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

Abstract iii  
Acknowledgements vi  

1 Introduction 1  
1.1 Background 2  
1.2 Technical Overview 4  
1.3 Contributions 7  
1.4 Related Work 9  
1.5 Works Contained in this Thesis 10  

2 Preliminaries 12  
2.1 Circuit Satisfiability Problems 14  
2.2 Succinct Non-Interactive Arguments 14  
2.3 Linear PCPs 16  

3 Lattice-Based Succint Arguments 23  
3.1 Linear PCPs over Extension Fields 23  
3.2 Linear-Only Vector Encryption 24  
3.3 Candidate Linear-Only Vector Encryption 27  
3.4 zkSNARKs from Linear-Only Encryption 37  

4 Implementation and Evaluation 43  
4.1 Linear PCP Implementation 43  
4.2 Lattice-Based zkSNARK Implementation 48  
4.3 Experiment Evaluation 51  

5 Conclusion 57  

A The Power Diffie-Hellman Assumption over Small Fields 59  

Bibliography 61
List of Tables

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 (○).

1.2 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 poly($\lambda$) factors (where $\lambda$ 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.
4.1 Lattice parameters for zkSNARK instantiations obtained by combining the linear PCP $\Pi_{lPCP}$ from Theorem 2.10 and Remark 2.12 with the linear-only vector encryption scheme $\Pi_{Enc}$ from Construction 3.11. Here, $\lambda_q$ is the estimated bits of quantum security, $\lambda_c$ 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, $\ell$ is the dimension of the plaintext space, and $\tau$ 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.

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.

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 $\mathbb{F}_{p^2}$.
List of Figures

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 $\mathbb{F}_p$. ................................. 53

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. ................................. 54

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 $p = 2^{13} - 1$. ................................. 55

4.4 Cost breakdowns as a function of the zero-knowledge parameter $\kappa$ (i.e., the zero-knowledge distinguishing advantage of any $\text{poly}(\lambda)$ adversary is bounded by $2^{-\kappa} + \text{negl}(\lambda)$). All measurements taken for an R1CS instance over $\mathbb{F}_{p^2}$ with $2^{20}$ constraints (and compiled using vector encryption over $\mathbb{F}_{p^2}$). ................................. 56
Chapter 1

Introduction

On a high level, a proof system for a language $L \subseteq \{0, 1\}^*$ is a two-party protocol between a prover $P$ and a verifier $V$. The goal of the prover is to convince the verifier of some statement $x \in \{0, 1\}^*$ is a member of the language $L$ (namely $x \in L$). 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 $x \in L$, and soundness, which roughly says a plausibly honest prover will not be able to prove a false statement $x \notin L$ 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 $L \subseteq \{0, 1\}^*$ is in NP, there exists a polynomial time algorithm $R$ (referred as an NP relation) such that for every $x \in \{0, 1\}^*$,

$$x \in L \iff \exists w \in \{0, 1\}^{\text{poly}(|x|)} : R(x, w) = 1.$$  

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

For an NP relation $R$, a zero-knowledge proof of knowledge [GMR85] enables a prover to convince a verifier of $x \in L$ without revealing anything about the NP witness $w$. 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 $\pi$ from prover to verifier, while the length of $\pi$ and the verification complexity are sublinear (ideally, polylogarithmic) in the size of the circuit computing $R$. 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 $L$ is defined by a linear oracle $\pi : \mathbb{F}^\ell \to \mathbb{F}$ over a finite field $\mathbb{F}$. On input a statement $x$, a verifier can submit a query matrix $Q \in \mathbb{F}^{\ell \times k}$ to the oracle and obtain the responses $a \leftarrow Q^T \pi \in \mathbb{F}^k$. Based on the responses, the verifier decides whether to accept or reject. We refer to $k$ as the number of queries and $\ell$ as the query length of the linear PCP. The linear PCP is sound if for a false statement $x \notin L$ and any proof vector $\pi \in \mathbb{F}^\ell$, the probability that the verifier accepts is negligible (where the probability is taken over the sampling of $Q$). 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 $L$ from a linear PCP for $L$, the Bitansky et al. compiler encrypts the linear PCP queries (i.e., the entries of $Q$) 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 $a = Q^T \pi$. 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 [BHII+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 \((a, c)\) where \( a \in \mathbb{Z}_n^q \) and \( c = s^T a + pe + x \in \mathbb{Z}_q \), where \( s \in \mathbb{Z}_n^q \) 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 \((a_1, c_1)\) and \((a_2, c_2)\) encrypt values \( x_1, x_2 \), respectively, then \((a_1 + a_2, c_1 + c_2)\) encrypts the value \( x_1 + x_2 \mod p \), 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 \( 2^\kappa \) where \( \kappa \) 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 zkSNARK to 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 \(F\)) supports encrypting a vector of field elements. Instead of encrypting each entry in the linear PCP query matrix \(Q \in F^{\ell \times k}\) separately, the Boneh et al. compiler encrypts rows of \(Q\). 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 \(\ell\) \(\mathbb{Z}_p\)-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 ciphertext 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 $F_p$ 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 $F_{p^k}$. We develop two instantiations of this approach:

- **Compile linear PCPs over $F_{p^k}$ to $F_p$:** Our first instantiation shows how to compile a linear PCP over $F_{p^k}$ to a zkSNARK using linear-only vector encryption over the base field $F_p$ (i.e., the same encryption scheme from above). We first show how to transform a linear PCP over $F_{p^k}$ to a linear PCP over $F_p$. 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 $F_{p^k}$-operations and linear transformations over the vector space $F_p^k$. 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. 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 $R_p \cong F_{p^2}$ when $p \equiv 3 \pmod{4}$. This gives a direct compilation from a linear PCP over a quadratic extension $F_{p^2}$ to a zkSNARK over $F_{p^2}$, 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 $F_{p^2}$ to $F_p$). We show the concrete performance in Table 1.1 and in Section 4.3. Leveraging encryption

---

2Even 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 \( \mathbb{F}_p^* \) 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 choice 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 $10.3 \times$ 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 $13.1 \times$ shorter than Fractal [COS20]. Compared to the lattice-based construction of Gennaro et al. [GMNO18], our zkSNARKs are $39.4 \times$ shorter. However, there remains a large gap ($128 \times$) 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 $4.6–6.3 \times$ 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 $60.2 \times$ faster for the prover and $5.1 \times$ faster for the verifier. Moreover, our main constructions (with provable zero-knowledge) are $1.1–1.6 \times$ faster for the prover and about $2.7–7.9 \times$ 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 $2.3 \times$ 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 $1.8 \times$ more expensive, but our proof sizes and verification times are over $874 \times$ 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).
### 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.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Structure</th>
<th>PQ</th>
<th>TP</th>
<th>PV</th>
<th>R1CS Size</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2(^{16})</td>
<td></td>
</tr>
<tr>
<td>[Gro16]</td>
<td>Pairings</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>12.4 MB</td>
<td>5.6 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>199 MB</td>
<td>72.0 s</td>
</tr>
<tr>
<td>[GMNO18] *</td>
<td>Lattices</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>17.3 MB(^{1})</td>
<td>167 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>199 MB</td>
<td>72.0 s</td>
</tr>
<tr>
<td>Ligero [AHIV17]</td>
<td>Random</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>4.3 MB</td>
<td>2.5 s</td>
</tr>
<tr>
<td></td>
<td>Oracle</td>
<td></td>
<td></td>
<td></td>
<td>14 MB</td>
<td>37.6 s</td>
</tr>
<tr>
<td>Aurora [BCR(^+)19]</td>
<td>Random</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>121 KB</td>
<td>17.8 s</td>
</tr>
<tr>
<td></td>
<td>Oracle</td>
<td></td>
<td></td>
<td></td>
<td>169 KB</td>
<td>304 s</td>
</tr>
<tr>
<td>Fractal [COS20](^‡)</td>
<td>Random</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>1.4 GB</td>
<td>11.6 s</td>
</tr>
<tr>
<td></td>
<td>Oracle</td>
<td></td>
<td></td>
<td></td>
<td>178 KB</td>
<td>21.3 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11 GB</td>
<td>115.8 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>215 KB</td>
<td>184.3 s</td>
</tr>
<tr>
<td>This work</td>
<td>Lattices</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>191 MB</td>
<td>88.3 s</td>
</tr>
<tr>
<td>(Shorter Proofs)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15.2 KB</td>
<td>3.9 s</td>
</tr>
<tr>
<td>This work</td>
<td>Lattices</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>104 MB</td>
<td>53.0 s</td>
</tr>
<tr>
<td>(Shorter CRS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>19.9 KB</td>
<td>3.4 s</td>
</tr>
</tbody>
</table>

This work Lattices 2\(^{16}\) 191 MB 15.2 KB 88.3 s 3.9 s 0.69 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\(\times\). 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.

\(^\dagger\)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\(^{15}\).

\(^\dagger\)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 \(F_{p^2}\) where \(p = 2^{13} - 1\) and the “Shorter CRS” setting operates over the field \(F_{p^2}\) 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, 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, 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 $\text{polylog}(N)$. 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 $\text{polylog}(N)$. 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].
### CHAPTER 1. INTRODUCTION

<table>
<thead>
<tr>
<th>PQ</th>
<th>TP</th>
<th>PV</th>
<th>Proof Size</th>
<th>Runtime</th>
<th>Crypto. Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asymptotic</td>
<td>Concrete</td>
<td>Prover</td>
<td>Verifier</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Construction</th>
<th>PQ</th>
<th>TP</th>
<th>PV</th>
<th>Proof Size</th>
<th>Runtime</th>
<th>Crypto. Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Groth [Gro16]</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>1</td>
<td>$128$ B</td>
<td>$N \log N$</td>
</tr>
<tr>
<td>Marlin [CHM+20]</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>1</td>
<td>$704$ B</td>
<td>$N \log N$</td>
</tr>
<tr>
<td>Sonic [MBKM19]</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>1</td>
<td>$1.1$ KB</td>
<td>$N \log N$</td>
</tr>
<tr>
<td>Xiphos [SL20]</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>$\log N$</td>
<td>$61$ KB</td>
<td>$N$</td>
</tr>
<tr>
<td>Spartan [Set20]</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>$\sqrt{N}$</td>
<td>$142$ KB</td>
<td>$N$</td>
</tr>
<tr>
<td>Fractal [COS20]</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>$\log^2 N$</td>
<td>$215$ KB$^\dagger$</td>
<td>$N \log N$</td>
</tr>
<tr>
<td>[GMNO18]$^*$</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>$1$</td>
<td>$640$ KB$^\dagger$</td>
<td>$N \log N$</td>
</tr>
<tr>
<td>STARK [BBHR18b]</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>$\log^2 N$</td>
<td>$3.2$ MB</td>
<td>$N \text{polylog}(N)$</td>
</tr>
</tbody>
</table>

$^*$For the asymptotic estimates for the lattice-based constructions, we consider an instantiation over a field of size $2^{2\lambda(\lambda)}$ (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 $F$, the lattice-based parameters scale with $\text{polylog}(|F|)$.

$^\dagger$Proof sizes for Fractal measured using the implementation from $\text{libiop}$ [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.

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

<table>
<thead>
<tr>
<th>Proof Size</th>
<th>Runtime</th>
<th>Crypto. Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N \log N$</td>
<td>$</td>
<td>x</td>
</tr>
<tr>
<td>$N \log N$</td>
<td>$</td>
<td>x</td>
</tr>
<tr>
<td>$N \log N$</td>
<td>$</td>
<td>x</td>
</tr>
<tr>
<td>$N$</td>
<td>$</td>
<td>x</td>
</tr>
<tr>
<td>$\sqrt{N}$</td>
<td>$</td>
<td>x</td>
</tr>
<tr>
<td>$215$ KB$^\dagger$</td>
<td>$</td>
<td>x</td>
</tr>
<tr>
<td>$640$ KB$^\dagger$</td>
<td>$</td>
<td>x</td>
</tr>
<tr>
<td>$3.2$ MB</td>
<td>$</td>
<td>x</td>
</tr>
</tbody>
</table>

| This work$^*$ | O | O | O | $1$ | $16$ KB | $N \log N$ | $|x|$ | Lattices |

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 $\text{poly}(\lambda)$ factors (where $\lambda$ 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 (O) or only classically secure (O). The “TP” column specifies whether the scheme is transparent (O), relies on a trusted setup for a universal CRS (O), or relies on a trusted sampling of a language-dependent CRS (O). The “PV” column specifies whether the argument is publicly-verifiable (O) or designated-verifier (O). 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 \( S \), we write \( x \overset{\$}{\leftarrow} S \) to denote that \( x \) is sampled uniformly at random from \( S \). For a distribution \( D \), we write \( x \leftarrow D \) to denote that \( x \) is sampled from \( D \).

Throughout this work, we use \( \lambda \) to denote a computational security parameter and \( \kappa \) 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 \( f(\lambda) = \text{negl}(\lambda) \). We use \( \text{poly}(\lambda) \) to denote a quantity that is upper-bounded by a fixed polynomial on input \( \lambda \).

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 \( A \) is efficient if it runs in probabilistic polynomial time at the length of the input. We denote \( A(x; r) \) the output of \( A \) on the input of \( x \) and randomness \( r \). In typical cases where we do not specify the randomness explicitly, we use \( A(x) \) to denote the output distribution of \( A \) on the input of \( x \), where the randomness is drawn from the uniform distribution.

We say two families of distributions \( \mathcal{D}_1 = \{D_1, \lambda\}_{\lambda \in \mathbb{N}} \) and \( \mathcal{D}_2 = \{D_2, \lambda\}_{\lambda \in \mathbb{N}} \) 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 \( \text{negl}(\kappa) \).

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 \( \ell_p \) 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 \( \forall s, t \in \mathbb{F}_{p^d}, \mathbf{M}_s \mathbf{v}_t = \mathbf{v}_{st} \).

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 \( \ell_\infty \)-norm of the vector of coefficient corresponding to \( r \). For a vector \( r \in R^n \), we write \( \|r\|_p \) to denote the \( \ell_p \)-norm of the vector of coefficient \( \mathbf{v}_r \in \mathbb{Z}^{dn} \) formed by concatenating \( \mathbf{v}_{r_i} \in \mathbb{Z}^d \) for each \( i \in [n] \). We write \( \gamma_R \) 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

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

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 \( \|s\|_2 \) where \( s = (s_1, \ldots, s_m) \).

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 \( e_2 \sim \mathbb{U} [-B', B'] \). 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 \( S \subset \mathbb{F} \),

\[
\Pr[f(\alpha_1, \ldots, \alpha_n) = 0 \mid \alpha_1, \ldots, \alpha_n \leftarrow \mathbb{F}] \leq \frac{d}{|S|}.
\]


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 [SBV13, BCG13, BCR19].

Definition 2.3 (Rank-1 Constraint Satisfiability [SBV13, BCG13, GGPR13, BCR19]). A rank-1 constraint satisfiability (R1CS) system over a finite field $\mathbb{F}$ is specified by a tuple $S = (n, N_g, N_w, \{a_i, b_i, c_i\}_{i \in [N_g]})$ where $n, N_g, N_w \in \mathbb{N}, n \leq N_w$, and $a_i, b_i, c_i \in \mathbb{F}^{N_w + 1}$. The system $S$ is satisfiable for a statement $x \in \mathbb{F}^n$ if there exists a witness $w \in \mathbb{F}^{N_w}$ such that

- $x = (w_1, \ldots, w_n)$ and
- $[1 \mid w^T]a_i = [1 \mid w^T]b_i = [1 \mid w^T]c_i$ for all $i \in [N_g]$.

We denote this by writing $S(x, w) = 1$ 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 $S$, we define the corresponding relation $\mathcal{R}_S = \{(x, w) \in \mathbb{F}^n \times \mathbb{F}^{N_w} : S(x, w) = 1\}$.

Remark 2.4 (Boolean and Arithmetic Circuit Satisfiability). As is shown in [BCI13, 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 $C : \{0, 1\}^n \times \{0, 1\}^h \rightarrow \{0, 1\}$ with $\alpha$ wires and $\beta$ bilinear gates yields an R1CS instance with $N_w = \alpha$ and $N_g = \beta + h + 1$ [BCI13]. An arithmetic circuit satisfiability instance for an arithmetic circuit $C : \mathbb{F}^n \times \mathbb{F}^h \rightarrow \mathbb{F}^\ell$, where $\ell$ denotes the output size, with $\alpha$ wires and $\beta$ 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 [BCI13, adapted]). Let $\mathcal{S} = (n, N_g, N_w, \{a_i, b_i, c_i\}_{i \in [N_g]})$ be a family of R1CS over a finite field $\mathbb{F}$, let $\mathcal{R}_S$ be the associated relation, and let $\mathcal{L}_S$ be the associated language. A succinct non-interactive argument of knowledge (SNARK) in preprocessing model

\[ \Pi^{\text{SNARK}} = (\text{Setup, Prove, Verify}) \]

with following properties:

\[ \text{In the preprocessing model, we allow for a statement-independent setup algorithm that runs in polynomial time in the size of } \mathcal{S}. \]

In “fully-succinct” SNARK, the setup and prover are required to run in sublinear (or polylogarithmic) in the size of $\mathcal{S}$ [BCI13]. Using recursive composition [BCCT13], it is possible to obtain fully-succinct SNARKs from preprocessing SNARKs.
• \((\text{crs}, \text{st}) \leftarrow \text{Setup}(1^\lambda)\): On input the security parameter \(\lambda\), the setup algorithm outputs a common reference string \(\text{crs}\) and verification state \(\text{st}\).

• \(\pi \leftarrow \text{Prove}(\text{crs}, \text{x}, \text{w})\): On input a common reference string \(\text{crs}\), a statement \(\text{x}\), and a witness \(\text{w}\), the prove algorithm outputs a proof \(\pi\).

• \(b \leftarrow \text{Verify}(\text{st}, \text{x}, \pi)\): On input the verification state \(\text{st}\), a statement \(\text{x}\) and a proof \(\pi\), the verification algorithm outputs a bit \(b \in \{0, 1\}\).

Moreover, \(\Pi_{\text{SNARK}}\) should satisfy all the following properties:

• **Completeness:** For all \(\lambda \in \mathbb{N}\) and all \((\text{x}, \text{w}) \in \mathcal{R}_S\),

\[
\Pr[\text{Verify}(\text{st}, \text{x}, \pi) = 1] = 1,
\]

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

• **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 \(z \in \{0, 1\}^{\text{poly}(\lambda)}\),

\[
\Pr[\text{Verify}(\text{st}, \text{x}, \pi) = 1 \land (\text{x}, \text{w}) \notin \mathcal{R}_S] = \text{negl}(\lambda),
\]

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

• **Efficiency:** There exists an universal polynomial \(p\) (independent of \(\mathcal{S}\)) such that the \(\text{Setup}\) and \(\text{Prove}\) run in time \(p(\lambda + |\mathcal{S}|)\), \(\text{Verify}\) runs in time \(p(\lambda + |\text{x}| + \log |\mathcal{S}|)\), 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 \(\text{st}\) is allowed to be public. Alternatively, a **designated-verifier** SNARK holds security only when the verification state \(\text{st}\) is kept secret, and only the holder of \(\text{st}\) can check proof.

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

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

where the experiment \(\text{ExptZK}_{\Pi_{\text{SNARK}}, \mathcal{A}, \mathcal{S}_{\text{SNARK}}}\) is defined as follows:

1. The challenger samples \(b \leftarrow \{0, 1\}\).

   (a) If \(b = 0\), the challenger computes \((\text{crs}, \text{st}) \leftarrow \text{Setup}(1^\lambda)\) and sends \((\text{crs}, \text{st})\) to \(\mathcal{A}\).

   (b) If \(b = 1\), the challengers computes \((\tilde{\text{crs}}, \tilde{\text{st}}, \text{st}_S) \leftarrow \mathcal{S}_1(1^\lambda)\) and sends \((\tilde{\text{crs}}, \tilde{\text{st}})\) 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 \( \pi \leftarrow \text{Prove}(\text{crs}, \mathbf{x}, \mathbf{w}) \).
   
   (b) If \( b = 1 \), the challenger replies with \( \pi \leftarrow S_2(\text{st}_S, \mathbf{x}) \).

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 \( \varepsilon \)-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 \( \mathcal{R} : \mathbb{F}^n \times \mathbb{F}^h \rightarrow \{0, 1\} \) 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 \( \ell \) and soundness error \( \varepsilon \) is a tuple of algorithms \( \Pi_{\text{LPCP}} = (Q_{\text{LPCP}}, P_{\text{LPCP}}, V_{\text{LPCP}}) \) with the following properties:

- \((\text{st}, \mathbf{Q}) \leftarrow Q_{\text{LPCP}}()\): Query-generator outputs a query matrix \( \mathbf{Q} \in \mathbb{F}^{\ell \times k} \) and a verification state \( \text{st} \).

- \( \pi \leftarrow P_{\text{LPCP}}(\mathbf{x}, \mathbf{w}) \): 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 \( \pi \in \mathbb{F}^\ell \).

- \( b \leftarrow V_{\text{LPCP}}(\text{st}, \mathbf{x}, \mathbf{a}) \): On input the verification state \( \text{st} \), 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, \( \Pi_{\text{LPCP}} \) should satisfy the following properties:

- **Completeness**: For all \( \mathbf{x} \in \mathbb{F}^n \) and \( \mathbf{w} \in \mathbb{F}^h \) where \( \mathcal{R}(\mathbf{x}, \mathbf{w}) = 1 \),
  
  \[
  \Pr[\mathcal{V}_{\text{LPCP}}(\text{st}, \mathbf{x}, Q^T \pi) = 1] = 1,
  \]

  where \((\text{st}, \mathbf{Q}) \leftarrow Q_{\text{LPCP}}()\) and \( \pi \leftarrow P_{\text{LPCP}}(\mathbf{x}, \mathbf{w}) \).

- **Knowledge**: There exists an efficient extractor \( \mathcal{E}_{\text{LPCP}} \) such that for every \( \pi^* \in \mathbb{F}^{\ell} \), if
  
  \[
  \Pr[\mathcal{V}_{\text{LPCP}}(\text{st}, \mathbf{x}, Q^T \pi^*) = 1 \mid (\text{st}, \mathbf{Q}) \leftarrow Q_{\text{LPCP}}()] > \varepsilon,
  \]

  then
  
  \[
  \Pr[\mathcal{R}(\mathbf{x}, \mathbf{w}) = 1 \mid \mathbf{w} \leftarrow \mathcal{E}_{\text{LPCP}}^{(\pi^*)}(x)] = 1.
  \]

\(^2\)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 \( 1^\kappa \) as input, and the parameters \( n, h, k, \ell, \varepsilon \) can all be functions of \( \kappa \).
We refer to $\varepsilon$ as the knowledge error of the linear PCP.

- **Perfect honest-verifier zero-knowledge (HVZK):** There exists an efficient simulator $S_{\text{LPCP}} = (S_1, S_2)$ such that for all $(x, w) \in R$,

\[
\{(st, Q, Q^T \pi) \} = \{(\tilde{st}, \tilde{Q}, \tilde{a})\},
\]

where $(st, Q) \leftarrow Q_{\text{LPCP}}()$, $\pi \leftarrow \mathcal{P}_{\text{LPCP}}(x, w)$, $(\tilde{st}, Q, Q_S) \leftarrow S_1()$, and $\tilde{a} \leftarrow S_2(st_S, x)$. When the statistical distance between these two distributions is $\delta$, we say that $\Pi_{\text{LPCP}}$ is $\delta$-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 $S = (n, N_g, N_w, \{a_i, b_i, c_i\}_{i \in [N_g]}$ be an R1CS system over a finite field $F$, where $a_i, b_i, c_i \in F^{N_g+1}$ (indexed from 0 to $N_w$). We additionally define the following components:

- $S = \{\alpha_1, \ldots, \alpha_{N_g}\} \subset F$ be an arbitrary subset of $F$.
- For each $i \in \{0, \ldots, N_w\}$, let $A_i, B_i, C_i : F \to F$ 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 $Z_S : F \to F$ be the polynomial $Z_S(z) := \prod_{j \in [N_g]}(z - \alpha_j)$, namely $Z_S$ is the polynomial whose roots are the elements of $S$.

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

- $Q_{\text{LPCP}}()$: Sample $\tau \overset{\$}{\leftarrow} F \setminus S$. Let $a = (A_1(\tau), \ldots, A_n(\tau))$, $b = (B_1(\tau), \ldots, B_n(\tau))$, and $c = (C_1(\tau), \ldots, C_n(\tau))$. Output the state $st = (A_0(\tau), B_0(\tau), C_0(\tau), a, b, c, Z_S(\tau))$ and the query matrix $Q$ shown as follows:

\[
Q = \begin{bmatrix}
Z_S(\tau) & 0 & 0 & A_{n+1}(\tau) & \cdots & A_{N_w}(\tau) & 0 & 0 & \cdots & 0 \\
0 & Z_S(\tau) & 0 & B_{n+1}(\tau) & \cdots & B_{N_w}(\tau) & 0 & 0 & \cdots & 0 \\
0 & 0 & Z_S(\tau) & C_{n+1}(\tau) & \cdots & C_{N_w}(\tau) & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & 0 & \cdots & 0 & 1 & \tau & \cdots & \tau^{N_g}
\end{bmatrix}^T \in F^{(4+N_w+N_g-n)\times 4}.
\]

\(3\)In particular, this definition separates the simulator $S_{\text{LPCP}}$ into a statement-independent algorithm $S_1$ and a statement-dependent algorithm $S_2$. This separation is important for arguing adaptive zero-knowledge of our zkSNARKs (where the adversary’s statement can depend on the common reference string).
• $P_{LCP}(x, w)$: On the input $(x, w) \in \mathcal{R}_S$, sample $\delta_1, \delta_2, \delta_3 \overset{R}{\leftarrow} \mathbb{F}$. Construct polynomials $A, B, C : \mathbb{F} \rightarrow \mathbb{F}$, each with degree $N_g$, where

$$A(z) := \delta_1 Z_S(z) + A_0(z) + \sum_{i \in [N_w]} w_i A_i(z) \quad B(z) := \delta_2 Z_S(z) + B_0(z) + \sum_{i \in [N_w]} w_i B_i(z) \quad C(z) := \delta_3 Z_S(z) + C_0(z) + \sum_{i \in [N_w]} w_i C_i(z)$$  \hspace{1cm} (2.3.2)

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

• $\mathcal{V}_{LCP}(st, x, a)$: On the input $st = (a_0, b_0, c_0, a, b, c, z), x \in \mathbb{F}^n$, and $a \in \mathbb{F}^4$, the verifier computes $a'_1 = a_0 + a_1 + x^T a$, $a'_2 = b_0 + a_2 + x^T b$, and $a'_3 = c_0 + a_3 + x^T c$. It accepts if

$$a'_1 a'_2 - a'_3 - a_4 z = 0.$$  \hspace{1cm} (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 $\mathcal{S} = (n, N_g, N_w, \{a_i, b_i, c_i\}_{i \in [N_g]})$ be an R1CS system over $\mathbb{F}$. We prove the properties separately:

• **Completeness:** Take any $(x, w) \in \mathcal{R}_S$, and let $(st, Q) \leftarrow Q_{LCP}(), \pi \overset{R}{\leftarrow} P_{LCP}(x, w), a \leftarrow Q^T \pi$. Consider the value of $\mathcal{V}_{LCP}(st, x, a)$. Let $a'_1, a'_2, a'_3$ be the values computed by $\mathcal{V}_{LCP}$. By definition,

$$a'_1 = a_1 + a_0 + x^T a$$
$$= \delta_1 Z_S(\tau) + A_0(\tau) + \sum_{i \in [n]} x_i A_i(\tau) + \sum_{i \in \mathcal{N}_w} w_i A_{n+i}(\tau)$$
$$= \delta_1 Z_S(\tau) + A_0(\tau) + \sum_{i \in \mathcal{N}_w} w_i A_i(\tau)$$
$$= A(\tau),$$

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 $Q_{LCP}$. 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 $\mathcal{E}_{LPCP}^{(\pi^{\dagger}, \cdot)}$ to be the algorithm that on input a statement $x$ and given linear access to a proof vector $\pi^* = (\delta_1^*, \delta_2^*, \delta_3^*, \hat{w}^*, h^*)$, outputs $\bar{w}^T = [x^T | (\hat{w}^*)^T] \in \mathbb{F}^{N_w}$. To show that the extractor works, take any $\pi^* = (\delta_1^*, \delta_2^*, \delta_3^*, \hat{w}^*, h^*)$ where

$$\Pr[V_{LPCP}(st, x, Q^T \pi^*) = 1 : (st, Q) \leftarrow Q_{LPCP}()] > \frac{2N_g}{|\mathbb{F}| - N_g}.$$  

We use $\pi^*$ and $S$ to define polynomials $A, B, C, H : \mathbb{F} \rightarrow \mathbb{F}$:

$$A(z) = \delta_1^* Z_S(z) + A_0(z) + \sum_{i \in [n]} x_i A_i(z) + \sum_{i \in [N_w-n]} \tilde{w}_i^* A_{n+i}(z)$$

$$B(z) = \delta_2^* Z_S(z) + B_0(z) + \sum_{i \in [n]} x_i B_i(z) + \sum_{i \in [N_w-n]} \tilde{w}_i^* B_{n+i}(z)$$

$$C(z) = \delta_3^* Z_S(z) + C_0(z) + \sum_{i \in [n]} x_i C_i(z) + \sum_{i \in [N_w-n]} \tilde{w}_i^* C_{n+i}(z)$$

$$H(z) = h_0 + \sum_{i \in [N_g]} h_i^* z^i$$

Let $Q$ be the query matrix output by $Q_{LPCP}$, $a \leftarrow Q^T \pi^*$ and $a_1', a_2', a_3'$ be the components computed by $V_{LPCP}$. By construction, we have $a_1' = A(\tau)$, $a_2' = B(\tau)$, $a_3' = C(\tau)$ and $a_4 = H(\tau)$. Define the polynomial $P : \mathbb{F} \rightarrow \mathbb{F}$ where $P(z) = A(z)B(z) - C(z) - H(z)Z_S(z)$. By construction, $\deg(P) \leq 2N_g$. $V_{LPCP}$ accepts if $a_1'a_2' - a_3' - a_4z = 0$, where $z = Z(\tau)$, or equivalently, if

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

Suppose Eq. (2.3.4) holds with probability $\varepsilon > 2N_g/(|\mathbb{F}| - N_g)$; namely, the verifier accepts $\pi^*$ with probability greater than $\varepsilon$. Since $Q_{LPCP}$ samples $\tau$ 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 $Z_S(a_j) = 0$ for all $j \in [N_g]$, this means $A(a_j)B(a_j) = C(a_j)$. By construction of $A, B, C$, let $u^T = [1 \mid x^T \mid (\hat{w}^*)^T]$, we have

$$u^T a_j \cdot u^T b_j = u^T c_j$$

for all $j \in [N_g]$. Since this holds for all $j \in [N_g]$, we have that $(x, \hat{w}^*) \in R_S$, thus knowledge holds.

- **HVZK:** We first construct the simulator $S_{LPCP} = (S_1, S_2)$:

  - $S_1()$: The statement-independent algorithm samples $(\tilde{st}, \tilde{Q}) \leftarrow Q_{LPCP}()$, outputs $\tilde{st}$, $\tilde{Q}$, $\text{st}_{S} = \tilde{st}$

  - $S_2$($\text{st}_{S}, x$): On input the state $\text{st}_{S} = (\tilde{a}_0, \tilde{b}_0, \tilde{c}_0, \tilde{a}, \tilde{b}, \tilde{z})$ and the statement $x$, the statement-dependent algorithm samples $\tilde{a}_1, \tilde{a}_2, \tilde{a}_3 \leftarrow \mathbb{F}$, and computes $\tilde{a}_1' = \tilde{a}_1 + \tilde{a}_0 + x^T \tilde{a}$, $\tilde{a}_2' = \tilde{a}_2 + \tilde{b}_0 + x^T \tilde{b}$,
and \( \bar{a}_3' = \bar{a}_3 + \bar{c}_0 + x^T \hat{e} \). Compute \( \bar{a}_4 = z^{-1}(\bar{a}_1' \bar{a}_2' - \bar{a}_3') \). It outputs \( \bar{a} = (\bar{a}_1, \bar{a}_2, \bar{a}_3, \bar{a}_4) \).

To prove HVZK, it suffices to show that the distribution of the simulation and the real one are identical, for any \((x, w) \in R_S\). By construction, the verification state and query matrix (output by \( S_1 \)) are identically distributed in the two cases. On the distribution of the responses, let \((s_t, Q) \leftarrow Q_{\text{LPCP}}()\), \( \pi \leftarrow \mathcal{P}_{\text{LPCP}}(x, w) \), and \( a \leftarrow Q^T \pi \). Write \( s_t = (a_0, b_0, c_0, a, b, c, z) \). First, \( z = Z_S(\tau) \) for some \( \tau \in F \setminus S \). 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 \( s_t \). The value of \( a_4 \) that satisfies Eq. (2.3.3) precisely coincides with the value \( \bar{a}_4 \) sampled by \( S_{\text{LPCP}} \) (based on the choice of \( \bar{a}_1, \bar{a}_2, \bar{a}_3 \) 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 \( \delta_1 \) is uniform over \( F \) and independent of all other components. Since \( Z_S(\tau) \neq 0 \), this means \( a_1 \) is uniform over \( F \). Similarly, such argument holds for \( a_2 \) and \( a_3 \) (by appealing to the randomness of \( \delta_2 \) and \( \delta_3 \), respectively). This is precisely the distribution of \( \bar{a}_1, \bar{a}_2, \bar{a}_3 \) in the simulation.

Thus, the simulated response is identically distributed as the real response, and perfect HVZK holds. \( \square \)

**Remark 2.11** (Knowledge against Affine Strategies). 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 exploiting 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 \( Q^T \) 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 \( (\pi, b) \) be an affine strategy where \( \pi \in F^m \), \( b \in F^k \). In this case, given the query matrix \( Q \in F^{m \times k} \), the responses are computed as \( a \leftarrow Q^T \pi + b \).
• Suppose that the $j$th column of $Q^T \in \mathbb{F}^{k \times m}$ is a basis vector $e_j \in \mathbb{F}^k$. In this case, an affine strategy of the form $(\pi, \delta e_j)$ is equivalent to a linear strategy $\pi'$ 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 $(\pi, \delta e_j)$.

• 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 $q_1, \ldots, q_k$ be the linear PCP queries sampled by $Q_{\text{LPCP}}$ (i.e., the rows of $Q^T \in \mathbb{F}^{k \times m}$). The modified query-generation algorithm first samples $\gamma_i \overset{\text{rand}}{\in} \mathbb{F}$ for $\ell \in [k]$ and $\ell \neq i$, then computes $q'_i = q_i + \sum_{\ell \neq i} \gamma_\ell q_\ell$ and outputs $(q_1, \ldots, q_{i-1}, q'_i, q_{i+1}, \ldots, q_k)$ 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 $a \in \mathbb{F}^k$, 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 $V_{\text{LPCP}}$ on $(a_1, \ldots, a_{i-1}, a'_i, a_{i+1}, \ldots, q_k)$.

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.\footnote{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.} Consider an arbitrary affine strategy $(\pi, b)$, and let $a_1, \ldots, a_k$ be the responses for the modified linear PCP defined above, we have $a_\ell = q'_i^T \pi + b_\ell$ for $\ell \neq i$ and $a_i = (q'_i)^T \pi + b_i$. The verification relation first computes

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

then checks the polynomial relation

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

Fixing $q_1, \ldots, q_k$, $\pi$, and $b$ (all of which are independent of the $\gamma_\ell$’s), we can view $p$ as a polynomial in the variables $\gamma_1, \ldots, \gamma_k$ for $\ell \neq i$. Since $\gamma_i$’s are sampled uniformly and independently over $\mathbb{F}$ by $Q_{\text{LPCP}}$, 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 $(\pi, b)$ 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 $(\pi, \delta e_j)$ is implied by soundness (resp., knowledge) against linear strategies when the query matrix $Q^T \in \mathbb{F}^{k \times m}$ contains one of its columns as a basis vector $e_j$ (or a scaled version thereof).
In the case of Construction 2.9, by observing $Q^T \in \mathbb{F}^{k \times m}$ 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 $\varepsilon$ 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 $\varepsilon$, we can obtain a $kp$-query LPCP with same query length and knowledge error $\varepsilon^p$. The setup algorithm samples $\rho$ sets of query matrices $Q_1, \ldots, Q_\rho \in \mathbb{F}^{m \times k}$ and construct a query matrix $Q = [Q_1 \mid \ldots \mid Q_\rho] \in \mathbb{F}^{m \times kp}$. The verifier accepts a response $a = [a_1 \mid \ldots \mid a_\rho]$ only if all the responses are valid.
Chapter 3

Lattice-Based Succinct 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 $\Pi_{\text{LPCP}}=(\mathcal{Q}_{\text{LPCP}}, \mathcal{P}_{\text{LPCP}}, \mathcal{V}_{\text{LPCP}})$ be a $k$-query linear PCP for a relation $\mathcal{R}$ over an extension field $\mathbb{F}_{p^d}$ with query length $\ell$. We construct a $(dk)$-query linear PCP $\Pi_{\text{LPCP}} = (\mathcal{Q}_{\text{LPCP}}, \mathcal{P}_{\text{LPCP}}, \mathcal{V}_{\text{LPCP}})$ for $\mathcal{R}$ with query length $d\ell$ over the base field $\mathbb{F}_p$ with following properties:

- $\mathcal{Q}_{\text{LPCP}}$: The query algorithm runs $(\text{st}, Q') \leftarrow \mathcal{Q}'_{\text{LPCP}}()$, where $Q' \in \mathbb{F}_{p^d}^{\ell \times k}$. Let $Q \in \mathbb{F}_p^{d\ell \times dk}$ be the matrix constructed by replacing each entry $q'_{i,j} \in \mathbb{F}_{p^d}$ in $Q'$ with the transpose of “multiply-by-$q'_{i,j}$” matrix $M^T_{q'_{i,j}} \in \mathbb{F}_p^{d \times d}$. Namely,

$$Q' = \begin{bmatrix} q'_{1,1} & \cdots & q'_{1,k} \\ \vdots & \ddots & \vdots \\ q'_{\ell,1} & \cdots & q'_{\ell,k} \end{bmatrix}$$

and

$$Q = \begin{bmatrix} M^T_{q_{1,1}} & \cdots & M^T_{q_{1,k}} \\ \vdots & \ddots & \vdots \\ M^T_{q_{\ell,1}} & \cdots & M^T_{q_{\ell,k}} \end{bmatrix}.$$
We begin by the definition of a vector encryption scheme (adapted from \cite{BISW17}), and then define the

\begin{theorem}[\(\mathbb{F}_p\)-Linear PCP to \(\mathbb{F}_p\)-Linear PCP] If \(\Pi'_{\text{LPCP}}\) is complete, has knowledge error \(\varepsilon\) and is perfect HVZK, then the same holds for \(\Pi_{\text{LPCP}}\) from Construction 3.1.
\end{theorem}

\begin{proof}
We prove the properties separately:

- **Completeness**: Take any \((x, w) \in \mathcal{R}\), compute \((st, Q') \in Q'_{\text{LPCP}}\), \(\pi' \leftarrow \mathcal{P}'_{\text{LPCP}}(x, w)\). Let \(Q \in \mathbb{F}_p^{d \times dk}\) and \(\pi \in \mathbb{F}_p^{d}\) be as specified in \(Q_{\text{LPCP}}\) and \(\mathcal{P}_{\text{LPCP}}\) and let \(\mathbf{a} \leftarrow Q^T \pi\). We write \(\mathbf{a} = [a_1, \ldots, a_k]\). By construction, for all \(i \in [k]\),

\[
    a_i = \sum_{j \in [\ell]} M_{q'_j, i} v_{q'_j, \pi'_j} = \sum_{j \in [\ell]} v_{q'_j, \pi'_j}^T \pi'_j \in \mathbb{F}_p^d,
\]

where \(q'_{i,j} \in \mathbb{F}_p^d\) corresponds to the \(i\)th column of \(Q'\). This means that the vector \(\mathbf{a}'\) computed by \(\mathcal{V}_{\text{LPCP}}\) satisfies \(\mathbf{a}' = (Q')^T \pi'\). Thus, completeness of \(\Pi_{\text{LPCP}}\) holds by following the completeness of \(\Pi'_{\text{LPCP}}\).

- **Knowledge**: Compute \((st, Q') \leftarrow Q'_{\text{LPCP}}\) and let \(Q\) be the matrix \(Q_{\text{LPCP}}\) constructs from \(Q'\). Take any \(\pi \in \mathbb{F}_p^{d}\) and let \(\pi' \in \mathbb{F}_p^{d}\) be the vector obtained by viewing \(\pi\) as \(\ell\) blocks of \(d\) elements. By construction, \(\mathcal{V}_{\text{LPCP}}(st, x, (Q')^T \pi') = 1\) if and only if \(\mathcal{V}'_{\text{LPCP}}(st, x, (Q')^T \pi') = 1\). The extractor \(\mathcal{E}_{\text{LPCP}}\) for \(\Pi_{\text{LPCP}}\) can invoke the extractor \(\mathcal{E}'_{\text{LPCP}}\) for \(\Pi'_{\text{LPCP}}\), since any linear query \(q'\) that \(\mathcal{E}'_{\text{LPCP}}\) makes to \((\pi', \cdot)\) can be simulated via a linear query to \((\cdot)^T \pi\) by expanding each component in \(q'\) into a matrix of \(\mathbb{F}_p^{d \times d}\). The knowledge of \(\Pi_{\text{LPCP}}\) now follows the knowledge soundness of \(\Pi'_{\text{LPCP}}\).

- **Perfect HVZK**: On the input \(x\), the \(\mathcal{S}_{\text{LPCP}}\) runs the simulator \(\mathcal{S}'_{\text{LPCP}}(x)\) for \(\Pi'_{\text{LPCP}}\) by outputs \((st, \hat{Q}', \hat{a}')\) where \(\hat{Q}' \in \mathbb{F}_p^{d \times k}\) and \(\hat{a}' \in \mathbb{F}_p^\ell\). The simulator \(\mathcal{S}_{\text{LPCP}}\) constructs \(\hat{Q} \in \mathbb{F}_p^{d \times dk}\) by replacing each entry \(\hat{q}_{i,j} \in \mathbb{F}_p^d\) in \(\hat{Q}'\) with \(\mathbf{M} \hat{q}_{i,j} \in \mathbb{F}_p^{d \times d}\) and constructs \(\hat{a} \in \mathbb{F}_p^d\) by replacing each element \(\hat{a}'_i \in \mathbb{F}_p^d\) in \(\hat{a}'\) with \(\mathbf{v} \hat{a}'_i \in \mathbb{F}_p^d\). \(\mathcal{S}_{\text{LPCP}}\) outputs \((st, \hat{Q}, \hat{a})\). Perfect HVZK of \(\Pi_{\text{LPCP}}\) follows by perfect HVZK of \(\Pi'_{\text{LPCP}}\).
\end{proof}

### 3.2 Linear-Only Vector Encryption

We begin by the definition of a vector encryption scheme (adapted from \cite{BISW17}), and then define the linear-only \cite{BC13, BISW17} property we rely on our zkSNARK construction.
CHAPTER 3. LATTICE-BASED SUCCINT ARGUMENTS

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 $\Pi_{\text{Enc}} = (\text{Setup}, \text{Encrypt}, \text{Decrypt}, \text{Add})$ with following properties:

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

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

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

- $ct^* \leftarrow \text{Add}(pp, \{ct_i\}_{i \in [n]}, \{c_i\}_{i \in [n]})$: On input the public parameters, a collection of ciphertexts $ct_1, \ldots, ct_n$ and scalars $c_1, \ldots, c_n \in \mathbb{F}$, the addition algorithm outputs a new ciphertext $ct^*$.

Moreover, $\Pi_{\text{Enc}}$ should satisfy the following properties:

- **Additive Homomorphism:** For all security parameter $\lambda \in \mathbb{N}$, a collection of plaintext vectors $v_1, \ldots, v_k \in \mathbb{F}^\ell$, and scalars $y_1, \ldots, y_k \in \mathbb{F}$, where $k = k(\lambda)$,

$$\Pr[\text{Decrypt}(sk, ct^*) = \sum_{i \in [k]} y_i v_i] = 1 - \text{negl}(\lambda), \quad (3.2.1)$$

where $(sk, pp) \leftarrow \text{Setup}(1^\lambda, 1^\ell)$, $ct_i \leftarrow \text{Encrypt}(sk, v_i)$ for all $i \in [k]$, and $ct^* \leftarrow \text{Add}(pp, \{ct_i\}_{i \in [k]}, \{y_i\}_{i \in [k]})$.

We say $\Pi_{\text{Enc}}$ is additively-homomorphic with respect to $S \subseteq \mathbb{R}^k$ 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 $A$,

$$\Pr[A^{O_b}(\cdot)(1^\lambda, pp) = b] = \frac{1}{2} + \text{negl}(\lambda), \quad (3.2.2)$$

where $(sk, pp) \leftarrow \text{Setup}(1^\lambda, 1^\ell)$, $O_b$ takes $sk$ and $v_1, v_2 \in \mathbb{F}^\ell$ as input, and output $ct_b \leftarrow \text{Encrypt}(sk, v_2)$.

We say $\Pi_{\text{Enc}}$ is $Q$-query CPA secure if Eq. (3.2.2) holds against at most $Q$ queries from $O_b$.

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

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

where the experiment $\text{ExptLinearExt}_{\Pi_{\text{Enc}}, A, M, E, z}(1^\lambda)$ is defined as follows:
This is important to argue zero knowledge (see Section 3.4 and Theorem 3.26). where the experiment 

**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 $S$. 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 $v$ 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 relaxation where Definition 3.4 holds only for auxiliary inputs sampled from “benign” distributions (e.g., in our applications, it suffices to consider auxiliary inputs that are 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 MLWE$_{n,m,d,q,\chi}$ states that for $A \leftarrow R_q^{n \times m}, s \leftarrow \chi^n, e \leftarrow \chi^m$ and $u \leftarrow R_q^m$, the following two distributions are computationally indistinguishable:

$$(A, s^T A + e^T)$$ and $$(A, u).$$

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 \(\left\lfloor k \cdot q/p \right\rfloor \mod q\) (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 \(\mathbf{ct} = (\mathbf{a}, \mathbf{c})\). 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 \(\mathbf{ct} = (\mathbf{ct}_1, \mathbf{ct}_2)\) 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 \(\tau\) 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).
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:

- \( \ell \): the plaintext dimension
- \( \tau \): the sparsification parameter
- \( B \): the noise smudging bound

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

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

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

- \( ct^* \leftarrow \text{Add}(pp, \{ct_i\}_{i \in [k]}, \{y_i\}_{i \in [k]})\): On the input of the public parameters \( pp = (A, D) \), ciphertexts \( ct_i = (a_i, c_i) \) for \( i \in [k] \), and scalars \( y_i \in R_p \), sample \( r \leftarrow \chi^n, e_a \leftarrow \chi^n, e_c \leftarrow \chi^{k - |B|} [-B, B]^\ell' \) and outputs

\[
ct^* = \left( \sum_{i \in [k]} y_i a_i + Ar + pe_a, \sum_{i \in [k]} y_i c_i + Dr + pe_c \right). \tag{3.3.1}
\]

- \( v/\perp \leftarrow \text{Decrypt}(sk, ct)\): On the input of the secret key \( sk = (S, T) \) and ciphertext \( ct = (a, c) \), compute \( z \leftarrow c - S^T a \in R_q^{\ell'} \). Compute \( u = z \mod p \), and parse \( u^T = [v_1^T \mid v_2^T] \) where \( v_1 \in R_p^{\ell'} \) and \( v_2 \in R_p^n \). Output \( v_1 \) if \( v_2 = TV_1 \), 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 $\lambda$ be a security parameter, and $p, q, n, \chi, \ell', B$ be as defined in Construction 3.11. Suppose $\chi$ is subgaussian \(^3\) with parameter $s$. If $n, \ell', s, d, \gamma_R = \text{poly}(\lambda)$, then for all $k = k(\lambda)$, there exists $q = (pB + kp^2) \cdot \text{poly}(\lambda)$ 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, $S = \{y \in R_p^k : \|y\|_1 \leq B_1$ and $\|y\|_2 \leq B_2\}$. If $\ell', n > 8$, and

$$q > 2p(B + \gamma_R B_2Cs + \gamma_R B_1/2 + 2\gamma_R nC^2s^2) + p,$$

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

**Proof.** We begin with the concrete statement. Take any collection of plaintext vectors $v_1, \ldots, v_k \in R_p^d$ and scalars $y_1, \ldots, y_k \in R_p$. Let $(pp, sk) \leftarrow \text{Setup}(1^\lambda, 1^{\ell'})$, $c_t \leftarrow \text{Encrypt}(sk, v_i)$ for $i \in [k]$. Define ciphertext $ct^* \leftarrow \text{Add}(pp, \{ct_i\}_{i \in [k]}, \{y_i\}_{i \in [k]})$. Here, $pp = (A, D)$, $sk = (S, T)$, $c_t = (a_i, c_i)$ and $ct^* = (a^*, c^*)$. By construction, $D = S^T A + pE^T$, $c^* = \sum_{i \in [k]} y_i a_i + Ar + pe_a$, and $a^* = \sum_{i \in [k]} y_i c_i + Dr + pe_c$.

For $i \in [k]$, we also have $c_i = S^T a_i + pe_i + u_i$, where $u_i^T = [v_i^T \mid (Tv_i)^T]$. Thus,

$$c^* = \sum_{i \in [k]} (y_i S^T a_i + y_i pe_i + y_i u_i) + S^T Ar + pE^T r + pe_c. \tag{3.3.3}$$

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

$$z^* = c^* - S^T a^* = \sum_{i \in [k]} y_i u_i + p \left( \sum_{i \in [k]} y_i e_i + E^T r + e_c - S^T e_a \right).$$

We write $\sum_{i \in [k]} y_i u_i = pu + u' \in R_{\ell'}$ where $u, u' \in R_{\ell'}$ and $\|u'||_\infty < p/2$. Then,

$$z^* = \sum_{i \in [k]} y_i u_i \mod p + p \left( u + \sum_{i \in [k]} y_i e_i + E^T r + e_c - S^T e_a \right) \in R_{\ell'}^q. \tag{3.3.4}$$

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

---

\(^3\)When $d > 1$, we assume that $\chi$ is the concatenation of $d$ independent subgaussian distributions over $Z$, each with parameter at most $s$. This is true for discrete Gaussian distributions over a power-of-two cyclotomic ring.
decryption algorithm computes

\[ u^T = \sum_{i \in [k]} y_i u_i^T = \left[ \sum_{i \in [k]} y_i v_i^T \mid \sum_{i \in [k]} y_i (Tv_i)^T \right] \in R_p^{e'}, \]

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

- By definition, \( \tilde{u} = 1/p \cdot (\sum_{i \in [k]} y_i u_i - \sum_{i \in k} y_i u_i \mod p) \). This means that \( \|\tilde{u}\|_\infty \leq 1/p \cdot \left\| \sum_{i \in [k]} y_i u_i \right\|_\infty \).

  Since \( u_i \in R_p^{e'} \), this means \( \|u_i\|_\infty \leq p/2 \), so \( \|\tilde{u}\|_\infty \leq \gamma_R \|y\|_1 / 2 \).

- The entries of \( e_i \in R_q^{e'} \) are sampled from \( \chi \), which by assumption, is the product of \( d \) independent subgaussian distributions over \( \mathbb{Z}_r \), each with parameter at most \( s \). Since \( R = \mathbb{Z}[x]/(x^d + 1) \), each component of \( \sum_{i \in [k]} y_i e_i \) is subgaussian with parameter \( \gamma_R \|y\|_2 C_s \). Thus, the magnitude of each component of \( \sum_{i \in [k]} y_i e_i \) is bounded by \( \gamma_R \|y\|_2 C_s \) with probability at least 1 − 2 exp(−\( \pi C^2 \)). By a union bound, \( \left\| \sum_{i \in [k]} y_i e_i \right\|_\infty \leq \gamma_R \|y\|_2 C_s \) with probability at least 1 − 2dn\( e' \) exp(−\( \pi C^2 \)).

- Since the entries of \( E \in R_q^{n \times d'} \) and \( r \in R_n^{d'} \) are sampled from \( \chi \), the magnitude of each element in \( E \) and \( r \) is bounded by \( C_s \) with probability at least 1 − 2d exp(−\( \pi C^2 \)). By a union bound, \( \|E^r\|_\infty \leq \gamma_R n C^2 s^2 \) with probability 1 − 2dn\( e' \) exp(−\( \pi C^2 \)).

- Since \( e_c \) is sampled from \([-B, B]^{d'} \), \( \|e_c\|_\infty \leq B \).

- Since the entries of \( S \in R_q^{n \times d'} \) and \( e_a \in R_n^{d'} \) are sampled from \( \chi \), we have that \( \|S^a e_a\|_\infty \leq \gamma_R n C^2 s^2 \) with probability at least 1 − 2dn\( e' \) exp(−\( \pi C^2 \)).

Again by a union bound, with probability at least 1 − (\( 4n + 2 \))dn\( e' \) exp(−\( \pi C^2 \)),

\[ \|e'\|_\infty \leq B + \gamma_R \|y\|_2 C_s + \gamma_R \|y\|_1 / 2 + 2\gamma_R n C^2 s^2. \]

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

\[ \square \]

**Theorem 3.13** (CPA Security). **Fix a security parameter \( \lambda \) and let \( p, q, n, d', \chi \) be as defined in Construction 3.11. Take any \( Q = \text{poly}(\lambda) \) and suppose that \( p, q \) are coprime. Under the 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 MLWE_{n,m,d,q,\chi} assumption implies that the following two distributions are computationally indistinguishable:

\[ (A', S^T A' + (E')^T) \] and \( (A', (U')^T), \]

(3.3.5)
where $A' \overset{\$}{\leftarrow} R_q^{n \times m}$, $S \leftarrow \chi^{n \times \ell'}$, $E' \leftarrow \chi^{m \times \ell'}$, and $U' \overset{\$}{\leftarrow} R_q^{m \times \ell'}$ (for any $\ell' = \text{poly}(\lambda)$). 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 $A \overset{\$}{\leftarrow} R_q^{n \times n}$, $S \leftarrow \chi^{n \times \ell'}$, $E \leftarrow \chi^{n \times \ell'}$, $T \overset{\$}{\leftarrow} R_p^{\ell' \times \ell}$, $b \overset{\$}{\leftarrow} \{0, 1\}$ and computes $D \leftarrow S^T A + pE^T$. The challenger sends $pp = (A, D)$ to $A$.

  (b) On the $i$th oracle query $(v_{i,0}, v_{i,1})$, the challenger samples $a_i \overset{\$}{\leftarrow} R_q^n$, $e_i \overset{\$}{\leftarrow} \chi^{\ell'}$, computes $u_i^T = [v_{i,0}^T | (Tv_{i,1})^T]$, $c_i \leftarrow S^T a_i + pe_i + u_i$, and sends $ct_i = (a_i, c_i)$ to $A$.

  (c) After $Q$ queries, $A$ 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 $D \overset{\$}{\leftarrow} R_q^{\ell' \times n}$ in the public parameters, and for each adversary’s oracle query, the challenger computes $c_i \leftarrow pr_i + u_i$, where $r_i \overset{\$}{\leftarrow} R_q^{\ell'}$.

We first argue that the outputs of Hyb$_0$ and Hyb$_1$ are computationally indistinguishable under the MLWE$_{n,m,d,q,\chi}$ assumption. Let $(A', Z')$ be the MLWE challenge and write $A' = [A | a_1 | \cdots | a_Q]$, where $A \in R_q^{n \times n}$ and $a_1, \ldots, a_Q \in R_q^n$. Analogously, write $Z' = [Z | z_1 | \cdots | z_Q]$. Consider a CPA-security experiment where we set $pp = (pA, pZ)$ and respond to the adversary’s oracle queries $(v_{i,0}, v_{i,1})$ with the ciphertext $ct_i = (pa_i, pz_i + u_i)$. In this case, if $Z' = S^T A' + (E')^T$ then this perfectly simulates Hyb$_0$. If $Z' \overset{\$}{\leftarrow} R_q^{\ell' \times m}$, this perfectly simulates Hyb$_1$. (In particular, if $A$ is uniform over $R_q^{n \times n}$, then so is $pA$ since $\gcd(p, q) = 1$, and likewise for $pa_i$ for $i \in [Q]$.) Thus, the outputs of Hyb$_0$ and Hyb$_1$ are computationally indistinguishable by MLWE$_{n,m,d,q,\chi}$ assumption.

Noticing that in Hyb$_1$, the advantage of distinguishing ciphertext is exactly $1/2$ since each $r_i$ for $i \in [Q]$ is uniform and independent over $R_q^{\ell'}$ and $\gcd(p, q) = 1$. Thus, the challenge ciphertexts perfectly hide $v_i$’s. □

**Theorem 3.14 (Circuit Privacy).** Let $\lambda$ be a security parameter and $p, q, n, \ell', \chi$ be as defined in Construction 3.11. Suppose that $\chi$ is subgaussian with parameter $\sigma$. If $n, \ell', s, d, \gamma_R = \text{poly}(\lambda)$, and $B = 2^{(\gamma_C, 2)} \cdot k p^2$, then under the MLWE$_{n,m,d,q,\chi}$ assumption with $m = n$, Construction 3.11 is circuit private with respect to the set $S = R_q^k$.

Concretely, let $C$ be a correctness parameter, $B_1, B_2$ be bounds, $S = \{ y \in R_q^k : \| y \|_1 \leq B_1 \text{ and } \| y \|_2 \leq B_2 \}$. Then under the MLWE$_{n,m,d,q,\chi}$ assumption with $m = n$, for every efficient adversary $A$ restricted to strategies in $S$, there exists an efficient simulator $S$ where

$$\Pr[\text{ExptCircuitPriv}_{\text{Hyb}}.A, S(1^{\lambda}) = 1] \leq 1/2 + \varepsilon + \text{negl}(\lambda),$$

and

$$\varepsilon = (4n + 2)dl \exp(-\pi C^2) + \frac{dl'(\gamma_R B_2 C + \gamma_R B_1 / 2 + 2\gamma_R n C^2 s^2)}{B}.$$ (3.3.6)
Proof. We first construct a simulator $S$. On input the security parameter $\lambda$, the public parameters $pp = (A, D)$, the secret key $sk = (S, T)$, and a vector $v \in R_p^e$, the simulator proceeds as follows:

1. Sample $a \overset{\$}{\leftarrow} R_q^n$ and $\bar{e} \leftarrow [-B, B]^{de'}$.
2. Compute $u^T = [v^T \mid (Tv)^T] \in R_p^{de'}$ and output the ciphertext $ct^* = (a, S^T a + p\bar{e} + u)$.

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

- **Hyb$_0$**: This is the real circuit privacy experiment.
  
  (a) The challenger begins by sampling $(pp, sk) \leftarrow \text{Setup}(1^\lambda, 1^e)$ where $pp = (A, D)$ and $sk = (S, T)$, then sends $pp$ and $sk$ to $A$.
  
  (b) $A$ outputs a collection of vectors $v_1, \ldots, v_k$.
  
  (c) The challenger computes $ct_i \leftarrow \text{Encrypt}(sk, v_i)$ for $i \in [k]$, where $ct_i$ can be written as $ct_i = (a_i, c_i)$, for $c_i = S^T a_i + pe_i + u_i$, $e_i \in R_q^n$ and $u_i^T = [v_i^T \mid (Tv_i)^T]$. Challenger sends $ct_1, \ldots, ct_k$ to $A$.
  
  (d) Adversary $A$ outputs a set of coefficients $y_1, \ldots, y_k \in R_p$.
  
  (e) The challenger computes $ct_0^*$ and $ct_i^*$ as follows:
    
    - For $ct_0^*$, the challenger uses the $\text{Add}$ algorithm: It samples $r \overset{\$}{\leftarrow} \chi^n$, $e_a \leftarrow \chi^n$, $e_c \leftarrow [-B, B]^{de'}$, computes $a_0^* \leftarrow \sum_{i \in [k]} y_i a_i + Ar + pe_a$ and $c_0^* \leftarrow \sum_{i \in [k]} y_i c_i + Dr + pe_c$, and sets $ct_0^* = (a_0^*, c_0^*)$.
    
    - For $ct_i^*$, the challenger uses the simulator $S$: It samples $a_i^* \overset{\$}{\leftarrow} R_q^n$, $\bar{e} \leftarrow [-B, B]^{de'}$, computes $c_i^* \leftarrow S^T a_i^* + p\bar{e} + u$, where $u^T = [\sum_{i \in [k]} y_i v_i^T \mid \sum_{i \in [k]} y_i (Tv_i)^T] \in R_p^{de'}$, and sets $ct_i^* = (a_i^*, c_i^*)$.
  
  (f) The challenger samples $b \overset{\$}{\leftarrow} \{0, 1\}$ and sends $ct_0^*$ to $A$.
  
  (g) $A$ 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 $u^T = [\sum_{i \in [k]} y_i v_i^T \mid \sum_{i \in [k]} y_i (Tv_i)^T] \in R_p^{de'}$ and $c_0^* \leftarrow S^T a_0^* + pe_c + u$.

- **Hyb$_2$**: Same as Hyb$_1$ except the challenger samples $a_0^* \overset{\$}{\leftarrow} R_q^n$.

For an adversary $A$, we write $\text{Hyb}_1(A)$ to denote the output distribution of $\text{Hyb}_1$ with adversary $A$. We argue that the output distribution of each pair of adjacent hybrids are computationally (or statistically) indistinguishable. Finally, we show that for all adversaries $A$, $\Pr[\text{Hyb}_2(A) = 1] = 1/2$.

**Lemma 3.15.** For all adversaries $A$ (restricted to strategies in $S$), the statistical distance between $\text{Hyb}_0(A)$ and $\text{Hyb}_1(A)$ is $\varepsilon$ (see Eq. (3.3.6)).
Proof. The only difference between $\text{Hyb}_0$ and $\text{Hyb}_1$ is how $c_0^*$ is constructed. As in the proof of Theorem 3.12 (see Eq. (3.3.3)), in $\text{Hyb}_0$,

$$c_0^* = \sum_{i \in [k]} (y_i S^T a_i + y_i p e_i + y_i u_i) + S^T Ar + p E^T r + pc$$

$$= S^T a_0^* + p \left[ \tilde{u} + \sum_{i \in [k]} y_i e_i + E^T r - S^T e_c + e_c \right] + u,$$

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

$$\|e'\|_\infty \leq \gamma_R \|y\|_2 Cs + \gamma_R \|y\|_1 / 2 + 2\gamma R n C^2 s^2.$$ 

In $\text{Hyb}_1$, $c_0^* = S^T a_0^* + pc + u$. By Lemma 2.1 and a union bound, the statistical distance between the distributions of $c_e$ and $e' + c_e$ is at most $\ell' \|e'\|_\infty / B$. The claim follows. \qed

**Lemma 3.16.** Suppose that $p, q$ are coprime. Under the MLWE$_{n,m,d,q,x}$ assumption with $m = n$, for all efficient adversaries $\mathcal{A}$, $\text{Hyb}_1(\mathcal{A})$ and $\text{Hyb}_2(\mathcal{A})$ are computationally indistinguishable.

Proof. The only difference between $\text{Hyb}_1$ and $\text{Hyb}_2$ is that in $\text{Hyb}_1$, $a_0^* \leftarrow \sum_{i \in [k]} y_i a_i + Ar + pc_a$ while in $\text{Hyb}_2$, $a_0^* \leftarrow R_q^n$. This follows from the MLWE assumption. To see this, suppose there is an adversary $\mathcal{A}$ where $|\Pr[\text{Hyb}_1(\mathcal{A}) = 1] - \Pr[\text{Hyb}_2(\mathcal{A}) = 1]| \geq \epsilon'$. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ for MLWE. Let $(A, z)$ be the MLWE challenge where $A \in R_q^n \times n$ and $z \in R_q^n$. Algorithm $\mathcal{B}$ simulates an execution of $\text{Hyb}_1$ or $\text{Hyb}_2$ as follows:

- Sample $S \leftarrow \chi^{n \times \ell'}$, $E \leftarrow \chi^{n \times \ell'}$ and $T \leftarrow R_p^{\ell \times \ell'}$. Let $D \leftarrow S^T (p A^T) + p E^T$, and set $pp = (p A^T, D)$ and $sk = (S, T)$. Since $A$ is uniform and $\gcd(p, q) = 1$, the matrix $p A^T$ is also uniform.

- Let $a_0^* \leftarrow \sum_{i \in [k]} y_i a_i + p z$. Construct the remaining components $c_0^*$, $a_1^*$, and $c_1^*$ as in $\text{Hyb}_1$ and $\text{Hyb}_2$.

- Sample $b \leftarrow \{0, 1\}$ and give $ct^*_0$ to $\mathcal{A}$.

- Let $b' \in \{0, 1\}$ be the adversary’s output. Output 1 if $b' = b$ and 0 otherwise.

If $z^T = s^T A + e^T$, then $a_0^* = \sum_{i \in [k]} y_i a_i + p A^T s + p e$ where $s \leftarrow \chi^n$ and $e \leftarrow \chi^n$. This is precisely the distribution in $\text{Hyb}_1$. Conversely, if $z \leftarrow R_q^n$, then $a_0^*$ is uniform (since $\gcd(p, q) = 1$ and $z$ is uniform and independent of the distribution of $y_i$ and $a_i$). This is precisely the distribution in $\text{Hyb}_2$. Thus, $\mathcal{B}$ breaks MLWE with advantage $\epsilon'$.

To conclude the proof, observe that $ct^*_0$ and $ct^*_1$ are identically distributed in $\text{Hyb}_2$. Thus, for all adversaries $\mathcal{A}$, $\Pr[\text{Hyb}_2(\mathcal{A}) = 1] = 1/2$. Together with Lemmas 3.15 and 3.16, this means that for all efficient adversaries $\mathcal{A}$, $\Pr[\text{Hyb}_0(\mathcal{A}) = 1] \leq 1/2 + \epsilon + \negl(\lambda)$. For the asymptotic statement, we set $C = \omega(\log \lambda)$ and $B_1, B_2 = pk$. \qed
Conjecture 3.17 (Linear-Only). Fix a security parameter $\lambda$, and let $p, d, \tau$ be defined as in Construction 3.11. If $|R_p|^\tau = p^{\tau d} = \lambda^{c(1)}$ 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 $u^T = [v^T | (Tv)^T]$, where $T$ 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 $\ell$ copies of $\mathbb{F}_p^k$ (i.e., when the polynomial $\Phi_m(x)$ splits into $\ell$ irreducible factors $F_1(x), \ldots, F_\ell(x) \mod p$), it is possible to pack $\ell$ 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 $\ell$ R1CS systems. In this way, a single ciphertext contains packed responses to $\ell$ different statements across $\ell$ 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. 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 \( \ell_1 \) and \( \ell_2 \) 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 \( \sum_{i=1}^m \alpha_i p^{(i-1)/m} \mathbf{v} \), where \( \alpha = \alpha_{m-1} \cdots \alpha_1 \alpha_0 \) is the value of \( \alpha \) 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 \( \ell_1 \) norm of the coefficients decreases from \( kp \) to \( mkp^{1/m} \); the \( \ell_2 \) norm decreases from \( \sqrt{kp} \) to \( \sqrt{mkp^{1/m}} \).

**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 \( \text{Scale} \) from Bitansky et al. [BGV12]:

- \( \mathbf{x}' \leftarrow \text{Scale}(\mathbf{x}, q', p) \): 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 \( \mathbf{x}' = \mathbf{x} \mod p \).

**Theorem 3.21** (Modulus Switching [BGV12, adapted]). Let \( \lambda \) be a security parameter and \( p, q, n, d, \ell', \chi \) be as defined in Construction 3.11. Let \( C \) be a correctness parameter. Suppose that \( \chi \) is subgaussian with parameter \( s \). Let \( q' < q \) be a positive integer where \( q' = q \mod p \). Take any vector \( \mathbf{a} \in R_q^n \), \( \mathbf{c} \in R_{q'}^\ell \), and let \( \mathbf{a}' \leftarrow \text{Scale}(\mathbf{a}, q, q', p) \), \( \mathbf{c}' \leftarrow \text{Scale}(\mathbf{c}, q, q', p) \). Sample \( \mathbf{S} \leftarrow \chi^{n \times \ell'} \). Let \( \mathbf{z} = \mathbf{c} - \mathbf{S}^T \mathbf{a} \in R_{q'}^\ell \) and suppose that

\[
\|\mathbf{z}\|_\infty < q/2 - (1 + n\gamma R C s) \cdot (p/2) \cdot (q/q').
\]

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

\footnote{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 \( 2p_i \) 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.}
We state the result of Bitansky et al. [BCI +] 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.

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.

Let $z = c - S^T a$ (mod $q$), we can write $z = c - S^T a + q \tilde{v} \in R^\ell$, where $\tilde{v} \in R^\ell$. Define the vector $\tilde{z} = c' - S^T a' + q' \tilde{v} \in R^\ell$. Then,

$$
\|\tilde{z}\|_\infty = \|c' - S^T a' + q' \tilde{v}\|_\infty
= \|c' - S^T a' + q' \tilde{v} + (q'/q) \cdot \left((c - S^T a + q \tilde{v}) - (c - S^T a + q \tilde{v})\right)\|_\infty
\leq \|c' - (q'/q) \cdot c\|_\infty + \|S^T (a' - (q'/q) \cdot a)\|_\infty + (q'/q) \cdot \|z\|_\infty.
$$

We analyze each term separately:

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

$$
(q'/q) \cdot \|z\|_\infty < q'/2 - (1 + n \gamma R Cs) \cdot (p/2) / (q'/p).
$$

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

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

since $c = c' \pmod{p}$, $a = a' \pmod{p}$, 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 $F$ be a finite field, $S$ be an RICS system over $F$, and let $R_S$ be the associated relation. The construction has following building blocks:

- Let $\Pi_{LPCP} = (Q_{LPCP}, P_{LPCP}, \nu_{LPCP})$ be a $k$-query input-oblivious linear PCP for $R_S$. 
Let $\Pi_{Enc} = (\text{Setup}, \text{Encrypt}, \text{Decrypt}, \text{Add})$ be a secret-key additively-homomorphic vector encryption scheme for $\mathbb{F}^k$. We construct a single-theorem designated-verifier zkSNARK $\Pi_{SNARK} = (\text{Setup}, \text{Prove}, \text{Verify})$ for $R_S$ as follows:

- **Setup($1^\lambda$, $1^\ell$):** On input the security parameter $\lambda$ and the circuit family parameter $\ell$, run $(\text{st}_{\text{LPCP}}, Q) \leftarrow Q_{\text{LPCP}}()$ where $Q \in \mathbb{F}^{m \times \ell}$. Write $q_i \in \mathbb{F}$ to denote the $i^{th}$ row of $Q$ for $i \in [m]$. Then compute $(pp, sk) \leftarrow \text{Setup}_{Enc}(1^\lambda, 1^\ell)$ and $ct_i \leftarrow \text{Encrypt}_{Enc}(sk, q_i^T)$ for $i \in [m]$. Output the common reference string $\text{crs} = (pp, ct_1, \ldots, ct_m)$ and the verification state $\text{st} = (\text{st}_{\text{LPCP}}, sk)$.

- **Prove($\text{crs}, x, w$):** On input the common reference string $\text{crs} = (pp, ct_1, \ldots, ct_m)$, a statement $x$, and a witness $w$, the prover first construct an LPCP proof $\pi \leftarrow P_{\text{LPCP}}(x, w)$. The prover then homomorphically computes the linear PCP response $ct^* \leftarrow \text{Add}_{Enc}(pp, \{ct_1, \ldots, ct_m\}, \{\pi_1, \ldots, \pi_m\})$, and outputs $\pi = ct^*$.

- **Verify($\text{st}, x, \pi$):** On input the verification state $\text{st} = (\text{st}_{\text{LPCP}}, sk)$, the statement $x$, and the proof $\pi = ct^*$, the verifier decrypts $a \leftarrow \text{Decrypt}_{Enc}(sk, ct^*)$. Then it outputs $0$ if $a = \perp$, otherwise $\lambda_{\text{LPCP}}(\text{st}_{\text{LPCP}}, x, a)$.

**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 $\Pi_{\text{LPCP}}$ is statistical sound against linear prover and $\Pi_{Enc}$ is CPA-secure (for up to $m$ messages) and strictly linear-only, then $\Pi_{\text{SNARK}}$ from Construction 3.22 is a designated-verifier SNARK for $R_S$ 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. 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 \( \Pi_{\text{LPCP}} \) and \( \Pi_{\text{Enc}} \) be as defined in Theorem 3.23. If \( \Pi_{\text{LPCP}} \) satisfies perfect honest-verifier zero-knowledge and \( \Pi_{\text{Enc}} \) is CPA-secure (for up to \( m \) message) and computationally (resp., statistically) circuit private, then \( \Pi_{\text{SNARK}} \) from Construction 3.22 is computationally (resp., statistically) zero-knowledge. More precisely, if \( \Pi_{\text{LPCP}} \) is \( \delta \)-HVZK and \( \Pi_{\text{Enc}} \) is \( \varepsilon \)-circuit private, then \( \Pi_{\text{SNARK}} \) from Construction 3.22 is \( 2(\delta + \varepsilon) \)-zero-knowledge.

**Proof.** Let \( S_{\text{LPCP}} = (S_{\text{LPCP,1}}, S_{\text{LPCP,2}}) \) be the simulator for \( \Pi_{\text{LPCP}} \), and \( S_{\text{Enc}} \) be the circuit privacy simulator for \( \Pi_{\text{Enc}} \). We use \( S_{\text{LPCP}} \) and \( S_{\text{Enc}} \) to construct a simulator \( S = (S_1, S_2) \) for \( \Pi_{\text{SNARK}} \):

- \( S_1(1^\lambda, 1^\ell) \): On the input of \( \lambda \) and \( \ell \), run \( (\tilde{s}_{\text{LPCP}}, \tilde{Q}, \tilde{s}_{\text{LPCP,s}}) \leftarrow S_{\text{LPCP,1}}() \), where \( \tilde{Q} \in \mathbb{F}^{m \times k} \). Compute \( (\tilde{p}_p, \tilde{s}_k) \leftarrow \text{Setup}_{\text{Enc}}(1^\lambda, 1^k) \) and \( \tilde{c}_i \leftarrow \text{Encrypt}_{\text{Enc}}(\tilde{s}_k, \tilde{q}_{i1}) \) for \( i \in [m] \). Output \( \tilde{\text{crs}} = (\tilde{p}_p, \tilde{c}_1, \ldots, \tilde{c}_m) \),

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 \(\tilde{s} = (\tilde{s}_{1\text{LPCP}}, \tilde{s}_k)\), and the simulation state \(s_{t, s} = (s_{1\text{LPCP}, s}, pp, sk)\).

- \(S_2(st, x)\): On the input of \(st = (s_{1\text{LPCP}, s}, pp, sk)\) and the statement \(x\), compute \(\tilde{a} \leftarrow S_{1\text{LPCP}, 2}(st_{1\text{LPCP}, s}, x)\) and output the simulated proof \(\tilde{x} = S_{\text{Enc}}(1^\lambda, pp, sk, \tilde{a})\).

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 \(b \xleftarrow{} \{0, 1\}\), \((crs, st) \leftarrow \text{Setup}(1^\lambda, 1^\ell)\), and \((\tilde{crs}, \tilde{s}, st_{1\text{LPCP}, s}) \leftarrow S_1(1^\lambda, 1^\ell)\). If \(b = 0\), it gives \((crs, st)\) to the adversary, and otherwise, it gives \((\tilde{crs}, \tilde{s})\) to the adversary. After the adversary outputs a statement \(x\) and witness \(w\), the challenger first checks that \(R(x, w) = 1\) (aborting the experiment otherwise), then computes \(\pi \leftarrow \text{Prove}(crs, x, w)\) and \(\tilde{\pi} \leftarrow S_1(st, x)\). It gives \(\pi\) to \(A\) 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 \(\pi\) using the circuit privacy simulator (in place of \(\text{Prove}(crs, x, w)\)). Let \(Q \in \mathbb{F}^{m \times k}\) be the query matrix the challenger sampled to construct \(crs = (pp, ct_1, \ldots, ct_m)\) and let \(st = (st_{1\text{LPCP}, s}, sk)\) be the corresponding verification state. To construct \(\pi\), the challenger now computes \(\pi \leftarrow \Pi_{1\text{PCP}}(x, w)\) and sets \(\pi \leftarrow S_{\text{Enc}}(1^\lambda, pp, sk, a)\), where \(a = Q^T\pi\).

- **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 \(Q_{1\text{LPCP}}\) to obtain \(st_{1\text{LPCP}}\) and \(Q\), the challenger instead samples \((st_{1\text{LPCP}}, Q, st_{1\text{LPCP}, s}) \leftarrow S_{1\text{LPCP}, 1}(\cdot)\). When constructing the proof, the challenger substitutes the simulated response \(a \leftarrow S_{1\text{LPCP}, 2}(st_{1\text{LPCP}, s}, x)\) in place of the value \(a = Q^T\pi\) from Hyb_1.

Let \(\text{Hyb}_1(A)\) be the output of an execution of experiment \(\text{Hyb}_1\). We now analyze the distribution of each pair of adjacent hybrid distributions as well as the distribution in \(\text{Hyb}_2\):

- The only difference between \(\text{Hyb}_0\) and \(\text{Hyb}_1\) is the challenger computes \(\pi\) using the simulator \(S_{\text{Enc}}\) (with target value \(a = Q^T\pi\)) instead of using \(\text{Add}_{\text{Enc}}(pp, \{ct_1, \ldots, ct_m\}, \{\pi_1, \ldots, \pi_m\}\)) where each \(ct_i\) is an encryption of \(q_i^T\) (in which case \(\sum_{i \in [m]} \pi_i q_i^T = Q^T\pi\)). If \(\Pi_{\text{Enc}}\) is \(\delta\)-circuit private, then

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

- The only difference between \(\text{Hyb}_1\) and \(\text{Hyb}_2\) is the challenger uses the linear PCP simulator \(S_{1\text{LPCP}}\) to sample the queries \(Q\), the verification state \(st\), and the responses \(a\) instead of using \(Q_{1\text{LPCP}}\) and \(\Pi_{1\text{LPCP}}\). If \(\Pi_{1\text{LPCP}}\) is \(\varepsilon\)-HVZK, then

\[
|\Pr[\text{Hyb}_1(A) = 1] - \Pr[\text{Hyb}_2(A) = 1]| \leq 2\varepsilon.
\]
• Finally, in $\text{Hyb}_2$, the behavior of the challenger is the same regardless of the bit $b$ (i.e., the challenger computes $\text{crs}$, $\text{st}$, and $\pi$ according to the specification of $S$). This means that for all adversaries $A$, $\Pr[\text{Hyb}_2(A) = 1] = 1/2$, and the claim holds.

\textbf{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 $2^{3.3}$ times 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

$$ct^* = (a^*, c^*) = \left( \sum_{i \in [k]} y_i a_i, \sum_{i \in [k]} y_i c_i \right).$$

(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

$$c^* = S^* a^* + \sum_{i \in [k]} y_i u_i \mod p + p \left( \tilde{u} + \sum_{i \in [k]} y_i e_i \right),$$

where $\tilde{u}, \tilde{u}' \in R^{\ell'}, \|\tilde{u}'\|_\infty < p/2$, and $\sum_{i \in [k]} y_i u_i = p\tilde{u} + \tilde{u}' \in R^{\ell'}$. In the standard circuit-privacy setting, the simulator is only given $\sum_{i \in [k]} y_i u_i \mod p$ (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 $\sum_{i \in [k]} y_i a_i \in \mathbb{R}^n$ and the value of $\tilde{u} + \sum_{i \in [k]} y_i e_i \in \mathbb{R}^{\ell'}$ (technically, the second relation is also over $\mathbb{R}_q^{\ell'}$, 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 $e_i$ with coefficients drawn from the linear PCP. In this case, the length $k$ of the linear PCP is significantly larger than the dimension $\ell'$ 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.

43
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 efficiently 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 notations from [BCG+13], for a polynomial \( A(z) \) of degree less than \( |D| \), we write \( \text{FFT}_D(A(z)) \) to denote the vector of evaluations \( \{A(\alpha)\}_{\alpha \in D} \). Similarly, we write \( \text{FFT}_D^{-1}((A'(\alpha))_{\alpha \in D}) \) 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 \( \omega \). Let \( \xi_1 = 1 \) and take \( \xi_i \in \mathbb{F}^* \setminus H_1 \) for \( i = 2, \ldots, k \), 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 \( \hat{\mathbf{V}}_H \in \mathbb{F}^{d \times 2^d} \) be the Vandermonde matrix associated with evaluating a polynomial of degree up to \( 2^d \) on \( H \) (i.e., the roots of unity). Then, we have that

\[
\mathbf{V}_S = \begin{bmatrix}
\hat{\mathbf{V}}_H \cdot \Xi_2 & \hat{\mathbf{V}}_H \cdot \xi_2^{2d} \Xi_2 & \cdots & \hat{\mathbf{V}}_H \cdot \xi_2^{(k-1)2d} \Xi_2 \\
\vdots & \vdots & & \vdots \\
\hat{\mathbf{V}}_H \cdot \Xi_k & \hat{\mathbf{V}}_H \cdot \xi_k^{2d} \Xi_k & \cdots & \hat{\mathbf{V}}_H \cdot \xi_k^{(k-1)2d} \Xi_k
\end{bmatrix}
= \begin{bmatrix}
\hat{\mathbf{V}}_H^T \\
\mathbf{V}_H^T
\end{bmatrix}^T
\begin{bmatrix}
\mathbf{I}_{2^d} & \mathbf{I}_{2^d} & \cdots & \mathbf{I}_{2^d} \\
\Xi_2 & \xi_2^{2d} \Xi_2 & \cdots & \xi_2^{(k-1)2d} \Xi_2 \\
\vdots & \vdots & & \vdots \\
\Xi_k & \xi_k^{2d} \Xi_k & \cdots & \xi_k^{(k-1)2d} \Xi_k
\end{bmatrix}
\]

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

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

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.

- Naively, we can compute \( \hat{\mathbf{b}}_i \) in \( O(k \cdot 2^d) \) time with brute force since \( \Xi_i \) is a diagonal matrix, so computing all of the entries \( \mathbf{b} = \left[ \hat{\mathbf{b}}_1^T \mid \ldots \mid \hat{\mathbf{b}}_k^T \right]^T \in \mathbb{F}^{k \cdot 2^d} \) 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(2^d)}^{-1} \xi_{\ell, j} \hat{a}_{\ell,j}.


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

\[
\tilde{b}_j = \text{diag}(\xi_1^{j-1}, \ldots, \xi_k^{j-1})
\]

\[
\tilde{a}_j.
\]

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 \( \xi_1^d, \ldots, \xi_k^d \). While \( \xi_1^d, \ldots, \xi_k^d \) 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 \( \tilde{b}_j \) can be done in just \( O(k \log k) \) time.

Repeating this for all \( j \in [2^d] \) yields an algorithm to compute \( b \) 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 \( Q_{\text{LPCP}} \) is dominated by the complexity of evaluating each \( A_i, B_i, C_i \) at the random element \( \tau \). 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 \( \omega \). 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 \( \xi_i = g^{2^{i-2}} \in \mathbb{F}_{p^2}^* \setminus H \) for \( i \in [k] \), such that \( H_i = \xi_i H \) are pairwise disjoint cosets of \( H \). Let \( S = \{ \alpha_i \}_{i \in [N_y]} = \bigcup_{i \in [k]} H_i \subset \mathbb{F}_{p^2} \) be the domain for multipoint evaluation interpolation.

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

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

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

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

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

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
   
   \[ \theta_i = \prod_{r \in [k], r \neq i} L_{j,r}(z) = \prod_{r \in [k], r \neq i} \frac{z^{2^{d+1}} - \xi_r^{2^{d+1}}}{\xi_r^{2^{d+1}} - \xi_r^{2^{d+1}}}. \]
   
   ii. Evaluate all the radix-2 FFT Lagrange interpolation polynomials \( \{L'_t(\cdot)\}_{t \in [2^{d+1}]} \) on \( \xi_r^{-1}z \).
   
   iii. For \( t \in [2^{d+1}] \), let \( k = (i-1) \cdot 2^{d+1} + t \), and compute \( L_k(z) = \theta_i \cdot L'_t(\xi_r^{-1}z) \).

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 \( \{a_i\}_{i \in \{N_g\}} \).

**LPCP Prover Computation.** The complexity of computing \( P_{\text{LPCP}} \) 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 extFFT\(_D\)(\( A(z) \)) to denote the evaluated results \( \{A(\alpha)\}_{\alpha \in D} \) from the extended radix-2 FFTs over cosets. Similarly, we use 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 \( D \).
CHAPTER 4. IMPLEMENTATION AND EVALUATION

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

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

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

   
   \begin{align*}
   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{align*}

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

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 inverting $\text{extFFT}^{-1}_T(\{H'(\beta_i)\}_{i \in [N_g]})$.

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 $Z_S(z) = \prod_{j \in [k]} \left( (g\xi_j)^{2d+1} - \xi_i^{2d+1} \right)$ 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 $\text{extFFT}_T$ in step 4, $\text{extFFT}^{-1}_T$ in step 6, and $\text{extFFT}^{-1}_S$ 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

\[
\text{extFFT}_T(\cdot) = (\text{extFFT}_S \circ \Xi_g)(\cdot)
\]
\[
\text{extFFT}^{-1}_T(\cdot) = (\Xi_g^{-1} \circ \text{extFFT}^{-1}_S)(\cdot).
\]
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 $n, q, \chi$ 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 $\kappa$ 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 $\mathbb{F}_p^*$ 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 $\mathbb{F}_p^*$ contains $2^{20}$-th roots of unity).

- The module rank $d$ is chosen based on whether we are working with a linear PCP over $\mathbb{F}_{p^2}$ ($d = 2$) or if we are working over a linear PCP over $\mathbb{F}_p$ ($d = 1$). Since we work over $R = \mathbb{Z}[x]/(x^d + 1)$, $\gamma_R = 1$ if $d = 1$ and $\gamma_R = 2$ if $d = 2$.

- The plaintext dimension $\ell$ 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 $\rho$ times to amplify soundness ($\rho$ is chosen such that $(2N_g/(p^2 - N_g))^\rho \leq 2^{-128}$). 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 $\tau$ is chosen based on Conjecture 3.17: namely, we choose $\tau$ to be the smallest value where $p^{-\tau d} \leq 2^{-128}$.

- The noise smudging bound $B$ is chosen so that $\varepsilon = 2^{-\kappa}$ in Eq. (3.3.6) of Theorem 3.14 (which ensures roughly $\kappa$ 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 $B_2 = \sqrt{dkp}$. 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 $B_2 = \sqrt{dkzp^{1/z}}$. 
The value of \( \varepsilon \) in Eq. (3.3.6) is essentially determined by \( \gamma R B_2 C s + \gamma R B_1 / 2 + 2 \gamma R n C^2 s^2 \). This term is effectively dominated by the first two terms \( \gamma R B_2 C s + \gamma R B_1 / 2 = \gamma R (\sqrt{k} p C s + pk / 2) \). We take our noise distribution to be a discrete Gaussian distribution with noise rate \( s \), and we choose \( s \) to balance the terms \( \sqrt{k} p C s \) and \( pk / 2 \). Since the security of LWE is determined by the modulus-to-noise ratio, using a larger noise rate reduces the lattice dimension; balancing these two terms allows us to use a higher noise rate without needing to increase the modulus size needed for correctness.

- We choose the modulus \( q \) to be the smallest power of two that satisfies Eq. (3.3.2). 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 factor 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 \( \text{ct} = (a, c) \) where \( a \in R_q^n \) is uniformly random and \( c \in R_q^\ell' \) encodes the message. For typical parameter settings, the module dimension \( n \) is much larger than the (extended) plaintext dimension \( \ell' \).

A heuristic approach to reduce the ciphertext size is to derive the random vector \( a \) as the output of a pseudorandom function (PRF) and include the PRF key in place of the vector \( a \) (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 \( \text{ct}_i = (a_i, c_i), \ldots, c_N = (a_N, c_N) \) in the CRS, the setup algorithm samples a PRF key \( k \) and sets \( a_i \leftarrow \text{PRF}(k, i) \). The sequence of ciphertexts in the CRS is then \( (k, c_1, \ldots, c_N) \). 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/
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 \( a \)'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 \( \chi \) 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 \( \chi_s \) 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

<table>
<thead>
<tr>
<th>Fields</th>
<th>( \lambda_q )</th>
<th>( \lambda_c )</th>
<th>( p )</th>
<th>( n \times d )</th>
<th>( \log q )</th>
<th>( \log q' )</th>
<th>( s )</th>
<th>( \ell )</th>
<th>( \tau )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbb{F}_p, \mathbb{F}_p )</td>
<td>128</td>
<td>138</td>
<td>( 5 \cdot 2^{25} + 1 )</td>
<td>(4700, 1)</td>
<td>123</td>
<td>49</td>
<td>80</td>
<td>82</td>
<td>5</td>
</tr>
<tr>
<td>( \mathbb{F}_{p^2}, \mathbb{F}_p )</td>
<td>128</td>
<td>138</td>
<td>( 2^{13} - 1 )</td>
<td>(3585, 1)</td>
<td>97</td>
<td>35</td>
<td>66</td>
<td>208</td>
<td>10</td>
</tr>
<tr>
<td>( \mathbb{F}<em>{p^2}, \mathbb{F}</em>{p^2} )</td>
<td>128</td>
<td>138</td>
<td>( 2^{13} - 1 )</td>
<td>(1815, 2)</td>
<td>98</td>
<td>35</td>
<td>64</td>
<td>104</td>
<td>5</td>
</tr>
</tbody>
</table>

* The first field listed is the base field for the linear PCP \( \Pi_{\text{LPCP}} \) and the second is the plaintext field for the linear-only vector encryption scheme \( \Pi_{\text{Enc}} \).
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.

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Time (s)</th>
<th>Rate (muls/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Native (uint64_t)</td>
<td>2.1</td>
<td>$4.68 \cdot 10^9$</td>
</tr>
<tr>
<td>Compiler Intrinsic (_uint128_t)</td>
<td>6.8</td>
<td>$1.47 \cdot 10^9$</td>
</tr>
<tr>
<td>Boost Fixed Precision (128-Bit)</td>
<td>6.8</td>
<td>$1.47 \cdot 10^9$</td>
</tr>
<tr>
<td>Boost Fixed Precision (192-Bit)</td>
<td>53.6</td>
<td>$0.19 \cdot 10^9$</td>
</tr>
<tr>
<td>Boost Fixed Precision (256-Bit)</td>
<td>61.9</td>
<td>$0.16 \cdot 10^9$</td>
</tr>
<tr>
<td>GMP Multi-precision (mod $2^{128}$)</td>
<td>114.9</td>
<td>$0.087 \cdot 10^9$</td>
</tr>
</tbody>
</table>

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×) 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 $3 \times$. 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.

3This 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.
CHAPTER 4. IMPLEMENTATION AND EVALUATION

Metrics and evaluation methodology. Following previous works \[\text{BCR}^{+19}, \text{COS}20, \text{SL}20\], 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 \texttt{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 \texttt{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 \(2^{8.8}\times\) (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 \(1.4\text{--}1.5\times\) 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 \(\mathbb{R}_q\) 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 \( 1.2 \times \) to \( 1.4 \times \) shorter and the proof is \( 1.5 \times \) to \( 2 \times \) 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 \( 2.2 \times \) to \( 2.4 \times \) shorter and the proof is \( 1.2 \times \) to \( 1.5 \times \) 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 \( a \in \mathbb{R}_q^n \) 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 \( 2 \times \), 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 $p = 2^{13} - 1$.

The standard power-of-two FFTs because of lacking in primitive roots of unity). By extrapolating the performance, our approach is about $7 \times$ slower than the basic radix-2 FFT. 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 $\kappa$. 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 $2.3 \times$ faster than the pairing-based construction of Groth et al. [Gro16]; the proof size is 11.1 KB, which is $89 \times$ 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 $20 \times$ 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

---

4When $p = 2^{13} - 1$, 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 $7 \times$ overhead.
Figure 4.4: Cost breakdowns as a function of the zero-knowledge parameter \( \kappa \) (i.e., the zero-knowledge distinguishing advantage of any \( \text{poly}(\lambda) \) adversary is bounded by \( 2^{-\kappa} + \negl(\lambda) \)). All measurements taken for an \( \text{R1CS} \) instance over \( \mathbb{F}_{p^2} \) with \( 2^{20} \) constraints (and compiled using 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 \texttt{libsnark} [Lab21c] for the implementation of Groth’s pairing-based construction [Gro16] and \texttt{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 soundness. As is discussed in Remark 3.25, while the strong soundness property holds for our the underlying linear PCP, our encryption scheme does not satsify 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 outsourcing. 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 \( \Pi_{\text{Enc}} \) over a finite field \( \mathbb{F} \) is a tuple of algorithms \( \Pi_{\text{Enc}} = (\text{Setup}, \text{Encode}, \text{Add}) \) with the following properties:

- \( \text{Setup}(1^\lambda) \rightarrow (pk, sk) \): On input the security parameter \( \lambda \), the setup algorithm outputs a public evaluation key \( pk \) and a secret encoding key \( sk \).

- \( \text{Encode}(sk, x) \rightarrow \text{enc}_x \): On input the secret key \( sk \) and an element \( x \in \mathbb{F} \), the encoding algorithm outputs an encoding \( \text{enc}_x \) of \( x \).

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

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 \( \alpha_1, \ldots, \alpha_k \in \mathbb{F}^d \),

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

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

**Definition A.2** (\( q \)-Power Diffie-Hellman Assumption [Gro10, GMNO18]). Fix a parameter \( q \in \mathbb{N} \). A linear encoding scheme \( \Pi_{\text{Enc}} = (\text{Setup}, \text{Encode}, \text{Add}) \) over a field \( \mathbb{F} \) satisfies the \( q \)-power Diffie-Hellman assumption...
(q-PDH) if for all efficient adversaries $A$, 

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

where $(pk, sk) \leftarrow \text{Setup}(1^\lambda), s \leftarrow \mathbb{F}, enc_i \leftarrow \text{Encode}(sk, s^i)$ for all $i \in \{0, \ldots, 2q\}, \sigma \leftarrow (pk, enc_0, \ldots, enc_q, enc_{q+2}, \ldots, enc_{2q}),$ and $S$ is the set of encodings in the support of $\text{Encode}(sk, s^{q+1})$.

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

**Proof.** The adversary $A$ starts by choosing $2q$ distinct points $z_1, \ldots, z_{2q} \in \mathbb{F}$, and forms the polynomial 

$$f(x) = \prod_{i \in [2q]} (x - z_i).$$

Write this as $f(x) = \sum_{i=0}^{2q} \alpha_i x^i$. Then, for all $i \in [2q]$, $z_i^{q+1} = -\alpha_i^{-1} \sum_{j \neq q+1} \alpha_j z_j^{q+1}$. Let $(pk, enc_0, \ldots, enc_q, enc_{q+2}, \ldots, enc_{2q})$ 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 $-\alpha_i^{-1} \sum_{j \neq q+1} \alpha_j s^i$. By the above analysis, if $s \in \{z_1, \ldots, z_{2q}\}$, then this quantity is exactly $s^{q+1}$. Since $s$ is uniform and independent of $z_1, \ldots, z_{2q}$, the probability that $s \in \{z_1, \ldots, z_{2q}\}$ is exactly $2q/|\mathbb{F}|$, which proves the claim.

**Remark 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


BIBLIOGRAPHY


