 4.2) indicate

	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 4.2: 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.

that using the compiler intrinsic representation for 128-bit arithmetic is over $16\times$ 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 8-9$\times$) 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. 4.3 and taking into consideration these microbenchmarks for elementary arithmetic operations, using a modulus even slightly larger than 2^{128} would increase the prover cost by a factor of $2\times$ to $3\times$. [3] 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 $8\times$ to $9\times$ based on our microbenchmarks.

4.3 Experiment Evaluation

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

System implementation. We implemented our construction in `C++`. We use `libsnark` [Lab21c] and `libfqfft` [Lab21a] to implement the linear PCP for R1CS satisfiability (Theorem 2.10). In particular, we use the linear PCP implementation from `libsnark` (with the minor changes from Section 2.3), 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.

[3] 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.

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.

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 in single-threaded setting.

General benchmarks. In Fig. 4.1, 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 setting 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 soundness 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 $2.8\times$ (with a corresponding decrease in setup time), but increases the proof size by $1.2\times$. The prover complexity is essentially the same in the two cases.

Turning to 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 $\mathbb{F}_p$, we see that 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–$1.5\times$ longer than the constructions that use the extension field.

Extension field vs. base field. To highlight the concrete performance improvement enabled by extension fields, we compare our zkSNARKs over $\mathbb{F}_{p^2}$ with an instantiation over $\mathbb{F}_p$ (i.e., compile the linear PCP from Theorem 2.10 over $\mathbb{F}_p$ using Construction 3.11 over $\mathbb{F}_p$). The results are summarized in Fig. 4.2. 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

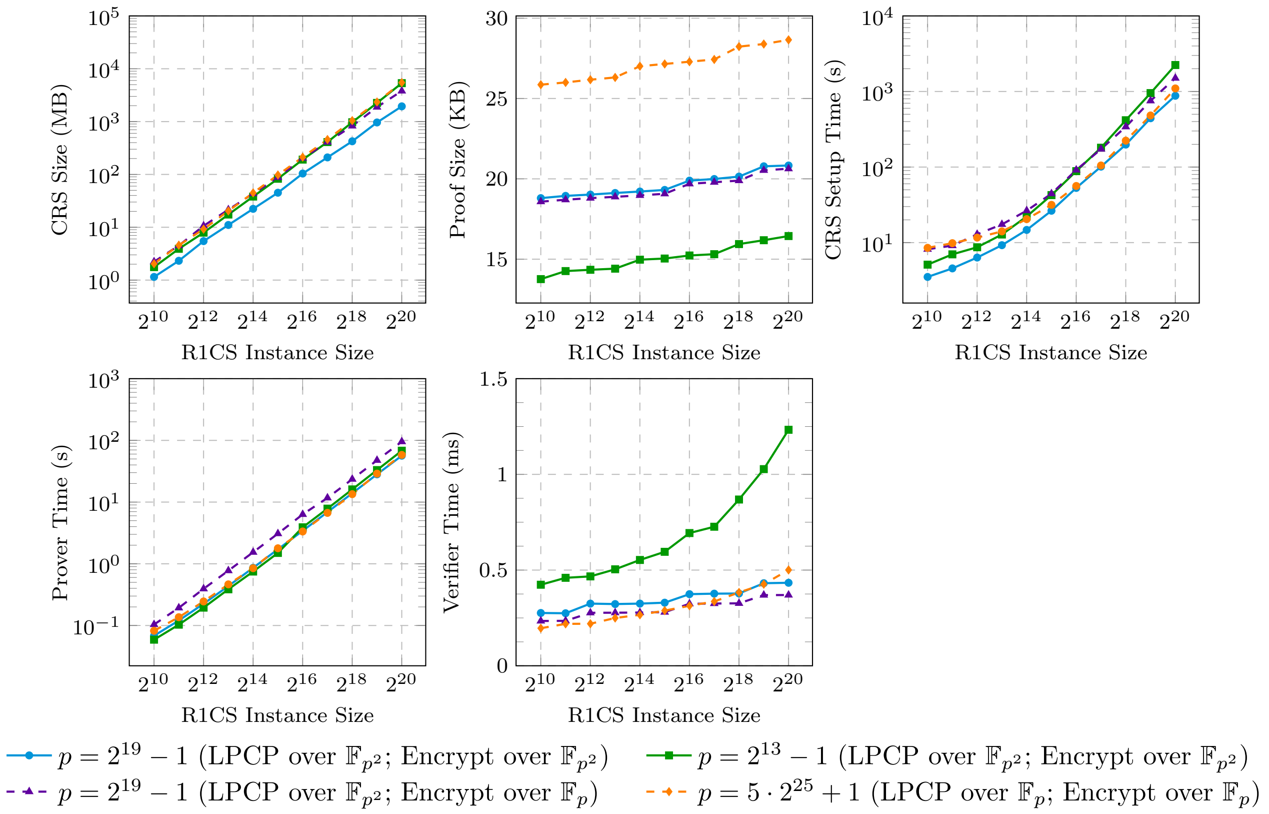

Figure 4.1: Performance comparison for different instantiations of our scheme for supporting R1CS instances of different sizes. The solid lines correspond to our primary instantiations using a linear PCP over $\mathbb{F}_{p^2}$ in conjunction with vector encryption over $\mathbb{F}_{p^2}$. The dashed 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 apply Construction 3.1 to first 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 $\mathbb{F}_p$.

Section 4.2 (and Table 4.2), this will incur considerable computational overhead for the big-integer arithmetic. Even disregarding the added computational overhead, we see that for *every* choice of field size, working over an extension field reduces *both* the CRS size and the proof size. Compared to the instantiation over $\mathbb{F}_p$, the CRS is 1.2× to 1.4× shorter and the proof is 1.5× to 2× shorter. Compared to an instantiation where we rewrite elements of $\mathbb{F}_p$ as two digits of magnitude $\sqrt{p}$ (Remark 3.20), the CRS is 2.2× to 2.4× shorter and the proof is 1.2× to 1.5× shorter.

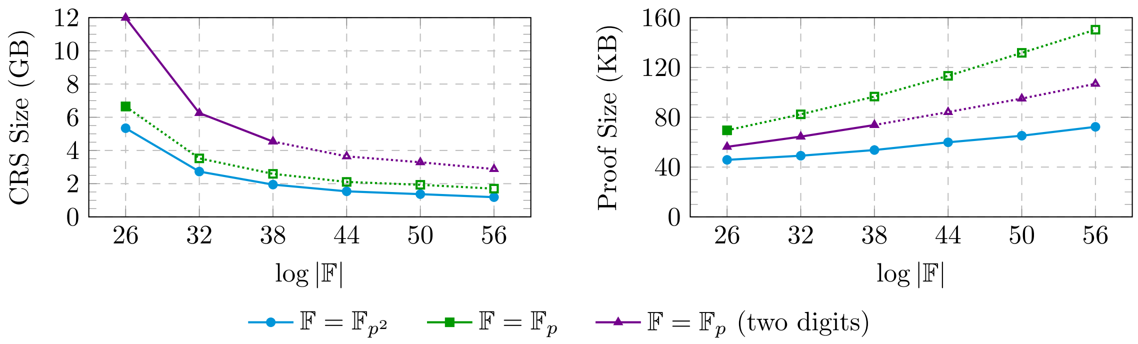

Figure 4.2: CRS size and proof size as a function of the field size $|\mathbb{F}|$, where $\mathbb{F}$ is either a quadratic extension $\mathbb{F}_{p^2}$ or a base field $\mathbb{F}_p$. The characteristic p is chosen so $\mathbb{F}$ has the prescribed size. Parameters based on instantiating Construction 3.22 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 3.20). Elements with a non-filled marker (and a dotted line) denote parameter settings where the modulus q exceeds 128 bits.

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. 4.3 (the breakdowns for other parameters 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} constraint, linear PCP query encryption constitutes 99% of the CRS generation time.

For the prover computation, we consider the cost of the FFT (to compute QAP coefficients), the time spent on CRS expansion (i.e., deriving the random ciphertext components $\mathbf{a} \in \mathbb{R}_q^n$ from the PRF key), and the cost of the homomorphic operations for computing the encrypted response. The microbenchmarks show that the majority (between 30% and 45%) of the time is spent on CRS expansion. For an instance of size 2^{20}, the expanded CRS is around 90 GB, and CRS expansion takes about 28s (note that we generate the vectors on an as needed basis and do *not* need to store the full CRS in memory). Here, we can consider a time-memory trade-off where part of the CRS is stored in memory and the rest dynamically derived from the PRF. This can reduce the concrete prover costs by over 2×, but at the expense of needing significantly more memory. The homomorphic operations represent about 28% of the prover cost and for the larger instances, the FFTs represent about 30%. 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 $\mathbb{F}_{p^2}$ cannot directly use

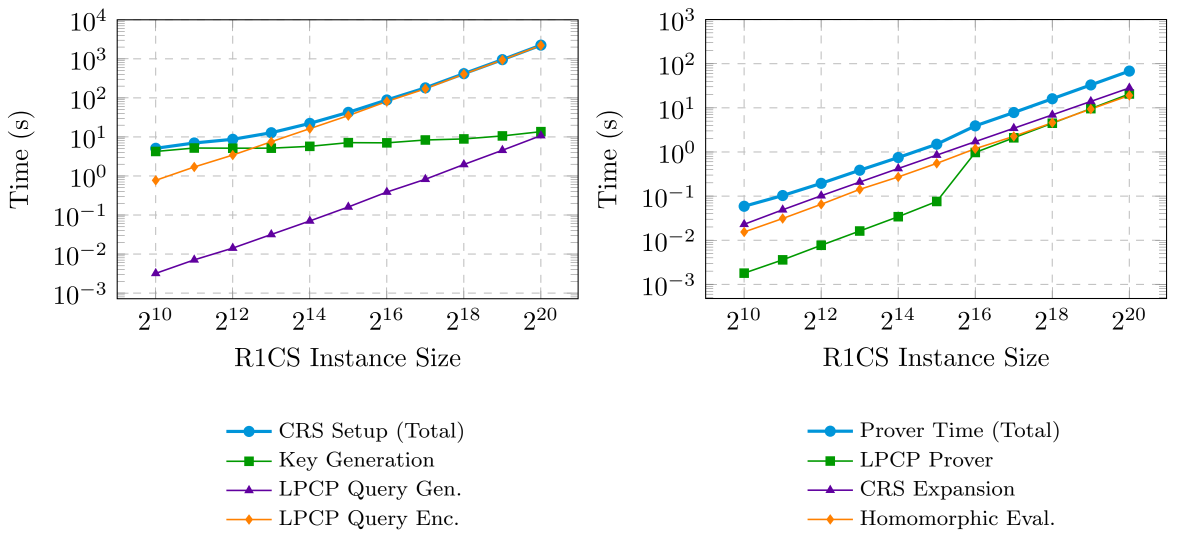

Figure 4.3: Cost breakdowns for CRS setup and prover for different R1CS instances. Measurements are based on instantiating Construction 3.22 with a linear PCP and a vector encryption scheme over $\mathbb{F}_{p^2}$ where $p = 2^{13} - 1$.

standard power-of-two FFTs because of lacking in primitive roots of unity). By extrapolating the performance, our approach is about 7× slower than the basic radix-2 FFT. [4] When consider an R1CS system with 2^{20} constraint over $\mathbb{F}_{p^2}$ where $p = 2^{19} - 1$ (where there are sufficient roots of unity to invoke standard FFTs), the FFTs, homomorphic operations, and CRS expansion account for 6% (3.2 s), 38% (21.4 s), and 56% (31.7 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. 4.4. In particular, when we work over $\mathbb{F}_{p^2}$ with $p = 2^{19} - 1$, and consider the setting *without* zero-knowledge (i.e., setting $\kappa = 0$), the prover time (for an R1CS instance of size 2^{20}) is just 34 s, which is 2.3× *faster* than the pairing-based construction of Groth et al. [Gro16]; the proof size is 11.1 KB, which is 89× longer than the construction of Groth. Working over a smaller base field, we can bring the proof size down to 8 KB, which is around 20× shorter than other post-quantum candidates. This 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 7% reduction in proof size. Realizing post-quantum security requires using a larger module 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

[4]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, but directly inline the multipoint evaluation and interpolation on two points. For instances larger than 2^{15}, we use the general Bostan-Schost algorithms [BS05], which introduces the 7× overhead.

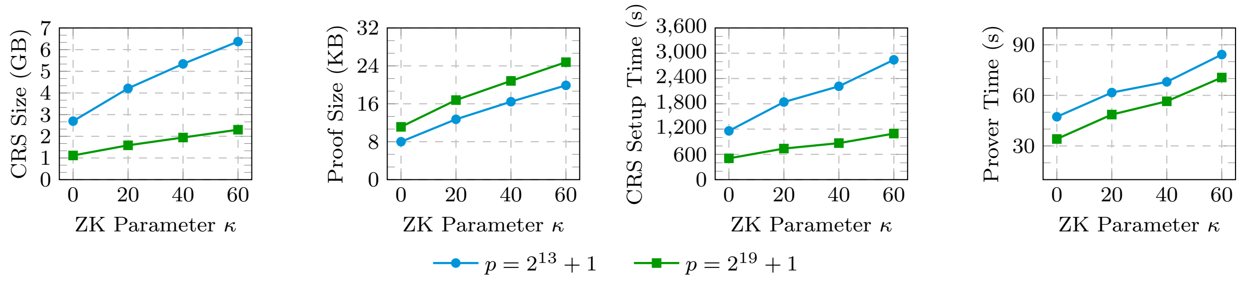

Figure 4.4: Cost breakdowns as a function of the zero-knowledge parameter κ (i.e., the zero-knowledge distinguishing advantage of any $\mathsf{poly}(\lambda)$ adversary is bounded by $2^{-\kappa} + \mathsf{negl}(\lambda)$). All measurements taken for an R1CS instance over $\mathbb{F}_{p^2}$ with 2^{20} constraints (and compiled using vector encryption over $\mathbb{F}_{p^2}$).

a PRF). We provide more details in Table 4.3.

p	Setting	Size		Time	
		CRS	**Proof**	**Setup**	**Prover**
$2^{13} - 1$	PQ	5.3 GB	16.4 KB	2215 s	68 s
	Classical	5.3 GB	15.2 KB	2225 s	69 s
$2^{19} - 1$	PQ	1.9 GB	20.8 KB	877 s	56 s
	Classical	1.9 GB	19.2 KB	865 s	56 s

Table 4.3: Performance comparison of zkSNARKs instantiated using parameters for 128-bits of classical vs. 128-bits of post-quantum security (denoted "PQ"). For all measurements, we consider R1CS instances over $\mathbb{F}_{p^2}$ with 2^{20} constraints and compile using linear-only vector encryption over $\mathbb{F}_{p^2}$.

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: 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` [Lab21c] for the implementation of Groth's pairing-based construction [Gro16] and `libiop` [Lab21b] 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 1.1 and refer to Chapter 1 for further discussion.

Chapter 5

Conclusion

The work described in this thesis presents a new construction of lattice-based zkSNARK in the designated-verifier preprocessing model, with a focus on shortening the concrete proof size. To briefly recall, our construction followed the blueprint of Bitansky et al. [BCI$^+$13] and Boneh et al. [BISW17]. We developed a concretely-efficient lattice-based zkSNARK over $\mathbb{F}_{p^2}$ by combining a linear PCP with a linear-only vector encryption scheme. We started by observing that the size of the modulus q scales with the plaintext modulus p rather than the field size. By working over quadratic extension fields, we achieved a smaller q size, which translates to a smaller proof size. We provided two instantiation by directly working over $\mathbb{F}_{p^2}$ or compiling the underlying linear PCP over $\mathbb{F}_{p^2}$ to $\mathbb{F}_p$ and work over base field $\mathbb{F}_p$. Relying on the linearly-only vector encryption introduced by Boneh et al. [BISW17] and Peikert et al. [PVW08], we reduced the cost of expanding the lattice ciphertexts, sparsified the valid ciphertexts, and enabled the linear PCP soundness amplification with additive overhead rather than multiplicative one. We achieved zero-knowledge property from circuit privacy by relying on the MLWE assumption (Definition 3.9) and the smudging lemma (Lemma 2.1). Moreover, we used the modulus switching technique (Theorem 3.21) to decrease the size of q, which shorten the concrete proof size by more than a factor of $2\times$. Finally, we implemented all instantiations of our constructions and experiments with `C++` using 128-bit arithmetic on a 64-bit architecture. We now propose some directions for future research as follows:

Publicly-Verifiable zkSNARKs from Lattices. A limitation of our construction is that it does not provide public verifiability, while our construction at the same time fail to imply a reusable soudness. As is discussed in Remark 3.25, while the strong soundness property holds for our the underlying linear PCP, our encryption scheme does not sastisfy the *interactive* linear-only assumption proposed by Bitansky et al. [BCI$^+$13]. However, a publicly-verifiable zkSNARK is automatically guaranteed with multi-theorem soundness. Moreover, publicly-verifiable zkSNARKs have a wider range of real world application (e.g., blockchain, digital currency) than the designated-verifier ones. Designated-verifier zkSNARK is more focusing on the verifiable computation and computation oursourcing. Constructing publicly-verifiable zkSNARKs from standard lattice assumption still remains to be an open problem, and it would be interesting to see follow up works focusing on a concretely-efficient publicly-verifiable lattice-based zkSNARK.

Appendix A

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 A.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 $x_1, \ldots, x_k \in \mathbb{F}$, and all scalars $\alpha_1, \ldots, \alpha_k \in \mathbb{F}^d$,

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

where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{enc}_i \leftarrow \mathsf{Encode}(\mathsf{sk}, x_i)$ for all $i \in [k]$, and S denotes the support of $\mathsf{Encode}(\mathsf{sk}, \sum_{i \in [k]} \alpha_i x_i)$.

Definition A.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 $\mathcal{A}$,

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

where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $s \xleftarrow{\text{R}} \mathbb{F}$, $\mathsf{enc}_i \leftarrow \mathsf{Encode}(\mathsf{sk}, s^i)$ for all $i \in \{0, \dots, 2q\}$, $\sigma \leftarrow (\mathsf{pk}, \mathsf{enc}_0, \dots, \mathsf{enc}_q, \mathsf{enc}_{q+2}, \dots, \mathsf{enc}_{2q})$, and S is the set of encodings in the support of $\mathsf{Encode}(\mathsf{sk}, s^{q+1})$.

Lemma A.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 $z_1, \dots, z_{2q} \in \mathbb{F}$, 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 $(\mathsf{pk}, \mathsf{enc}_0, \dots, \mathsf{enc}_q, \mathsf{enc}_{q+2}, \dots, \mathsf{enc}_{2q})$ 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 $s \in \{z_1, \dots, z_{2q}\}$, then this quantity is exactly s^{q+1}. Since s is uniform and independent of $z_1, \dots, z_{2q}$, the probability that $s \in \{z_1, \dots, z_{2q}\}$ is exactly $2q / |\mathbb{F}|$, which proves the claim. $\square$

Remark A.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 A.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 A.3 shows that the q-PDH assumption is false 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 A.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 15 bits of soundness at best. To obtain 128-bits of soundness, they would either need to apply soundness amplification (which increases all parameters by a factor of $128/15 \approx 8.5$) or instantiate their Regev-based encoding scheme over a super-polynomial size field (which would also incur significant 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 2.12). 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 significantly more efficient construction over *small* fields compared to the Gennaro et al. construction (see Table 1.1).

Bibliography

[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.

[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.

[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.

[BSCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *TCC*, pages 31–60, 2016.

[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 *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, 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.

[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.

[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.

[Lab21a] SCIPR Lab. libfqfft: C++ library for FFTs in finite fields. https://github.com/scipr-lab/libfqfft/, 2021.

[Lab21b] SCIPR Lab. libiop: a c++ library for iop-based zksnarks. https://github.com/scipr-lab/libiop, 2021.

[Lab21c] SCIPR Lab. libsnark: a c++ library for zkSNARK proofs. https://github.com/scipr-lab/libsnark/, 2021.

[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, 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.

[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.

[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.

[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.