Algebraic Techniques for Universal Succinct Arguments

Victoria Arantxa Zapico Barrionuevo

TESI DOCTORAL UPF / 2022

SUPERVISORS: Vanesa Daza and Carla Ràfols

Department of Information and Communication Technologies

A la memoria de mi abuela,
Nelly Colombres,

y a Alejandra.

Acknowledgments

After these years, all the places, experiences, things I learned, tried and changed, the big conclusion is the same as in any other big step: it has all been about the people.

Four years ago I had a terrible concept of myself as a mathematician and almost no expectation of this thesis but thanks to the patience, love, support, and knowledge of my supervisors, Carla Ràfols and Vanesa Daza, I happen to want to, and am sure I can, work on this for the rest of my life. I am struggling to find the words to express how grateful I am for the luck of working with them because they are not only good cryptographers, but amazing women, and I admire them in many senses. *Moltes gràcies* for being an example, for letting and actually pushing me to prioritize what excites me and makes me happy. To both of you, thanks for taking the time to know me and find the best way to guide me, and for making it fun. To Vanesa, for always having a solution to *everything*. To Carla, for talking to me as a pair, and therefore forcing me to be up to it. I have enjoyed every day of the last four years, inside and outside the office, the last hours before deadlines and entire weeks of holidays, and that is almost entirely thanks to them.

Thanks to the Crypto Community, for the effort on making science open, inclusive, and collaborative, which gives a bit of sense to all of this. To Mary Maller, for being such a role model and later giving me the academic honour and personal pleasure of working with her. To the Ethereum Foundation research group for being so welcoming and patient.

Thanks to my co-authors: Vitalik Buterin, Matteo Campanelli, Vanesa Daza, Abida Haque, Dmitry Khovratovich, Mary Maller, Carla Ràfols, Alessandra Scafuro, Mark Simkin, Alexandros Zacharakis, and Anca Nitulescu. To the latter, for being nice and friendly since day zero to a very novice me. To Dario Fiore, Chaya Ganesh and Markulf Kohlweiss, for accepting being part of my jury. To the Criptolatino community, for building the perfect place for my career and convictions to live together.

For the privilege of working with friends, starting every day with a bunch of smiles and always having someone close by to complain with, I want to thank Pablo Aragón, Javier Silva, Zaira Pindado, Rasoul Silab, Federico Franzoni, Sergi Rovira, Conor McMeimam, Mohamed Ben Zaghta, and Masoud Sadeghian. Thank you guys for always letting me win Catan.

I am not sure where to name Alexandros Zacharakis, if as one of my supervisors, the friends I had the privilege to share work with every day, the brilliant co-authors, or the people who blew up my mind. In any case, it has been nothing but a pleasure (and somewhat cheating, I am sure) to share this path with him. Thanks Alex for making everything more interesting, exciting, and easier.

To all the great people that work and have worked in the DTIC, building such an international, open, interesting and friendly environment. It is difficult for me to confess this, but joining the Volleyball tournament has been a great decision, almost as good as going to Sopena that very first time. Kudos to Adrià Arbués, for being so good at being only himself, and letting me be mine.

To my favourite and safest place in Europe: Yasmin Soares de Lima and Silvia Butti. When I look back and think about all the amazing things that have happened to me these four years, I can not see myself alone at any point, they have been by my side at every step of this journey. They have been love, lessons, challenges, example, support, convictions, encouragement, home, fun, science, feminism and even the never-thought vegetarianism. You girls are a huge part of what I am and like to be today, thanks.

To LaCaixitos, the strongest and funniest support network I could have asked for, for giving this experience that extra touch that made it so special, for blowing up my mind and for the opportunity to know and travel the world in so many ways. Especially, to Bradley Higginson, Christoph Herbert, Enrico Almici, Ifeanyi Ezeonwumelu, Ignasi Granero, Iván Fernández, Ivan Milenkovic, Loïc Reymond, Luca Morelli, Maximilian Loeck, and Poonam Nebhnani, because their friendship summarizes pretty much everything I left Argentina looking for.

To all the amazing friends I have made these years in Barcelona, for keeping me mentally safe and sound. To Savvas and Bruno, for being home and hugs.

A la Universidad Nacional, pública y gratuita de Córdoba por formarme no sólo en el ámbito académico, si no también en el ámbito social y personal, creando los valores que hoy me definen y de los que espero estar a la altura el resto de mi carrera profesional. A las personas que forman la FaMAF, por transmitir tanta pasión por la ciencia y por la calidez humana. A mis compañerxs durante esos años, a Belén y Franco por hacerlo siquiera un poco más fácil.

A mis amigas y amigos de Córdoba, por estar siempre del otro lado, a pesar de los años y la distancia, para multiplicar la felicidad y alivianar las tristezas con tanto amor. Si hay algo que aprendí en estos años es que soy muy afortunada de tener un lugar al cual siempre quiero volver, un hogar tan mio. Por hacerlo todo tan simple y auténtico, gracias.

A Martín, porque su tan contagioso amor por la vida y la ciencia me llevaron a lugares increíbles, y este doctorado es uno de ellos.

A mi familia, porque su apoyo, felicidad y amor son mis mayores tesoros y los motores de todo lo que hago. Por ayudarme a crecer, madurar y después volar lejos. Por estar siempre para abrazarme y decirme que siga cuando todo falla y miro atrás.

A Patricia, Norma, Luis y Hugo, por ser guía y paciencia. A Marcelo y Mariano,

por desafiarme de las peores formas posibles, y a Daira, por hacerlo siempre de la mejor. A mis mejores amigas, Patricia, Daiana y Candelaria, por la sabidura, la generosidad, y la complicidad, desde siempre y para siempre.

El gracias más grande es, sin duda, a las tres personas que más me apoyan, exigen e impulsan en cada una de mis ocurrencias. A mi papá, por las convicciones y la libertad de hacer con ellas *lo que se me dé la gana.* A mi mamá, por la inagotable perseverancia, por ser ejemplo y encontrar siempre la manera de adaptarse para ayudarme. A mi hermana Alejandra, la persona más leal y compañera de mi mundo, por su infinito amor y enseñanzas.

Por último, a mi primera y mejor maestra, mi abuela Coca, por impulsarme (obligarme) a ir siempre por más, y darme la confianza, las herramientas y la familia para hacerlo, por enseñarme a ser yo y a ser feliz. Esta tesis es para ella y por nosotras, nuestras tardes y el día en que me enseñó a contar con tapitas de colores.

Abstract

In this thesis, we make theoretical and practical contributions to the design of succinct arguments with universal setups in the pairing-based setting. We first introduce a new primitive, Checkable Subspace Sampling (CSS) schemes, and use it to build a framework for designing zero-knowledge succinct arguments of knowledge (zkSNARKs) for NP-complete problems. We present several instantiations of CSS that lead to zkSNARKs whose efficiency is competitive, and in most of the cases superior to all previous constructions in the state-of-the-art. Our second contribution is to present a framework for constructing Linear-Map Vector Commitment schemes with updatability and unbounded aggregation from simpler arguments, that prove a committed vector satisfies an inner product relation. We present two constructions of such arguments, that can be used as building blocks in many different succinct arguments, and the first pairing-based maintainable linear-map vector commitment scheme with flexible space/time trade-offs in the univariate, universal SRS model. Finally, we introduce the definition of Position-Hiding linkability for vector commitments and the first scheme that achieves logarithmic prover and constant proof for membership proofs and lookup tables.

Resumen

En esta tesis, contribuímos en los ámbitos práctico y teórico al desarrollo de arguments sucintos en grupos bilineales y con parámetros universales. Como primer resultado, definimos esquemas verificables de sampleo en un subespacio (CSS), y los empleamos en la construcción de un marco para el diseño de argumentos de conocimiento, sucintos, no interactivos y de conocimiento nulo (zkSNARKs) para problemas NP completos. Asimismo, presentamos diversos esquemas CSS que conducen a zkSNARKs cuya eficiencia es competitiva, y en la mayora de los casos superior, a la de todas las construcciones existentes en la literatura. Nuestra segunda contribución es un marco para el diseño de esquemas de compromiso a vectores para mapeos lineales que permite actualizar y agregar pruebas, a partir de argumentos más simples que prueban a partir de su compromiso, que un vector satisface una relación de producto interno. Presentamos dos construcciones de este tipo de argumentos, que pueden ser usadas en diferentes esquemas sucintos, y el primer argumento que, en el escenario de los grupos bilineales con parmetros universales y univariados, permite al probador elegir de manera flexible un equilibrio entre el coste en tiempo y espacio, y actualizar eficientemente las pruebas almacenadas. Finalmente, definimos enlazabilidad con conocimiento nulo para esquemas de compromiso a vectores y el primer esquema con probador logarítmico y prueba de tamaño constante para argumentos de pertenencia a un conjunto y tablas de búsqueda.

Resum

En aquesta tesi, contribuïm en els àmbits pràctic i teòric al desenvolupament d'arguments succints en grups bilineals i amb paràmetres universals. Com a primer resultat, definim esquemes verificables de sample en un subespai (CSS), i els fem servir en la construcció d'un marc per al disseny d'arguments de coneixement, succints, no interactius i de coneixement nul (zkSNARKs) per a problemes NP complets. Així mateix, presentem diversos esquemes CSS que condueixen a zkSNARKs l'eficincia dels quals és competitiva, i en la majoria dels casos superior, a la de totes les construccions existents a la literatura. La nostra segona contribució és un marc per al disseny d'esquemes de compromís a vectors per a mapeigs lineals que permet actualitzar i afegir proves, a partir d'arguments més simples que proven a partir del seu compromís, que un vector satisfà una relació de producte intern. Presentem dues construccions d'aquest tipus d'arguments, que poden ser usades en diferents esquemes succints, i el primer argument que, a l'escenari dels grups bilineals amb paràmetres universals i univariats, permet al provador escollir de manera flexible un equilibri entre el cost en temps i espai, i actualitzar eficientment les proves emmagatzemades. Finalment, definim enllaabilitat amb conoixement nul a esquemes de compromás a vectors i el primer esquema amb provador local i prova de mida constant per a arguments de pertinena a un conjunt i taules de cerca.

Contents

List of Figures

Chapter 1

Introduction

Cryptography emerged, as its etymology reveals, as the art of secret writing. We tend to think of it primarily as a technology that enables and protects the privacy of our communications. For instance, proving systems, that are the focus of this thesis, allow to a party to prove that some statement is true without leaking unnecessary information. While privacy is their most demanded feature, the potential of these cryptographic techniques for scalability, understood as the ability to handle big amounts of data, spend few resources and get quick responses, is getting increasingly recognized.

In a proof system, a *prover* wants to convince a *verifier* about the truth of some statement. If the prover is right, they hold a *witness*, some piece of data that is linked to the statement and proves it. Using the witness, the prover constructs a *proof* that they send to the verifier and that should suffice for the latter to decide on the validity of the statement. Such a system must ensure two conditions: *completeness,* meaning that any honest prover should be able to convince the verifier and *soundness,* which captures the fact that if the statement is not true, the verifier should reject it with *some* probability. If this probability is one we say the system is a proof, and whenever is smaller, we have an *argument*. *Zero-knowledge*, defined as the requirement that no information about the witness is leaked to the verifier, can be additionally requested.

The efficiency of arguments is at the core of this thesis, and it will be determined according to the amount of work each of the parties has to perform, the memory they use and the size of the messages that are sent. If these quantities lie within some bounds that will be defined later, we say the argument is *succinct*.

Due to the ability to prove in a concise manner arbitrarily large statements, succinct arguments [Kil92, Mic00] have a wide range of applications. In systems for delegation of computation, for instance, we have a computationally weak verifier that asks some stronger party to perform an expensive procedure for them. The strong party becomes the prover and, upon sending the result, has to convince

the verifier of its correctness: the need for a succinct proof and a cheap verifier is fundamental due to the resources of the latter. This scenario of a weak verifier and stronger prover arises a lot in a digital world where users operate through small devices, such as laptops, phones and even watches while delegating to more capable entities (companies, governments, professionals) many of their tasks. In all cases, the inefficiency of these systems understood as time and resources, translates into a monetary cost.

In particular, since the advent of cryptocurrencies [Nak08], the importance of privacy and efficiency of proving arguments has become of massive interest, as transactions in many of these schemes consist of proving the satisfiability of some very large arithmetic circuit. Making short proofs then becomes essential for scalability of the blockchains and fee costs, while the need for anonymity brought zero-knowledge on stage, especially since 2014 when Zcash [BCG+14] suggested the usage of zero-knowledge proofs for making transactions anonymously.

Zero-Knowledge proof systems were introduced by Goldwasser, Micali and Rackoff [GMR89], and its non-interactive version (NIZK) by Blum, Feldman and Micali [BFM88]. These works along with subsequent ones [Dam93, FLS99, KP98] have proven the existence of NIZK arguments for all NP languages. Albeit being seminal works on the theory of proving systems, the constructions presented in the aforementioned papers lack *concrete* efficiency, and their usage would have required a prohibitive cost in practice.

By relaxing security for the sake of efficiency, cryptographers introduced Zero-Knowledge *Succinct Non-interactive ARguments of Knowledge* (zkSNARKs) with highly convenient concrete efficiency: proof systems for any NP-complete problem where the size of the proof and the work required by the verifier are independent of the size of the statement [Gro10, Lip12, Lip13, PHGR13, BCTV14, Gro16]. In particular, proof size is only 3 group elements for the best construction [Gro16]. As a security drawback, these schemes require the presence of a trusted third party to compute some public parameters, the *structured reference string (SRS)*, that constitute the key point for succinctness. *Structured* captures the fact that the elements on the trapdoor are related to each other, as opposite to the case for a *uniform* reference string where they are randomly sampled.

The SRS contains a short description of the relation to be proven so the verifier does not need to read the statement, which will take them linear time on its size, and can be constant. It also includes encodings of some secret elements, the *trapdoor*, that are used to produce short proofs. Keeping the trapdoor secret is crucial, as any party in possession of it can produce valid proofs for false statements.

Despite the significant research effort in finding alternatives to bypass the need for a trusted third party by constructing *transparent* arguments, i.e. in the uniform random string model (URS) [BBB+18, BCC+16, BBHR18, AHIV17, BBHR19, COS20, XZZ+19, WTs+18], pairing-based SNARKs such as [Gro16] still seem the

most practical alternative in many settings due to their very fast verification, which is essential in many blockchain applications. On the other hand, multiparty solutions for generating the SRS [BCG+15, BGG19, BGM17] are not fully scalable as they require all parties to be online during the entire execution of an extensive procedure [BGM17, KMSV21].

In 2018 Groth et al. [GMMM18] introduced the concept of *universal and updatable* structured reference string (SRS). An SRS is *updatable* if many parties in a non-interactive and verifiable way can contribute to its generation, namely, can update it at any time. Such an SRS has the advantage that, if at least one party behaves honestly and does not reveal its contribution to the simulation trapdoor, the latter is safe. *Universal* captures the fact that the SRS can be used for any (universal) argument with some fixed bound on size. Furthermore, the authors prove that any SRS that consists solely of monomials, meaning powers of some elements given in a group where the discrete logarithm problem is difficult to solve, is updatable. Also, such an SRS is universal, as contains only uniformly sampled elements and no information about the statement.

Even though the concept of updatability was of immense interest as an alternative for expensive MPC procedures, four years later we can say that it is the universality of the SRS that the cryptographic community cares the most about nowadays. In fact, using Multi-Party Computation to generate the parameters seems to be enough at a security level, even though performing such a procedure requires prohibitive amounts of computational and logistical resources. For this reason, there is a huge interest in the cryptography community, and especially in those who pursue practical solutions, in creating protocols that can reuse already existing structured reference strings.

Nowadays, universal structured reference strings (computed through MPC ceremonies) are used in a wide spectrum of arguments and one of the main reasons for this is the fact that they have been proven to sufficient for many *commitment schemes*, such as Pedersen Commitments [Ped92](work also in the URS model) and the polynomial commitments by Kate, Zaverucha and Goldberg [KZG10]. Commitment schemes and, in particular, vector commitment schemes [CF13, LY10] are widely used in decentralized settings as they allow to compress big amounts of data in a succinct manner (where the size of the commitment is constant on the size of the data set) and later retrieve pieces of it, in an efficient way. We can see vector commitments as proving systems for a specific relation whose instance includes some previously committed data, following the commit and prove methodology [CLOS02]. The data stored in the vector can be public, such as a set of values corresponding to some range, or private, such as a set of secret keys.

Besides being a middle point on trust models, universal and updatable SRS bring schemes whose efficiency lies between the ones that can be constructed using trusted parameters and using transparent ones. Throughout this thesis, we push forward on the design of cryptographic protocols that, while implemented with a universal

and updatable SRS, achieve similar efficiency to those that use trusted ones.

In this thesis, we tackle aspects of the theory, efficiency and practicality of succinct arguments, considering zkSNARKs for general computations and Vector Commitment schemes. Contributions include cryptographic primitives, frameworks for constructing such systems with highly demanded properties, and protocols whose efficiency is competitive with the state of the art in all aspects, in most the cases over-performing them in many different efficiency measures.

The security of all the protocols presented in this thesis holds under *non-falsifiable* assumptions [Nao03], meaning that even when given the opportunity to interact with a successful adversary, at the end of the interaction we cannot efficiently decide if it has broken the assumption or not. As proven by Gentry and Wichs [GW11], these kinds of assumptions are necessary for the construction of SNARKs, as they do not exist under falsifiable ones.

In particular, we use the Algebraic Group Model (AGM) [FKL18]. When working in the AGM, we restrict to *algebraic* adversaries that upon producing some proof element A, also output a list of coefficients $\mathbf{z}$ such that A is a linear combination of the elements in $\mathbf{z}$ and those encoded in the SRS. Pairing-based arguments rely on commitments to elements for succinctness, and the AGM allows us to *extract* the algebraic objects behind them and work on the information-theoretical component of the proof, using well-known algebraic principles.

In terms of practicality, all our constructions are in the universal and updatable SRS model. Furthermore, they can be instantiated with any already existing SRS that contains the powers of some secret element . In particular, we state that all the protocols presented below can reuse trusted setups such as "powers of tau" that were run for pairing-based SNARK schemes used in real-world applications, such as the one used by ZCash [ZCary], or Filecoin [Fil20].

The results of this thesis are presented as follows:

- In Chapter 3, we focus on building proving systems for the NP-complete problem of circuit satisfiability that can compete in performance with the best constructions that use a relation-dependent SRS. We design a framework for building zkSNARKs, that is, the prover does not leak information about the witness it uses to perform the proof. Our framework is modular and starts from the design of what we call Checkable Subspace Sampling arguments. We present several constructions of these schemes that lead to many alternatives for different efficiency measures on zkSNARKs.

- In Chapter 4 we focus on Vector Commitments, a primitive that has become crucial in the design of succinct proofs. We design a framework to construct such schemes from already existing primitives and present our own instantiations. Importantly, this chapter focuses on the practicality and properties

of such schemes for their application to decentralized networks. This requires flexibility for the data, as it may change or be used to perform further computations, and the balance between the prover work and the memory they use, as they have limited time and storage resources.

- Finally, in Chapter 5 we analyze how we can exploit the conciseness of vector commitments to store public data while still being able to link it, in zero-knowledge, with smaller pieces of information. That is, construct arguments to prove that some set of hidden data has been taken from a public one. We formalize these arguments and present a construction that performs better than the state of the art, with applications that include membership proofs, proofs of ownership and lookup tables, among others.

Chapter 2

Preliminaries

For $m \in \mathbb{N}$, $[m]$ denotes the set of integers $\{1, \ldots, m\}$. Vectors and matrices are denoted in boldface. Given two vectors $\mathbf{a}, \mathbf{b}$, their Hadamard product is denoted as $\mathbf{a} \circ \mathbf{b}$, and their inner product as $\mathbf{a} \cdot \mathbf{b}$ or $\mathbf{a}^\top \mathbf{b}$. The subspace of polynomials of degree at most d in $\mathbb{F}[X]$ is denoted as $\mathbb{F}_{\leq d}[X]$. Given a matrix $\mathbf{M}$, $|\mathbf{M}|$ denotes the number of its non-zero entries.

2.1 Security proofs

Let $\lambda \in \mathbb{N}$ denote some general security parameter and 1^λ its unary representation. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ is called *negligible* if for all $c > 0$, there exists k_0 such that for all $k > k_0$, $\mathsf{negl}(k) < \frac{1}{k^c}$, and we say a function f is overwhelming if $f(k) > 1 - \mathsf{negl}(k)$ for all $k > k_0$. We will use negligible and overwhelming to refer to the probability of the security of our schemes to fail.

For a non-empty set S, let $x \leftarrow S$ denote sampling an element of S uniformly at random and assigning it to x.

All the algorithms in this work are probabilistic polynomial-time (PPT). Let $y \leftarrow \mathsf{A}(x; r)$ denote running algorithm A on input x and randomness r and assigning its output to y. Note that deterministic polynomial-time algorithms are PPT ones with $r = 0$. Let $y \leftarrow \mathsf{A}(x)$ denote $y \leftarrow \mathsf{A}(x; r)$ for a uniformly random r.

We will represent the probability of events with the notation $\mathsf{Pr}[\text{event} \mid \text{sampling}]$, which equals the probability of the event in the left, given that the inputs have been sampled as explained in the right.

We often used code-based games in security definitions [BR06]. In this framework, we have and adversary $\mathcal{A}$ that plays the security game sec and breaks the

security of some scheme if it has a non-negligible probability of winning. When the same algorithm appears at different stages of a security game, we assume it is stateful, that is, keeps information about all the previous steps it was involved in.

Note that if the protocols involve elements sampled uniformly at random from finite sets, the adversary will have some possibility of winning the security game by guessing the sampled element. The distance between the probability of winning of $\mathcal{A}$ by guessing and the probability of winning with some strategy is what we call the advantage of $\mathcal{A}$ in game Game, and we denote it $\mathsf{Adv}^{\mathsf{Game}}_{\mathcal{A}}$.

A cryptographic scheme is consider *computationally* secure if $\mathsf{Adv}^{\mathsf{sec}}_{\mathcal{A}} \leq \mathsf{negl}(\lambda)$ for all adversaries $\mathcal{A}$ that run in polynomial time in the security parameter. If the security notion holds for unbounded adversaries we say it is *statistically* secure and it is *perfectly* secure if $\mathsf{Adv}^{\mathsf{sec}}_{\mathcal{A}} = 0$ for any $\mathcal{A}$. In order to achieve these bounds, in our security definitions we will often derive from a winning adversary $\mathcal{A}$ an adversary $\mathcal{B}$ against a computationally hard problem prob and prove that $\mathsf{Adv}^{\mathsf{sec}}_{\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{prob}}_{\mathcal{B}}$. That is, if prob is known to be intractable, our protocol satisfies the security notion sec.

2.2 Polynomials

The following well-known lemma will be crucial for proving the security of interactive protocols.

Lemma 1 (Schwartz-Zippel, [Sch80, Zip79], univariate case)**.** *Let $P(X)$ be a polynomial of degree d with coefficients in $\mathbb{F}$, that is $P(X) \in \mathbb{F}_d[X]$. Let $S \subset \mathbb{F}$ be a finite subset. Then,* $\mathsf{Pr}[P(x) = 0 | x \leftarrow S] \leq \frac{d}{|S|}$.

2.2.1 Lagrange Polynomials

Given some finite field $\mathbb{F}$, let $\mathbb{H} = \{\mathsf{h}_1, \ldots, \mathsf{h}_m\}$ be an arbitrary set of cardinal m, with some predefined canonical order, and h_i the ith element in this order. The ith Lagrange basis polynomial associated to $\mathbb{H}$ is denoted by $\lambda_i(X)$. The vector $\boldsymbol{\lambda}(X)$ is defined as $\boldsymbol{\lambda}(X)^\top = (\lambda_1(X), \ldots, \lambda_m(X))$. The vanishing polynomial of $\mathbb{H}$ will be denoted by $z_H(X)$. That is,

$$z_H(X) = \prod_{i=1}^{m}(X - \mathsf{h}_i), \text{ and for every } i \in [m], \quad \lambda_i(X) = \prod_{j \neq i} \frac{(X - \mathsf{h}_j)}{\mathsf{h}_i - \mathsf{h}_j}.$$

We recall that for any set $\mathbb{H}$ of arbitrary size m, the sum of its Lagrange interpolation polynomials is one, that is, $\sum_{i=1}^{m} \lambda_i(X) = 1$.

When $\mathbb{H}$ is a multiplicative subgroup, the following properties are known to hold for all $i \in [m]$:

$$z_H(X) = X^m - 1, \qquad \lambda_i(X) = \frac{\mathsf{h}_i}{m}\frac{(X^m - 1)}{(X - \mathsf{h}_i)}, \qquad \lambda_i(0) = \frac{1}{m},$$

That is, vanishing and Lagrange polynomials corresponding to sets of roots of unity are sparse. This representation makes their computation particularly efficient: both $z_H(X)$ and $\lambda_i(X)$ can be evaluated in $O(\log m)$ field operations. For this reason, we will always consider $\mathbb{H}$ as a multiplicative subgroup, unless the opposite is explicit (only in Section 3.3).

For simplicity, in all the chapters of this thesis we work with a subgroup of roots of unity $\mathbb{H}$ of size m, but for some results we use further multiplicative subgroups, $\mathbb{U}$ and $\mathbb{K}$, whose size and Lagrange and vanishing polynomials will be introduced when necessary.

2.3 Cryptographic Primitives

In this section we introduce the cryptographic primitives, definition and security properties, that will be used as building blocks in the constructions of this thesis. Given that they consist mostly of proving arguments, or include proving phases, some of them have algorithms with the same name, as they have the same task in different context, so we will denote $\mathsf{P.Alg}$ to the algorithm Alg used in the argument P. The only case were we do not do this is for NIZK arguments and in most cases for the KeyGen algorithm, as we want to emphasize that even when we specify a bound on the key generation algorithm, just for tightness, this algorithm can be the same for *all* our schemes.

2.3.1 Non-Interactive Arguments of Knowledge

Let $\mathcal{R}$ be a polynomial time decidable relation. Given instance x we call w a *witness* for x if $(\mathsf{x}, w) \in \mathcal{R}$, $\mathcal{L}(\mathcal{R}) = \{\mathsf{x} |\ \exists w : (\mathsf{x}, w) \in \mathcal{R}\}$ is the language of all the x that have a witness w in the relation $\mathcal{R}$, while $\mathcal{L}(\mathcal{R})$ is the language of all the pairs $(\mathsf{x}, \mathcal{R})$ such that $\mathsf{x} \in \mathcal{L}(\mathcal{R})$. We will assume $\mathcal{R}$ it is implicit as prover and verifier input.

Definition 1. *A Non-Interactive Argument of Knowledge is a tuple of PPT algorithms* $(\mathsf{KeyGen}, \mathsf{Prove}, \mathsf{Verify})$ *such that:*

- $(\mathsf{srs}, \tau) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathcal{R})$*: On input the parameters of the system, a relation* $\mathcal{R}$*,* KeyGen *outputs a structured reference string* srs *and a trapdoor* τ *as a private output;*

- $\pi \leftarrow \mathsf{Prove}(\mathsf{srs}, (\mathsf{x}, w))$*: On input a pair* $(\mathsf{x}, w) \in \mathsf{R}$*, it outputs a proof* π *of the fact that* $\mathsf{x} \in \mathcal{L}(\mathcal{R})$*;*

- $1/0 \leftarrow \mathsf{Verify}(\mathsf{srs}, \mathsf{x}, \pi)$*: On input the srs, the instance* x *and the proof, it produces a bit expressing acceptance* (1)*, or rejection* (0)*;*

and that satisfies completeness, knowledge soundness and zero-knowledge as defined below.

Completeness: *holds if an honest prover will always convince an honest verifier. Formally,* $\forall\ \mathcal{R}, (\mathsf{x}, w) \in \mathcal{R}$*,*

$$\Pr\left[\mathsf{Verify}(\mathsf{srs}, \mathsf{x}, \pi) = 1 \,\middle|\, \begin{array}{c}(\mathsf{srs}, \tau) \leftarrow \mathsf{KeyGen}(\mathcal{R}) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{srs}, (\mathsf{x}, w))\end{array}\right] = 1.$$

Knowledge-Soundness: *captures the fact that a cheating prover cannot, except with negligible probability, create a proof* π *accepted by the verification algorithm unless it has a witness* w *such that* $(\mathsf{x}, w) \in \mathcal{R}$*. Formally, for all PPT adversaries* $\mathcal{A}$*, there exists a PPT extractor* $\mathcal{E}$ *such that*

$$\Pr\left[(\mathsf{x}, w) \notin \mathcal{R} \wedge \mathsf{Verify}(\mathsf{srs}, \mathsf{x}, \pi) = 1 \,\middle|\, \begin{array}{c}(\mathsf{srs}, \tau) \leftarrow \mathsf{KeyGen}(\mathcal{R}) \\ (\mathsf{x}, \pi) \leftarrow \mathcal{A}(\mathsf{srs}) \\ w \leftarrow \mathcal{E}(\mathsf{srs}, \mathsf{x}, \pi)\end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

Zero-Knowledge: *A Non-interactive argument of knowledge is additionally zero-knowledge if there exists an algorithm* Simulate *that behaves as follows*

- $\pi_{\text{sim}} \leftarrow \mathsf{Simulate}(\mathsf{srs}, \tau, \mathsf{x})$*: The simulator has the srs, the trapdoor* τ *and the instance* x *as inputs and it generates a simulated proof* π_{sim}*,*

and for all relations $\mathcal{R}$*, instances* x *and PPT adversaries* $\mathcal{A}$*,*

$$\Pr\left[\mathcal{A}(\mathsf{srs}, \pi) = 1 \,\middle|\, \begin{array}{c}(\mathsf{srs}, \tau) \leftarrow \mathsf{KeyGen}(\mathcal{R}) \\ \mathsf{x} \leftarrow \mathcal{A}(\mathsf{srs}) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{srs}, (\mathsf{x}, w))\end{array}\right] \approx \Pr\left[\mathcal{A}(\mathsf{srs}, \pi_{sim}) = 1 \,\middle|\, \begin{array}{c}(\mathsf{srs}, \tau) \leftarrow \mathsf{KeyGen}(\mathcal{R}) \\ \mathsf{x} \leftarrow \mathcal{A}(\mathsf{srs}) \\ \pi_{sim} \leftarrow \mathsf{Simulate}(\mathsf{srs}, \tau, \mathsf{x})\end{array}\right].$$

Succinctness: *Succinctness holds if the size of the proof π is $poly(\lambda + \log(|\mathsf{w}|))$ and* Verify *runs in time $poly(\lambda + |\mathsf{x}| + \log(|\mathsf{w}|))$. If* $(\mathsf{KeyGen}, \mathsf{Prove}, \mathsf{Verify})$ *satisfies succinctness, we say it is a Succinct, Non-Interactive Argument of Knowledge (SNARK). A zero-knowledge SNARK will be denoted as zkSNARK.*

2.3.2 Universal and Updatable Parameters

Below, we present the definition of Universal and Updatable SRS schemes introduced by Groth et al. in [GKM+18]. All its algorithms together replace KeyGen in a NIZK or SNARK, that is, there is a party that generates the parameters in first place and then many other parties that can update them later on. Importantly, because none of these parties is trusted, they all provide a proof of well formation of its contribution that any entity can check. It will be explained at the end of this Section that in the rest of this thesis we will omit the algorithms that generate or verify such proofs.

For universal structured reference strings, we will no longer set $\mathcal{R}$ to be a relation but rather a family of universal relations. Then, given a specific relation $\mathsf{R} \in \mathcal{R}$, we consider the language $\mathcal{L}(\mathsf{R}) = \{\mathsf{x} | \ \exists w : (\mathsf{x}, w) \in \mathsf{R}\}$ of all the instances that have a witness in the relation R, and the language $\mathcal{L}(\mathsf{R})$ of pairs (x, R) such that $\mathsf{x} \in \mathcal{L}(\mathsf{R})$. We will assume that both, $\mathcal{R}$ and R are implicit as prover and verifier input.

Definition 2. *A Universal and Updatable SRS scheme consists a tuple of PPT algorithms* $(\mathsf{KeyGen}, \mathsf{SRS.Update}, \mathsf{SRS.Verify})$ *that work as follows,*

- $(\mathsf{srs_u}, \tau, \pi_{\mathsf{srs_u}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathcal{R})$*: On input the parameters of the system, a universal relation $\mathcal{R}$,* KeyGen *outputs a universal structured common reference string* $\mathsf{srs_u}$ *(that includes the parameters of the system), a trapdoor τ, and a proof of correctness $\pi_{\mathsf{srs_u}}$;*
- $(\mathsf{srs'_u}, \tau', \pi'_{\mathsf{srs_u}}) \leftarrow \mathsf{SRS.Update}(\mathsf{srs_u}, \{\pi_{\mathsf{srs_u},i}\}_{i=1}^{n})$*: On input a common reference string and a list of update proofs, it outputs a new* $\mathsf{srs_u}$*, a contribution trapdoor τ' (which is a private output), and the corresponding proof of correctness.*
- $1/0 \leftarrow \mathsf{SRS.Verify}(\mathsf{srs_u}, \{\pi_{\mathsf{srs_u},i}\}_{i=1}^{n})$*: On input a common reference string and a list of proofs, produces a bit expressing acceptance (*1*) or rejection* (0)*;*

When using universal and updatable SRSs in proof systems, an extra deterministic algorithm, called Derive and denoted as KeyGenD is usually included. This algorithm takes as input the universal SRS constructed for $\mathcal{R}$ and a relation $\mathsf{R} \in \mathcal{R}$ and outputs an *specific* srs that both prover and verifier have as input. The aim of this algorithm is to provide a succinct description of the circuit to the verifier, so it can run in time less than linear. Importantly, this step is what allows universal and updatable SNARKs to achieve verifier costs comparable to those that use a trusted

setup. Actually, as mentioned in the seminal work [GKM+18], the derive step can be run as part of the proving and verifying algorithms if needed, but it is useful to think about it as some pre-processing step of the prover algorithm.

- $\mathsf{srs}_{\mathsf{R}} \leftarrow \mathsf{KeyGenD}(\mathsf{srs}_{\mathsf{u}}, \mathsf{R})$: *On input* $\mathsf{R} \in \mathcal{R}$*, this algorithm outputs a relation dependent srs that includes* $\mathsf{srs}_{\mathsf{u}}$*;*

Note that universal and updatable srs schemes as defined above can be seen as a NIZK argument of the well formation of the parameters, that is, a NIZK for the following relation:

$$\mathsf{R}_{\mathsf{srs}_{\mathsf{u}}} = \left\{(\mathsf{srs}_{\mathsf{u}}, \hat{\mathsf{srs}}_{\mathsf{u}}, \tau, \{\pi_{\mathsf{srs}_{\mathsf{u}},i}\}_{i=1}^{n}, \hat{\pi}_{\mathsf{srs}_{\mathsf{u}}}) : (\hat{\mathsf{srs}}_{\mathsf{u}}, \tau, \hat{\pi}_{\mathsf{srs}_{\mathsf{u}}}) \leftarrow \mathsf{SRS.Update}(\mathsf{srs}_{\mathsf{u}}, \{\pi_{\mathsf{srs}_{\mathsf{u}},i}\}_{i=1}^{n})\right\}$$

Universal and updatable SRS schemes must satisfy correctness and trapdoor extraction as defined below.

Completeness: *An updatable SRS scheme satisfies perfect completeness if both of the following probabilities are* 1*.*

$$\Pr\left[\mathsf{SRS.Verify}(\mathsf{srs}_{\mathsf{u}}, \pi_{\mathsf{srs}_{\mathsf{u}}}) = 1 \,\middle|\, (\mathsf{srs}_{\mathsf{u}}, \tau, \pi_{\mathsf{srs}_{\mathsf{u}}}) \leftarrow \mathsf{KeyGen}(1^{\lambda})\right]$$

and

$$\Pr\left[\mathsf{SRS.Verify}(\mathsf{srs}_{\mathsf{u}}, \{\pi_{\mathsf{srs}_{\mathsf{u}},i}\}_{i=1}^{n}) = 1 \,\middle|\, \begin{array}{c}(\mathsf{srs}_{\mathsf{u}}, \tau, \pi_{\mathsf{srs}_{\mathsf{u}}}) \leftarrow \mathsf{KeyGen}(1^{\lambda}) \\ (\mathsf{srs}'_{\mathsf{u}}, \tau', \pi'_{\mathsf{srs}_{\mathsf{u}}}) \leftarrow \mathsf{SRS.Update}(\mathsf{srs}_{\mathsf{u}}, \{\pi_{\mathsf{srs}_{\mathsf{u}},i}\}_{i=1}^{n})\end{array}\right]$$

The notion of trapdoor extraction can be seen as *knowledge soundness* of the adversary's contribution to the srs. That is, if an adversary has created an srs from scratch, it must know the trapdoor τ it has used for it. Similarly, if it outputs a valid update to an existing srs, it must be the case that the update has been done correctly, and the adversary knows the contribution it has made.

Trapdoor Extraction : *An updatable SRS scheme satisfies Trapdoor Extraction for subvertible SRSs if for all PPT adversaries* $\mathcal{A}$ *there exists an PT extractor* $\mathcal{E}$ *such that:*

$$\Pr\left[\begin{array}{c}\mathsf{SRS.Verify}(\mathsf{srs}_{\mathsf{u}}, \pi_{\mathsf{srs}_{\mathsf{u}}}) = 1 \\ \wedge\ \exists\ \tau'\ s.t.\ (\mathsf{srs}_{\mathsf{u}}, \tau', \pi_{\mathsf{srs}_{\mathsf{u}}}) \leftarrow \mathsf{KeyGen}(1^{\lambda}) \\ \wedge\ \tau' \neq \tau\end{array} \,\middle|\, \begin{array}{c}(\mathsf{srs}_{\mathsf{u}}, \pi_{\mathsf{srs}_{\mathsf{u}}}) \leftarrow \mathcal{A}(1^{\lambda}) \\ \tau \leftarrow \mathcal{E}(\mathsf{srs}_{\mathsf{u}}, \pi_{\mathsf{srs}_{\mathsf{u}}})\end{array}\right] \leq \mathsf{negl}(\lambda)$$

An updatable SRS scheme satisfies Trapdoor Extraction for updatable SRSs if the following probability is negligible in λ:

$$\Pr\left[\begin{array}{c} \mathsf{SRS.Verify}(\hat{\mathsf{srs}}_{\mathsf{u}}, \hat{\pi}_{\mathsf{srs}_{\mathsf{u}}}) = 1 \\ \wedge\ (\mathsf{srs}'_{\mathsf{u}}, \hat{\mathsf{srs}}_{\mathsf{u}}, \tau, \{\pi_{\mathsf{srs}_{\mathsf{u}},i}\}_{i=1}^{n}, \pi'_{\mathsf{srs}_{\mathsf{u}}}) \notin \mathsf{R}_{\mathsf{srs}_{\mathsf{u}}} \end{array} \middle| \begin{array}{c} (\mathsf{srs}_{\mathsf{u}}, \tau, \pi_{\mathsf{srs}_{\mathsf{u}}}) \leftarrow \mathsf{KeyGen}(1^{\lambda}) \\ (\mathsf{srs}'_{\mathsf{u}}, \pi'_{\mathsf{srs}_{\mathsf{u}}}) \leftarrow \mathsf{SRS.Update}(\mathsf{srs}_{\mathsf{u}}, \{\pi_{\mathsf{srs}_{\mathsf{u}},i}\}_{i=1}^{n}) \\ (\hat{\mathsf{srs}}_{\mathsf{u}}, \hat{\pi}_{\mathsf{srs}_{\mathsf{u}}}) \leftarrow \mathcal{A}(\mathsf{srs}'_{\mathsf{u}}, \pi'_{\mathsf{srs}_{\mathsf{u}}}) \\ \tau \leftarrow \mathcal{E}(\hat{\mathsf{srs}}_{\mathsf{u}}, \hat{\pi}_{\mathsf{srs}_{\mathsf{u}}}) \end{array}\right]$$

It is proven in [GKM+18] that a SRS that consists solely of monomials is updatable and satisfies trapdoor extraction in both cases. In the rest of this work, we will use setups that contain only monomials and thus omit the description of the srs.Update and srs.Verify algorithms, as well as the security proofs.

2.3.3 Polynomial Commitments

Polynomial Commitment schemes have been introduced by Kate, Zaverucha and Goldberg [KZG10]. In the same work, they present a construction based in bilinear groups, with constant size proof and verifier work, which is broadly used nowadays and we will present in Section 2.5.1.

Definition 3 (Polynomial Commitment Scheme)**.** *A* Polynomial Commitment Scheme *is a tuple of algorithms* $\big(\mathsf{KeyGen}, \mathsf{PC.Commit}, \mathsf{PC.Open}, \mathsf{PC.Verify}\big)$ *such that:*

- $(\mathsf{srs}, \tau) \leftarrow \mathsf{KeyGen}\big(\mathsf{pp}, d\big)$*: On input the system parameters and a degree bound d, it outputs a structured reference string and trapdoor τ.*
- $\mathsf{C} \leftarrow \mathsf{PC.Commit}\big(\mathsf{srs}, p(X), r\big)$*: On input the srs and a polynomial $p(X)$, and randomness r it outputs a commitment C to $p(X)$.*
- $(s, \pi) \leftarrow \mathsf{PC.Open}\big(\mathsf{srs}, p(X), r, \alpha\big)$*: On input the srs, the polynomial, commitment randomness r, a query point $\alpha \in \mathbb{F}$, it outputs $s \in \mathbb{F}$ and an evaluation proof π that $s = p(\alpha)$ and $deg(p) = \deg$[1].*
- $1/0 \leftarrow \mathsf{PC.Verify}\big(\mathsf{srs}, \mathsf{C}, \deg, \alpha, s, \pi\big)$*: On input the srs, the commitment, degree bound, query and evaluation points α, s, and the proof of correct evaluation, it outputs a bit indicating acceptance or rejection.*

A polynomial commitment scheme should satisfy the following properties:

[1]Note that d denotes the degree bound used by the system, while deg denotes the claim value for $deg(p)$

Completeness: *It captures the fact that an honest prover will always convince an honest verifier. Formally, for any polynomial $p(X)$ such that $deg(p) \leq d$ and query point $\alpha \in \mathbb{F}$,*

$$\mathsf{Pr}\left[\mathsf{PC.Verify}\big(\mathsf{srs}, \mathsf{C}, \deg, \alpha, s, \pi\big) = 1 \;\middle|\; \begin{array}{c} (\mathsf{srs}, \tau) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, d) \\ \mathsf{C} \leftarrow \mathsf{PC.Commit}\big(\mathsf{srs}, p(X), r\big) \\ s = p(\alpha), \deg(p) = \deg \\ (s, \pi) \leftarrow \mathsf{PC.Open}\big(\mathsf{srs}, p(X), r, \alpha\big) \end{array}\right] = 1.$$

Soundness: *Captures the fact that a cheating prover should not be able to convince the verifier of a false opening. Formally, for all stateful PPT adversaries $\mathcal{A}$:*

$$\mathsf{Pr}\left[\begin{array}{c} \big(p(\alpha) \neq s \;\vee\; deg(p) > \deg\big) \\ \wedge \\ \mathsf{PC.Verify}\big(\mathsf{srs}, \mathsf{C}, \deg, \alpha, s, \pi\big) = 1 \end{array} \;\middle|\; \begin{array}{c} (\mathsf{srs}, \tau) \leftarrow \mathsf{KeyGen}\big(\mathsf{pp}, d\big) \\ (p(X), \mathsf{C}) \leftarrow \mathcal{A}(\mathsf{srs}) \\ \alpha \leftarrow \mathbb{F} \\ (s, \pi) \leftarrow \mathcal{A}(\alpha) \end{array}\right] \approx 0$$

Evaluation Binding: *Captures the fact that no PPT adversary $\mathcal{A}$ should be able to present two valid openings for different values but same evaluation point. Formally:*

$$\mathsf{Pr}\left[\begin{array}{c} \mathsf{PC.Verify}\big(\mathsf{srs}, \mathsf{C}, \deg, \alpha, s, \pi\big) = 1 \\ \mathsf{PC.Verify}\big(\mathsf{srs}, \mathsf{C}, \deg, \alpha, s', \pi'\big) = 1 \\ \text{and } s \neq s' \end{array} \;\middle|\; \begin{array}{c} (\mathsf{srs}, \tau) \leftarrow \mathsf{KeyGen}\big(\mathsf{pp}, d\big) \\ (\mathsf{C}, \alpha, s, s', \pi, \pi') \leftarrow \mathcal{A}\big(\mathsf{srs}\big) \end{array}\right] \approx 0$$

Additionally, we will ask polynomial commitments to satisfy extractability.

Extractability: *Captures the fact that whenever the prover provides a valid opening, it knows a valid pair $(p(X), p(\alpha)) \in \mathbb{F}[X] \times \mathbb{F}$, where $deg(p) \leq \deg$. Formally, for all PPT adversaries $\mathcal{A}$ there exists an efficient extractor $\mathcal{E}$ such that:*

$$\mathsf{Pr}\left[\begin{array}{c} \mathsf{PC.Verify}\big(\mathsf{srs}, \mathsf{C}, \deg, \alpha, s, \pi\big) = 1 \\ \wedge \\ \big(p(\alpha) \neq s \;\vee\; deg(p) > \deg\big) \end{array} \;\middle|\; \begin{array}{c} (\mathsf{srs}, \tau) \leftarrow \mathsf{KeyGen}\big(\mathsf{pp}, \deg\big) \\ \mathsf{C} \leftarrow \mathcal{A}\big(\mathsf{srs}\big) \\ p(X) \leftarrow \mathcal{E}\big(\mathsf{srs}, \mathsf{C}, \deg\big) \\ \alpha \leftarrow \mathcal{A}\big(\mathsf{srs}, \mathsf{C}, \deg\big) \\ (s, \pi) \leftarrow \mathcal{A}\big(\mathsf{srs}, p(X), \deg, \alpha\big) \end{array}\right] \approx 0$$

2.3.4 Vector Commitments

In this section we provide the classical definitions of vector commitments (VC), introduced by Catalano and Fiore [CF13].

Definition 4 (Vector Commitment Scheme)**.** *A* Vector Commitment Scheme *is a tuple of algorithms* $(\mathsf{KeyGen}, \mathsf{VC.Commit}, \mathsf{VC.Open}, \mathsf{VC.Verify})$ *such that:*

- $(\mathsf{srs}, \tau) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, d)$*: On input the system parameters and a bound* d *on the size of the vectors, it outputs a structured reference string* srs *consisting on a prover key* prk *and a verifier key* vrk*, and trapdoor* τ*.*
- $(\mathsf{C}, \mathsf{aux}) \leftarrow \mathsf{VC.Commit}(\mathsf{srs}, \mathbf{v}, r)$*: On input the srs, a vector* $\mathbf{v}$*, and randomness* r *it outputs a commitment* C *and auxiliary information* aux*.*
- $\pi \leftarrow \mathsf{VC.Open}(\mathsf{srs}, \mathbf{v}, r, i)$*: On input the srs, the vector, its size, the commitment randomness, and a position* i *it outputs* $v_i \in \mathbb{F}$ *and proof* π *that* v_i *is the* i*th element of vector* $\mathbf{v}$*.*
- $1/0 \leftarrow \mathsf{VC.Verify}(\mathsf{srs}, \mathsf{C}, i, v_i, \pi)$*: On input the srs, the commitment, position, claimed value* v_i*, and the proof, it outputs a bit indicating acceptance or rejection.*

A vector commitment scheme should satisfy the following properties:

Correctness: *It captures the fact that an honest prover will always convince an honest verifier. Namely, for all vectors* $\mathbf{v} \in \mathbb{F}^N$ *and* $i \in [N]$

$$\Pr\left[\mathsf{VC.Verify}(\mathsf{srs}, \mathsf{C}, i, v_i, \pi) = 1 \;\middle|\; \begin{array}{c} (\mathsf{srs}, \tau) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, d) \\ (\mathsf{C}, \mathsf{aux}) \leftarrow \mathsf{VC.Commit}(\mathsf{srs}, \mathbf{v}, r) \\ \pi \leftarrow \mathsf{VC.Open}(\mathsf{srs}, \mathbf{v}, r, i) \end{array}\right] = 1.$$

(Weak) Position Binding: *Captures the fact that no PPT adversary* $\mathcal{A}$ *should be able to present for one commitment two valid openings for the same position. Formally:*

$$\Pr\left[\begin{array}{c} \mathsf{VC.Verify}(\mathsf{srs}, \mathsf{C}, i, y, \pi) = 1, \\ \mathsf{VC.Verify}(\mathsf{srs}, \mathsf{C}, i, y', \pi') = 1 \\ \text{and } y \neq y' \end{array} \;\middle|\; \begin{array}{c} (\mathsf{srs}, \tau) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, d) \\ (\mathbf{v}, r, i, y, y', \pi, \pi') \leftarrow \mathcal{A}(\mathsf{srs}) \\ (\mathsf{C}, \mathsf{aux}) \leftarrow \mathsf{VC.Commit}(\mathsf{srs}, \mathbf{v}, r) \end{array}\right] \approx 0.$$

(Strong) Position Binding: *Captures the fact that no PPT adversary $\mathcal{A}$ should be able to present for one commitment two valid openings for the same position. Formally:*

$$\Pr\left[\begin{array}{c}\mathsf{VC.Verify}\big(\mathsf{srs},\mathsf{C},i,y,\pi\big)=1,\\ \mathsf{VC.Verify}\big(\mathsf{srs},\mathsf{C},i,y',\pi'\big)=1\\ \text{and } y\neq y'\end{array}\middle|\begin{array}{c}(\mathsf{srs},\tau)\leftarrow\mathsf{KeyGen}\big(\mathsf{pp},d\big)\\ (\mathsf{C},i,y,y',\pi,\pi')\leftarrow\mathcal{A}\big(\mathsf{srs}\big)\end{array}\right]\approx 0.$$

Additionally, we can ask a Vector Commitment scheme to satisfy the following properties:

Hiding: *Captures the fact that the commitment should not leak information about the vector* $\mathbf{v}$*. Formally:*

$$\Pr\left[b=b'\middle|\begin{array}{c}(\mathsf{srs},\tau)\leftarrow\mathsf{KeyGen}\big(\mathsf{pp},d\big)\\ (\mathbf{v}_1,\mathbf{v}_2)\leftarrow\mathcal{A}\big(\mathsf{srs}\big)\\ (\mathsf{C}_b,\mathsf{aux})\leftarrow\mathsf{VC.Commit}(\mathsf{srs},\mathbf{v}_b;r)\\ b'\leftarrow\mathcal{A}\end{array}\right]=\frac{1}{2}+\mathsf{negl}\,(\lambda)\,.$$

Extractability: *Captures the fact that whenever the prover provides a valid opening, it knows a valid pair* $(\mathbf{v},y)\in\mathbb{F}^d\times\mathbb{F}$*, where* $v_i=y$*. Formally, for all PPT adversaries* $\mathcal{A}$ *there exists an efficient extractor* $\mathcal{E}$ *such that:*

$$\Pr\left[\begin{array}{c}\mathsf{VC.Verify}\big(\mathsf{srs},\mathsf{C},i,y,\pi\big)=1\\ \wedge\ v_i\neq y\end{array}\middle|\begin{array}{c}(\mathsf{srs},\tau)\leftarrow\mathsf{KeyGen}\big(\mathsf{pp},d\big)\\ \mathsf{C}\leftarrow\mathcal{A}\big(\mathsf{srs}\big)\\ \mathbf{v}\leftarrow\mathcal{E}\big(\mathsf{srs},\mathsf{C},d\big)\\ (i,y,\pi)\leftarrow\mathcal{A}\big(\mathsf{srs},\mathbf{v},d,i\big)\end{array}\right]\approx 0.$$

Remark. *Note that for vector commitments (and later for Linear-map Vector Commitments) the* srs *consists on two different keys, one for the prover and one for the verifier. We will consider them separately in Chapter 4 to highlight the difference between the information prover and verifier need access to. We stress that the* srs *is still universal and dependent only in the scheme and the size of the admitted vectors.*

Subvector Commitments

Subvector commitment schemes have been defined for first time in two independent works [BBF19, LM19]. Still, in this thesis we define Subvector Openings as a additional property of already existing Vector Commitments. That is, we add additional algorithms and security properties to an existing VC.

Definition 5 (Sub-Vector Commitment)**.** *A Sub-Vector Commitment scheme is a VC scheme that opens subsets rather than positions. It consists on algorithms* $\big(\mathsf{KeyGen},\mathsf{SVC.Commit},\mathsf{SVC.Open},\mathsf{SVC.Verify}\big)$ *that work as follows:*

- $\pi_I \leftarrow \mathsf{SVC.Open}(\mathsf{prk}, \mathsf{aux}, I, \mathbf{v}_I)$: *Takes as input* prk, aux, *a set of index* $I \subset [m]$ *and values* $\mathbf{v}_I = \{v_i\}_{i \in I}$ *and outputs a proof* π_I *that* v_i *is the value in position* i, *for all* $i \in I$.

- $b \leftarrow \mathsf{SVC.Verify}(\mathsf{vrk}, \mathsf{C}, I, \mathbf{y}, \pi_I)$: *Takes as input* vrk, C, I, *a vector* $\mathbf{y} = \{y_i\}_{i \in I}$ *and* π_I. *It outputs 1 for accept or 0 for reject.*

Correctness: *An SVC scheme is* perfectly correct *if, for all* $\lambda \in$, *any vector length* $m = \mathsf{poly}(\lambda)$, *any index set* $I \subset [m]$, *and any* $\mathbf{v} \in \mathcal{M}^m$,

$$\Pr\left[\mathsf{SVC.Verify}(\mathsf{vrk}, C, I, \mathbf{v}_I, \pi_I) = 1 \;\middle|\; \begin{array}{c}(\mathsf{prk}, \mathsf{vrk}) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathcal{M}, m) \\ (\mathsf{C}, \mathsf{aux}) \leftarrow \mathsf{VC.Commit}(\mathsf{prk}, \mathbf{v}) \\ \pi_I \leftarrow \mathsf{SVC.Open}(\mathsf{prk}, \mathsf{aux}, I, \mathbf{v}_I)\end{array}\right] = 1.$$

Binding: *Binding captures the impossibility of creating inconsistent openings for subvectors. An SVC scheme satisfies strong position binding if, for all* PPT *adversaries* $\mathcal{A}$, *for all* $\lambda \in$, *any vector length* $m = \mathsf{poly}(\lambda)$, *the following probability is negligible in* λ*:*

$$\Pr\left[\begin{array}{c}\mathsf{SVC.Verify}(\mathsf{vrk}, \mathsf{C}, I, \mathbf{y}, \pi_I) = 1 \wedge \\ \mathsf{SVC.Verify}(\mathsf{vrk}, \mathsf{C}, J, \mathbf{y}', \pi_J) = 1 \\ \wedge\, \exists i \in I \cap J \text{ s.t. } y_i \neq y'_i\end{array} \;\middle|\; \begin{array}{c}(\mathsf{prk}, \mathsf{vrk}) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathcal{M}, m) \\ \left(\begin{array}{c}\mathsf{C}, I, J, \\ \mathbf{y}, \pi_I, \mathbf{y}', \pi_J\end{array}\right) \leftarrow \mathcal{A}(\mathsf{prk}, \mathsf{vrk})\end{array}\right].$$

Weak Position Binding is considering the definition above to hold only for honestly-generated *commitments* C *computed via* $\mathsf{VC.Commit}$.

2.3.5 Linear-map Vector Commitment

In the following, we define what we call Linear-map Vector Commitments (LVC) schemes. Notably, this definition has been introduced by Lai and Malavolta in [LM19] (except that there the name is *Linear Map Commitments*) to capture further functionalities of vector commitments, whose definition before only account for proofs of *position openings* (Vector Commitments) or more generally *subvector openings* (Sub-vector commitments). We introduce the definition and security properties of LVC.

Definition 6. *[Linear-Map Vector Commitments (LVC)] A linear-map vector commitment scheme for function families* $\mathcal{F} \subset \{f : \mathcal{M}^m \to \mathcal{M}^n\}$ *is a tuple of PPT algorithms* $\big(\mathsf{LVC.KeyGen}, \mathsf{LVC.Commit}, \mathsf{LVC.Open}, \mathsf{LVC}, \mathsf{Vf}\big)$ *that work as follows:*

- $(\mathsf{prk}, \mathsf{vrk}) \leftarrow \mathsf{LVC.KeyGen}(1^\lambda, \mathcal{F})$*: The setup algorithm takes the security parameter* λ*, a family of functions* $\mathcal{F}$ *implicitly defining the message space* $\mathcal{M}$*, and the maximum vector length* $m = \mathsf{poly}(\lambda)$*, and outputs a pair of keys* $(\mathsf{prk}, \mathsf{vrk})$*.*

- $(\mathsf{C}, \mathsf{aux}) \leftarrow \mathsf{LVC.Commit}(\mathsf{prk}, \mathbf{v})$*: On input the proving key* prk*, and a vector* $\mathbf{v} = (v_1, v_2 \ldots, v_m) \in \mathcal{M}^m$*, returns a commitment* C *and auxiliary information* aux*. This algorithm is deterministic.*

- $\pi_f \leftarrow \mathsf{LVC.Open}(\mathsf{prk}, \mathsf{aux}, f, \mathbf{y})$ *: Takes as input* prk*, the auxiliary information* aux*, a function* $f \in \mathcal{F}$*, and a claimed result* $\mathbf{y} \in \mathcal{M}^n$*. It outputs a proof* π_f *that* $f(\mathbf{v}) = \mathbf{y}$*.*

- $b \leftarrow \mathsf{LVC.Vf}(\mathsf{vrk}, \mathsf{C}, f, \mathbf{y}, \pi_f)$ *: Takes as input the verification key* vrk*,* C*, function* f*,* $\mathbf{y} \in \mathcal{M}^n$*, and proof* π_f*. It accepts or rejects.*

An LVC scheme must satisfy the following properties:

Correctness: *An* LVC *scheme is* perfectly correct *if for all* $\lambda \in$*, for any family of functions* $\mathcal{F} \subset \{f : \mathcal{M}^m \to \mathcal{M}^n\}$ *and any* $\mathbf{v} \in \mathcal{M}^m$*,*

$$\Pr\left[\mathsf{LVC.Vf}(\mathsf{vrk}, \mathsf{C}, f, \mathbf{y}, \pi_f) = 1 \;\middle|\; \begin{array}{c}(\mathsf{prk}, \mathsf{vrk}) \leftarrow \mathsf{LVC.KeyGen}(1^\lambda, \mathcal{F}) \\ (C, \mathsf{aux}) \leftarrow \mathsf{LVC.Commit}(\mathsf{prk}, \mathbf{v}) \\ \pi_f \leftarrow \mathsf{LVC.Open}(\mathsf{prk}, \mathsf{aux}, f, \mathbf{y})\end{array}\right] = 1.$$

(Strong) Function Binding: *A linear map commitment* LVC *satisfies strong function binding if, for any* PPT *adversary* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$*, for all integers* $K \in \mathsf{poly}(\lambda)$*, and for any family of functions* $\mathcal{F}$*, the following probability is negligible in* λ*:*

$$\Pr\left[\begin{array}{c}\forall k \in [K]: \\ \mathsf{LVC.Vf}(\mathsf{vrk}, \mathsf{C}, f_k, \mathbf{y}_k, \pi_{f_k}) = 1 \\ \wedge \;\; \nexists\, \mathbf{v} \in \mathcal{M}^m \;\; s.\;\; t. \\ \forall k \in [K]: \;\; f_k(\mathbf{v}) = \mathbf{y}_k\end{array} \;\middle|\; \begin{array}{c}(\mathsf{prk}, \mathsf{vrk}) \leftarrow \mathsf{LVC.KeyGen}(1^\lambda, \mathcal{F}) \\ \left(\mathsf{C}, \{f_k, \mathbf{y}_k, \pi_{f_k}\}_{k \in [K]}\right) \leftarrow \mathcal{A}(\mathsf{prk}, \mathsf{vrk})\end{array}\right]$$

The definition above can be relaxed to hold only for honestly-generated *commitments* C*, raising to the* weak function binding *notion. In the weak definition, the adversary* $\mathcal{A}$ *returns a vector* $\mathbf{v}$ *while the commitment* C *is computed via* $\mathsf{LVC.Commit}$*. In this work, constructions are proven strong function binding.*

2.3.6 Homomorphic Properties for LVC

Linear-map vector commitment schemes that satisfy *homomorphic commitments* allow to combine commitments of two vectors into a single one of their sum(or any linear combination). Namely, for all λ, and $(\mathsf{vrk}, \mathsf{prk}) \leftarrow \mathsf{LVC.KeyGen}(1^\lambda, \mathcal{F})$, if $(\mathsf{C}_1, \mathsf{aux}_1) \leftarrow \mathsf{LVC.Commit}(\mathsf{prk}, \mathbf{v}_1)$ and $(\mathsf{C}_2, \mathsf{aux}_2) \leftarrow \mathsf{LVC.Commit}(\mathsf{prk}, \mathbf{v}_2)$, then $\tilde{\mathsf{C}} = (\alpha\mathsf{C}_1 + \beta\mathsf{C}_2)$ is a valid commitment to $\tilde{\mathbf{v}} = (\alpha\mathbf{v}_1 + \beta\mathbf{v}_2)$ for any $\alpha, \beta \in \mathcal{M}$.

In this work, we are particularly interested in LVC that also have *homomorphic proofs* for different functions applied to a committed vector and *homomorphic openings* for the same function applied to different initial vectors.

Homomorphic Proofs. An LVC scheme has *homomorphic proofs* if it allows recombine two proofs π_1, π_2 corresponding to linear maps f_1, f_2 into a new proof $\tilde{\pi}$ that opens to a linear combination of f_1 and f_2 applied to the same committed vector. Namely, for all λ, $\mathcal{F} \subset \{f : \mathcal{M}^m \to \mathcal{M}^n\}$ and all vectors $\mathbf{v} \in \mathcal{M}^m$, and $(\mathsf{vrk}, \mathsf{prk}) \leftarrow \mathsf{LVC.KeyGen}(1^\lambda, \mathcal{F})$, $(\mathsf{C}, \mathsf{aux}) \leftarrow \mathsf{LVC.Commit}(\mathsf{prk}, \mathbf{v})$, if $\pi_1 \leftarrow \mathsf{LVC.Open}(\mathsf{prk}, \mathsf{aux}, f_1, \mathbf{y}_1)$ and $\pi_2 \leftarrow \mathsf{LVC.Open}(\mathsf{prk}, \mathsf{aux}, f_2, \mathbf{y}_2)$, then for all $\alpha, \beta \in \mathcal{M}$:
$\tilde{\pi} = (\alpha\pi_1 + \beta\pi_2)$ verifies $\mathsf{LVC.Vf}\big(\mathsf{vrk}, \mathsf{C}, \tilde{f} = (\alpha f_1 + \beta f_2), \tilde{\mathbf{y}} = (\alpha\mathbf{y}_1 + \beta\mathbf{y}_2), \tilde{\pi}\big) = 1$.

Homomorphic Openings. An LVC scheme has *homomorphic openings* if we can combine opening proofs for the same linear-map f applied to two different vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ to obtain a new proof of opening $\tilde{\pi}$ that verifies with respect to the linear combination $\tilde{\mathsf{C}}$ of the two initial commitments $\mathsf{C}_1, \mathsf{C}_2$ and show the result of $\mathbf{f}$ applied to the linear combination of the vectors $\mathbf{v}_1$ and $\mathbf{v}_2$.

More formally, for all λ, $\mathcal{F} \subset \{f : \mathcal{M}^m \to \mathcal{M}^n\}$, vectors $\mathbf{v}_1, \mathbf{v}_2 \in \mathcal{M}^m$, and $(\mathsf{vrk}, \mathsf{prk}) \leftarrow \mathsf{LVC.KeyGen}(1^\lambda, \mathcal{F})$, if $\pi_1 \leftarrow \mathsf{LVC.Open}(\mathsf{prk}, \mathsf{aux}_1, f, \mathbf{y}_1)$ and $\pi_2 \leftarrow \mathsf{LVC.Open}(\mathsf{prk}, \mathsf{aux}_2, f, \mathbf{y}_2)$, where $(\mathsf{C}_1, \mathsf{aux}_2) \leftarrow \mathsf{LVC.Commit}(\mathsf{prk}, \mathbf{v}_1)$ and $(\mathsf{C}_2, \mathsf{aux}_2) \leftarrow \mathsf{LVC.Commit}(\mathsf{prk}, \mathbf{v}_2)$, then for all $\alpha, \beta \in \mathcal{M}$:

$\tilde{\pi} = (\alpha\pi_1 + \beta\pi_2)$ verifies $\mathsf{LVC.Vf}\big(\mathsf{vrk}, \tilde{\mathsf{C}} = (\alpha\mathsf{C}_1 + \beta\mathsf{C}_2), f, \tilde{\mathbf{y}} = (\alpha\mathbf{y}_1 + \beta\mathbf{y}_2), \tilde{\pi}\big) = 1$.

2.3.7 Polynomial Holographic Proofs

Following previous works [GWC19, CHM+20, CFF+21], in chapter 3 we will construct zkSNARKs by building first an information theoretically secure argument and then compiling it using a polynomial commitment scheme. For the latter, we will use schemes satisfying the properties presented in Definition 3 and for the former we use the notion of Polynomial Holographic Interactive Oracle Proofs (PHP), introduced by Campanelli et al. [CFF+21]. It is a refinement and quite similar to other notions used in the literature to construct SNARKs in a modular way, such as

Algebraic Holographic Proofs (AHP) [CHM+20] or idealized polynomial protocols [GWC19].

A proof system for a relation R is holographic if the verifier does not read the full description of the relation, but rather has access to an encoding of the statement produced by some holographic relation encoder, also called indexer, that outputs oracle polynomials. In all these models, the prover is restricted to send oracle polynomials or field elements, except that, for additional flexibility, the PHP model of [CFF+21] also lets the prover send arbitrary messages. In PHPs, the queries of the verifier are algebraic checks over the polynomials sent by the verifier, as opposed to being limited to polynomial evaluations as in AHPs.

The following definitions are taken almost verbatim from [CFF+21].

Definition 7. *A family of polynomial time computable relations $\mathcal{R}$ is field dependent if each relation $\mathsf{R} \in \mathcal{R}$, specifies a unique finite field. More precisely, for any pair $(\mathsf{x}, \mathsf{w}) \in \mathsf{R}$, x specifies the same finite field $\mathbb{F}_{\mathsf{R}}$ (simply denoted as $\mathbb{F}$ if there is no ambiguity).*

Definition 8. *[Polynomial Holographic IOPs (PHP)] A Polynomial Holographic IOP for a family of field-dependent relations $\mathcal{R}$ is a tuple $\mathsf{PHP} = (\mathsf{rnd}, \mathsf{n}, \mathsf{m}, \mathsf{d}, \mathsf{n}_e, \mathcal{I}, \mathcal{P}, \mathcal{V})$, where $\mathsf{rnd}, \mathsf{n}, \mathsf{m}, \mathsf{d}, \mathsf{n}_e : \{0,1\}^* \to \mathbb{N}$ are polynomial-time computable functions, and $\mathcal{I}, \mathcal{P}, \mathcal{V}$ are three algorithms that work as follows:*

- ***Offline phase**: The encoder or indexer $\mathcal{I}(\mathsf{R})$ is executed on a relation description R, and it returns $\mathsf{n}(0)$ polynomials $\{p_{0,j}\}_{j=1}^{\mathsf{n}(0)} \in \mathbb{F}[X]$ encoding the relation R and where $\mathbb{F}$ is the field specified by R.*
- ***Online phase**: The prover $\mathcal{P}(\mathsf{R}, \mathsf{x}, \mathsf{w})$ and the verifier $\mathcal{V}^{\mathcal{I}(\mathsf{R})}(\mathsf{x})$ are executed for $\mathsf{rnd}(|\mathsf{R}|)$ rounds, the prover has a tuple $(\mathsf{R}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}$, and the verifier has an instance x and oracle access to the polynomials encoding R. In the i-th round, $\mathcal{V}$ sends a message $\rho_i \in \mathbb{F}$ to the prover, and $\mathcal{P}$ replies with $\mathsf{m}(i)$ messages $\{\pi_{i,j} \in \mathbb{F}\}_{j=1}^{\mathsf{m}(i)}$, and $\mathsf{n}(i)$ oracle polynomials $\{p_{i,j} \in \mathbb{F}[X]\}_{j=1}^{\mathsf{n}(i)}$, such that $\deg(p_{i,j}) < \mathsf{d}(|\mathsf{R}|, i, j)$.*
- ***Decision phase**: After the $\mathsf{rnd}(|\mathsf{R}|)$-th round, the verifier outputs two sets of algebraic checks of the following type:*
  - *Degree checks: to check a bound on the degree of the polynomials sent by the prover. More in detail, let $\mathsf{n}_p = \sum_{k=1}^{\mathsf{rnd}(|\mathsf{R}|)} \mathsf{n}(k)$ and let $(p_1, \ldots, p_{\mathsf{n}_p})$ be the polynomials sent by $\mathcal{P}$. The verifier specifies a vector of integers $\mathbf{d} \in \mathbb{N}^{\mathsf{n}_p}$, which satisfies the following condition*

$$\forall k \in [\mathsf{n}_p] : \deg(p_k) \leq d_k.$$

  - *Polynomial checks: to verify that certain polynomial identities hold between the oracle polynomials and the messages sent by the prover. Let $\mathsf{n}^* = \sum_{k=0}^{\mathsf{rnd}(|\mathsf{R}|)} \mathsf{n}(k)$ and $\mathsf{m}^* = \sum_{k=0}^{\mathsf{rnd}(|\mathsf{R}|)} \mathsf{m}(k)$, and denote by $(p_1, \ldots, p_{\mathsf{n}^*})$*

and $(\pi_1, \ldots, \pi_{\mathsf{n}^*})$ *all the oracle polynomials (including the* $\mathsf{n}(0)$ *ones from the encoder) and all the messages sent by the prover. The verifier can specify a list of* n_e *tuples, each of the form* $(G, v_1, \ldots, v_{\mathsf{n}^*})$*, where* $G \in \mathbb{F}[X, X_1, \ldots, X_{\mathsf{n}^*}, Y_1, \ldots, Y_{\mathsf{m}^*}]$ *and every* $v_k \in \mathbb{F}[X]$*. Then a tuple* $(G, v_1, \ldots, v_{\mathsf{n}^*})$ *is satisfied if and only if* $F(X) \equiv 0$ *where*

$$F(X) := G\big(X, \{p_k(v_k(X))\}_{k=1,\ldots,\mathsf{n}^*}, \{\pi_k\}_{k=1,\ldots,\mathsf{m}^*}\big).$$

The verifier accepts if and only if all the checks are satisfied.

Completeness: *A* PHP *is complete if for any triple* $(\mathsf{R}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}$*, the checks returned by* $\mathcal{V}^{\mathcal{I}(\mathsf{R})}$ *after interacting with the honest prover* $\mathcal{P}(\mathsf{R}, \mathsf{x}, \mathsf{w})$*, are satisfied with probability 1.*

ϵ-Soundness: *A* PHP *is ϵ-sound if for every relation-instance tuple* $(\mathsf{R}, \mathsf{x}) \notin \mathcal{L}(\mathcal{R})$ *and polynomial time prover* $\mathcal{P}^*$ *we have*

$$\Pr\left[\langle \mathcal{P}^*, \mathcal{V}^{\mathcal{I}(\mathsf{R})}(\mathsf{x})\rangle = 1\right] \leq \epsilon.$$

ϵ-Knowledge Soundness: *A* PHP *is ϵ-knowledge sound if there exists a polynomial time knowledge extractor* $\mathcal{E}$ *such that for any prover* $\mathcal{P}^*$*, relation* R*, instance* x *and auxiliary input* z *we have*

$$\Pr\left[(\mathsf{R}, \mathsf{x}, \mathsf{w}) \in \mathcal{R} : \mathsf{w} \leftarrow \mathcal{E}^{\mathcal{P}^*}(\mathsf{R}, \mathsf{x}, z)\right] \geq \Pr\left[\langle \mathcal{P}^*(\mathsf{R}, \mathsf{x}, z), \mathcal{V}^{\mathcal{I}(\mathsf{R})}(\mathsf{x})\rangle = 1\right] - \epsilon,$$

where $\mathcal{E}$ *has oracle access to* $\mathcal{P}^*$*, it can query the next message function of* $\mathcal{P}^*$ *(and also rewind it) and obtain all the messages and polynomials returned by it.*

ϵ-Zero-Knowledge: *A* PHP *is ϵ-zero-knowledge if there exists a PPT simulator* $\mathcal{S}$ *such that for every triple* $(\mathsf{R}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}$*, and every algorithm* $\mathcal{V}^*$*, the following random variables are within ϵ-statistical distance:*

$$\mathsf{View}\left(\mathcal{P}(\mathsf{R}, \mathsf{x}, \mathsf{w}), \mathcal{V}^*\right) \approx_c \mathsf{View}\left(\mathcal{S}^{\mathcal{V}^*}(\mathsf{R}, \mathsf{x})\right),$$

where $\mathsf{View}\left(\mathcal{P}(\mathsf{R}, \mathsf{x}, \mathsf{w}), \mathcal{V}^*\right)$ *consists of* $\mathcal{V}^*$*'s randomness,* $\mathcal{P}$*'s messages (which do not include the oracles) and* $\mathcal{V}^*$*'s list of checks, while* $\mathsf{View}\big(\mathcal{S}^{\mathcal{V}^*}(\mathsf{R}, \mathsf{x})\big)$ *consists of* $\mathcal{V}^*$*'s randomness followed by* $\mathcal{S}$*'s output, obtained after having straightline access to* $\mathcal{V}^*$*, and* $\mathcal{V}^*$*'s list of checks.*

We assume that in every PHP scheme there is an implicit maximum degree for all the polynomials used in the scheme. Thus, we include only degree checks that differ from this maximum. In all our PHPs, the verifier is public coin.

The following definition captures de fact that zero-knowledge should hold even when the verifier has access to a bounded amount of evaluations of the polynomials that contain

information about the witness. Let $\mathcal{Q}$ be a list of queries; we say that $\mathcal{Q}$ is (b, C)-bounded for $\mathsf{b} \in \mathbb{N}^{\mathsf{n}_p}$ and C a PT algorithm, if for every $i \in [\mathsf{n}_p]$, $|\{(i,z) : (i,z) \in \mathcal{Q}\}| \leq \mathsf{b}_i$, and for all $(i,z) \in \mathcal{Q}$, $\mathsf{C}(i,z) = 1$.

(b, C)-Zero-Knowledge: *A PHP is (b, C)-zero-knowledge if for every triple $(\mathsf{R}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}$, and every (b, C)-bounded list $\mathcal{Q}$, the follow random variables are within ϵ statistical distance:*

$$\left(\mathsf{View}\big(\mathcal{P}(\mathbb{F}, \mathsf{R}, \mathsf{x}, \mathsf{w}), \mathcal{V}\big), (p_i(z))_{(i,z)\in\mathcal{Q}}\right) \approx_\epsilon \mathcal{S}\left(\mathbb{F}, \mathsf{R}, \mathsf{x}, \mathcal{V}(\mathbb{F}, \mathsf{x}), \mathcal{Q}\right),$$

where the $p_i(X)$ are the polynomials returned by the prover.

Honest-Verifier Zero-Knowledge: *A PHP is honest-verifier zero-knowledge with query bound b if there exists a PT algorithm C such that PHP is (b, C)-zero-knowledge and for all $i \in \mathbb{N}$, $Pr[\mathsf{C}(i,z) = 0]$ is negligible, where z is uniformly sampled over $\mathbb{F}$.*

2.4 Cryptographic Assumptions

Random Oracle Model (ROM). When working in the Random Oracle Model [BR93], we assume the existence of an oracle that generates uniformly sampled outputs, and we use it as a black-box. In practice, this oracle is instantiated with a cryptographic hash function and, even though it fails out of the standard model, it is a well-known and trusted assumption.

In this thesis, we design some proving systems that require interaction between the prover and the verifier. Still, we always set a *public coin* verifier, meaning that the challenges it computes are uniformly sampled elements and independent to each other, lying then in the ROM. For simplicity, even when building non-interactive arguments, we will abuse notation and describe them as interactive ones, as we claim they can also be made non-interactive by applying the Fiat-Shamir transform [FS87] we define next.

Let $\mathcal{P}$ denote the prover and $\mathcal{V}$ denote the verifier in an interactive two rounds proving system, that is

$\mathsf{srs} \leftarrow \mathsf{KeyGen}(1^\lambda)$

$\pi_1 \leftarrow \mathcal{P}(\mathsf{srs}, \mathsf{x}, \mathsf{w})$

$\alpha \leftarrow \mathcal{V}(\mathsf{srs}, \mathsf{x}, \pi_1)$

$\pi_2 \leftarrow \mathcal{P}(\pi_1, \alpha)$

$1/0 \leftarrow \mathcal{V}(\mathsf{srs}, \pi_1, \pi_2),$

and let $\mathsf{H} : \{0,1\}^* \to \{0,1\}^c$ be a hash function modeled as a random oracle. If we define $\bar{\mathcal{P}}$ and $\bar{\mathcal{V}}$ as prover and verifier in the following procedure:

- $\mathsf{srs} \leftarrow \mathsf{KeyGen}(1^\lambda)$
- $\pi \leftarrow \bar{\mathcal{P}}(\mathsf{srs}, \mathsf{x}, \mathsf{w})$:
  - $\pi_1 \leftarrow \mathcal{P}(\mathsf{srs}, \mathsf{x}, \mathsf{w})$
  - $\alpha = \mathsf{H}\pi_1$
  - $\pi_2 \leftarrow \mathcal{P}(\pi_1, \alpha)$
  - $\pi = (\pi_1, \pi_2)$
- $b \leftarrow \bar{\mathcal{V}}(\mathsf{srs}, \mathsf{x}, \pi)$
  - $\alpha = \mathsf{H}(\pi_1)$
  - $b \leftarrow \mathcal{V}(\mathsf{srs}, \mathsf{x}, \pi, \alpha)$.

and if $(\mathcal{P}, \mathcal{V})$ is a complete, knowledge-sound and zero-knowledge interactive argument of knowledge, then $(\bar{\mathcal{P}}, \bar{\mathcal{V}})$ is a complete, knowledge-sound and zero-knowledge non-interactive argument of knowledge, in the random oracle model.

Pairing-based setting A bilinear group gk is a tuple $gk = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathcal{P}_1, \mathcal{P}_2)$ where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are groups of prime order q, the elements $\mathcal{P}_1, \mathcal{P}_2$ are generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is no efficiently computable, non-degenerate bilinear map, and there is an efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. Elements in $\mathbb{G}_\gamma$, are denoted implicitly as $[a]_\gamma = a\mathcal{P}_\gamma$, where $\gamma \in \{1, 2, T\}$ and $\mathcal{P}_T = e(\mathcal{P}_1, \mathcal{P}_2)$. With this notation, $e([a]_1, [b]_2) = [ab]_T$.

Algebraic Group Model (AGM). The security of all protocols in this thesis holds in the Algebraic Group Model (AGM) of Fuchsbauer et al. [FKL18], using the dlog, qDHE, qFrac and qSDH assumptions [GG17, BB04] for asymmetric groups. The algebraic group model [FKL18] lies between the standard model and the stronger generic group model(GGM, [Sho97]). In AGM, we consider only so-called algebraic adversaries. Such adversaries have direct access to group elements and, in particular, can use their bit representation, like in the standard model. However, these adversaries are assumed to output new group elements only by applying the group operation to received group elements (like in the generic group model). This requirement is formalized as follows:

Definition 9 (Algebraic Adversary)**.** *Let $\mathbb{G}$ be a cyclic group of order p. We say that a PPT adversary $\mathcal{A}$ is algebraic if there exists an efficient extractor $\mathcal{E}_\mathcal{A}$ that, given*

the inputs $([\tau_1], \ldots, [\tau_m])$ *of* $\mathcal{A}$*, outputs a representation* $\mathbf{z} = (z_1, \ldots, z_m)^\top \in \mathbb{Z}_p^m$ *for every group element* $[y]$ *in the output of* $\mathcal{A}$ *such that:*

$$Adv^{alg}_{\mathbb{G},\mathcal{A}}(\lambda) = \left[\begin{array}{c} [y] \leftarrow \mathcal{A}([\tau_1], \ldots, [\tau_m]), \mathbf{z} \leftarrow \mathcal{E}_{\mathcal{A}}([y], [\tau_1], \ldots, [\tau_m]), \\ \text{and } [\tau] \neq \prod_{j=1}^{m} z_j [x_j] \end{array} \right] \leq \mathsf{negl}(\lambda)\,.$$

We will always assume our adversaries are algebraic and use the extractor for elements in both groups $\mathbb{G}_1$ and $\mathbb{G}_2$. The security of our schemes is proven in the algebraic group model under the following assumptions:

Definition 10 (qDlog Asymmetric Assumption)**.** *The* $q(\lambda)$*-discrete logarithm assumption holds for* $gk \leftarrow \mathcal{G}(1^\lambda)$ *if for all PPT algorithm* $\mathcal{A}$

$$Adv^{q-dlog}_{gk,\mathcal{A}}(\lambda) = \mathsf{Pr}\left[\tau \leftarrow \mathcal{A}(gk, [\tau]_{1,2}, \ldots, [\tau^q]_{1,2})\right] = \mathsf{negl}(\lambda).$$

Definition 11 (qBSDH Assumption)**.** *The* q*-bilinear strong Diffie-Hellman assumption holds for* $gk \leftarrow \mathcal{G}(1^\lambda)$ *if for all PPT adversaries* $\mathcal{A}$*,*

$$\Pr\left[(c, \tfrac{1}{(\tau - c)} e([1]_1, [1]_2)) \leftarrow \mathcal{A}(\mathsf{gk}, [1]_{1,2}, [\tau]_{1,2}, \ldots, [\tau^q]_{1,2}) \,\middle|\, \begin{array}{c} gk \leftarrow \mathcal{G}(1^\lambda) \\ \tau \leftarrow \mathbb{F}_p \end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

Definition 12 (qDHE Assumption)**.** *The* q*-Diffie-Hellman exponent assumption holds relative to* $\mathcal{G}(1^\lambda)$ *if for all PPT adversaries* $\mathcal{A}$*,*

$$\Pr\left[\tau^{q+1} e([1]_1, [1]_2) \leftarrow \mathcal{A}(\mathsf{gk}, [1]_{1,2}, [\tau]_{1,2}, \ldots, [\tau^q]_{1,2}) \,\middle|\, \begin{array}{c} gk \leftarrow \mathcal{G}(1^\lambda) \\ \tau \leftarrow \mathbb{F}_p \end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

Definition 13 (qSFrac Assumption [GG17])**.** *The* qS *fractional assumption holds relative to* $\mathcal{G}(1^\lambda)$ *if for all PPT adversaries* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{c} \frac{r(\tau)}{s(\tau)} e([1]_1, [1]_2) \leftarrow \mathcal{A}(\mathsf{gk}, [1]_{1,2}, [\tau]_{1,2}, \ldots, [\tau^q]_{1,2}) \\ \wedge \deg(r) < \deg(s) \leq q. \end{array} \,\middle|\, \begin{array}{c} gk \leftarrow \mathcal{G}(1^\lambda) \\ \tau \leftarrow \mathbb{F}_p \end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

2.5 Other Preliminaries

2.5.1 The KZG Polynomial Commitment Scheme

All our constructions use the KZG polynomial commitment scheme that we describe below, or its adaptation for vector commitment that we explain in the next section. For efficiency of our construction in Chapter 5, we slightly modify the polynomial commitment in order to add degree checks to the original protocol, without incurring in extra proof elements or pairings. The polynomial commitment introduced by Kate, Zaverucha and Goldberg in [KZG10] is a tuple of algorithms $(\mathsf{KeyGen}, \mathsf{PC.Commit}, \mathsf{PC.Open}, \mathsf{PC.Verify})$ such that:

- $\mathsf{PC.KeyGen}\big(\mathsf{pp}_{\mathsf{PC}}, d\big)$: On input the system parameters and a degree bound d, it outputs a structured reference string $\mathsf{srs} = \big(\{[\tau^i]_{1,2}\}_{i=1}^{d}\big)$.

- $\mathsf{PC.Commit}\big(\mathsf{srs}, p(X)\big)$: It outputs $\mathsf{C} = [p(\tau)]_1$.

- $\mathsf{PC.Open}\big(\mathsf{srs}, p(X), \alpha\big)$: Prover computes

$$q(X) = \frac{p(X) - p(\alpha)}{X - \alpha} \ ,$$

  sets $s = p(\alpha), [Q]_1 = [q(\tau)\tau^{d-\deg+2}]_1$, and outputs $(s, \pi_{\mathsf{PC}} = [Q]_1)$.

- $\mathsf{PC.Verify}\big(\mathsf{srs}, \mathsf{C}, \deg, \alpha, s, \pi_{\mathsf{PC}}\big)$: Verifier accepts if and only if

$$e(\mathsf{C} - s, [\tau^{d-\deg+2}]_2) = e([Q]_1, [\tau - \alpha]_2).$$

Security. It has been proven in [KZG10, CHM+20, GWC19] that the original KZG protocol, i.e., where $[Q]_1 = [q(\tau)]_1$ and the pairing equation is $e(\mathsf{C} - s, [1]_2) = e([Q]_1, [\tau - \alpha]_2)$, is a polynomial commitment scheme that satisfies completeness, evaluation blinding and extractability as in Def. 3 in the AGM, under the dlog assumption. What is more, Marlin presents an alternative version of KZG with degree checks that does not require additional powers in $\mathbb{G}_2$. For our construction, we claim that adding $\tau^{d-\deg+2}$ to the pairing and element $[Q]_1$ does not affect completeness or extractability. We also argue that under the AGM, no PPT adversary $\mathcal{A}$ can break soundness by providing a commitment to a polynomial $p(X)$ such that $\deg(p) > \deg$. Indeed, if that is the case, $\deg(Q) = d + 1$ for $Q(X)$ the algebraic representation of $[Q]_1$, which will imply an attack to the qDHE assumption, as the srs only contains powers $[\tau^i]_1$ up to d.

2.5.2 KZG as Vector Commitment Scheme

There is a natural isomorphism between vectors of size m and polynomials of degree $m - 1$; where we can represent $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{F}^m$ as $V(X) = \sum_{j=1}^{m} v_j B_j(X)$, for a basis $\mathcal{B} = \{B_j(X)\}_{j=1}^{m}$ of the space of polynomials of degree up to $m - 1$, and vice versa. This fact implies as well a natural relation between polynomial and vector commitments (Def. 4), where in particular, the former implies the latter. What is more, when the basis $\mathcal{B}$ chosen to encode the vector consists of Lagrange polynomials we have vector commitments with *easy* individual position openings: evaluating $V(X)$ in the ith interpolation point returns v_i.

In this work we will use the protocol by Kate et al. for both cases, polynomial and vector commitments. For the latter, we will not only consider individual openings but also subset openings. In particular, let $\mathbb{H} = \{\mathsf{h}_1, \ldots, \mathsf{h}_m\}$ be a set of roots of

unity and $\{\lambda_i(X)\}_{i=1}^m$ its corresponding Lagrange interpolation set, with vanishing polynomial $z_H(X)$. We have that for some polynomial $Q(X)$,

$$V(X) - s = (X - \mathsf{h}_i)Q(X) \text{ if and only if } V(\mathsf{h}_i) = v_i = s.$$

2.5.3 Subset openings

For a vector $\mathbf{v} \in \mathbb{F}^m$ and a subset $I \subset [m]$, the subvector opening scheme of Tomescu et. al [TAB+20] that works for the VC inspired by KZG presented above, consists on algorithms $(\mathsf{SVC.Open}, \mathsf{SVC.Verify})$ such that:

- $\mathsf{SVC.Open}(\mathsf{prk}, \mathsf{aux}, I, \mathbf{v}_I)$: Compute $C_I(X) = \sum_{i \in I} v_i \eta_i(X)$, where $\{\eta_i(X)\}$ are the Lagrange interpolation polynomials of the set $\{\mathsf{h}_i\}_{i \in I}$, and find $Q(X)$ such that for $z_I(X) = \prod_{i \in I}(X - \mathsf{h}_i)$,

  $$C(X) - C_I(X) = z_I(X)Q(X).$$

  Output $\pi_I = [Q]_1$.

- $\mathsf{SVC.Verify}(\mathsf{vrk}, \mathsf{C}, I, \mathbf{v}_I, \pi_I)$: Compute $[z_I]_2 = [z_I(\tau)]_2$, $C_I(X) = \sum_{i \in I} v_i \eta_i(X)$, and $\mathsf{C}_I = [C_I(\tau)]_1$ and output 1 if and only if

  $$e\big(\mathsf{C} - \mathsf{C}_I, [1]_2\big) = e\big([Q]_1, [z_I]_2\big).$$

$\mathsf{SVC.Open}$ as aggregation of individual proofs We will additionally use a result by Tomescu et al. [TAB+20] that allows the prover to compute $[Q]_1$ in time $\mathcal{O}(m \log^2(m))$ given it already has stored proofs $\{[Q_i]_1\}_{i \in I}$ that $C(\mathsf{h}_i) = s_i$. Indeed the prover sets

$$[Q]_1 = \sum_{i \in I} \left(\prod_{k=1, k \neq i}^{m} \frac{1}{(\mathsf{h}_i - \mathsf{h}_k)} \right) [Q_i]_1$$

Remark. *We remark that precomputing all the proofs $[Q_1]_1, \ldots, [Q_n]_1$ that $C(\mathsf{h}_i) = s_i$ can be achieved in time $\mathcal{O}(n \log n)$ using techniques by Feist and Khovratovich [FK]. The overview of this technique by Tomescu et al. ([TAB+20], Section 3.4.4,* "Computing All u_i's Fast"*) is explained well.*

Multiple Openings

A KZG proof of opening can naturally be extended to open one polynomial in many points. Indeed, let $p(X)$ be a polynomial, $\alpha \in \mathbb{F}^m$ a vector of opening points and $\mathbf{s}$ such that $s_i = p(\alpha_i)$ for all $i = 1, \ldots, m$. Define $C_\alpha(X)$ as the unique polynomial of

degree $m-1$ such that $C_\alpha(\alpha_i) = s_i$ for all $i \in [m]$. We have that $p(\alpha_i) = s_i$ for all $i = 1, \ldots, m$ if and only if there exists $q(X)$ such that

$$p(X) - C_\alpha(X) = \prod_{i=1}^{m} (X - \alpha_i) Q(X)$$

We can thus redefine the KZG prover and verifier the following way:

- $\mathsf{PC.Open}\big(\mathsf{srs}, p(X), \deg, \alpha\big)$: Prover computes $\{\eta_i(X)\}_{i=1}^m$ the interpolation Lagrange polynomials for the set $\{\alpha_i\}_{i=1}^m$, $z_\alpha(X) = \prod_{i=1}^m (X - \alpha_i)$ and define $C_\alpha(X) = \sum_{i=1}^m p(\alpha_i)\eta_i(X)$. Then, it computes

  $$Q(X) = \frac{p(X) - C_\alpha(X)}{z_\alpha(X)} \ ,$$

  sets $s_i = p(\alpha_i), [Q]_1 = [Q(\tau)]_1$, and outputs $(\mathbf{s}, \pi_{\mathsf{PC}} = [Q]_1)$.

- $\mathsf{PC.Verify}\big(\mathsf{srs}, \mathsf{C}, \deg, \alpha, \mathbf{s}, \pi_{\mathsf{PC}}\big)$: The verifier computes $\{\eta_i(X)\}_{i=1}^m$, $\mathsf{C}_\alpha = [C_\alpha(\tau)]_1$, $[z_\alpha(\tau)]_2$ and verifies

  $$p(X) - C_\alpha(X) = Q(X) z_\alpha(X)$$

  by making the pairing check

  $$e(\mathsf{C} - \mathsf{C}_\alpha, [1]_2) = e([Q]_1, [z_\alpha(\tau)]_2),$$

  and outputs 1 if and only if the equation is satisfied and $\deg(p) \leq d$.

KZG for Bivariate Polynomials

For the protocol in Section 5.4.1 we will use bivariate polynomials, or polynomials of higher degree. What this mean is that, if we have a bivariate polynomial $P(X, Y)$ with degree up to $d_1 - 1$ in X and $d_2 - 1$ in Y then we require a universal setup with $d_1 d_2$ powers. We work with a version of KZG that uses a univariate setup because these are already available for multiple different curves (i.e. we do *not* need a specialist setup just for our protocol and can work with prior KZG setups).

We observe that, by using the KZG open algorithm, we can commit to $P(X, Y)$ as $[P(\tau^{d_2}, \tau)]_1$. We must open $P(X, Y)$ in two steps. First we *partially* open $P(X, Y)$ at some point $X = \alpha$ to a commitment $[P(\alpha, \tau)]_1$. The partial proof is given by a commitment $[w_\alpha(\tau^{d_2}, \tau)]$ to a partial witness

$$w_\alpha(X, Y) = \frac{P(X, Y) - P(\alpha, Y)}{X - \alpha}$$

We then fully evaluate $P(\alpha, X)$ at $Y = \beta$ via a standard KZG proof with a degree bound of $d_2 - 1$ on $[P(\alpha, \tau)]_1$.

2.5.4 Pedersen Commitment Schemes

Pedersen commitment schemes are a particular case of vector commitments. We will consider them for committing to single values in a zero knowledge way. Thus, the srs will additionally output $[\mathsf{g}]_1$ for some secret g and the commitment to some element s is computed as $v[1]_1 + r[\mathsf{g}]_1 = [v + \mathsf{g}r]$, for some randomly sampled $\mathsf{h} \in \mathbb{F}$. We suggest a standard Fiat-Shamired Sigma protocol [Mau09] to demonstrate knowledge of v, r such that $\mathsf{cm} = [v + \mathsf{g}r]_1$ for some v, r:

$$R_{\mathsf{ped}} = \{(\mathsf{cm};\ (v, r)) : \quad \mathsf{cm} = [v + \mathsf{g}r]_1\}$$

The proof consists of $R = [s_1 + \mathsf{h}s_2]_1$, $t_1 = s_1 + vc$ and $t_2 = s_2 + rc$, where $c = \mathsf{H}(\mathsf{cm}, R)$ and s_1, s_2 are elements chosen by the verifier. At the end, the verifier checks that $R + c \cdot \mathsf{cm} = [t_1 + \mathsf{g}t_2]_1$.

Chapter 3

Universal and Updatable SNARKs

This chapter is based on the paper 'An Algebraic Framework for Universal and Updatable SNARKs' [RZ21], which is a joint work with Carla Ràfols.

3.1 Introduction

(zk)SNARKs as presented in Definition 1 are proving systems for general computations, which we represent as arithmetic circuits. Arithmetic Circuit Satisfiability can be reduced to a set of quadratic and affine constraints over a finite field. The quadratic ones are universal, in the sense that they are the same for all circuits of same size, and can be easily proven in the pairing-based setting with a Hadamard product argument, the basic core of most zkSNARKS constructions starting from [GGPR13]. On the other hand, affine constraints are circuit-dependent, and it is a challenging task to efficiently prove them with a universal SRS [MBKM19, CHM+20, GWC19, CFF+21, DRZ20, Set20, Gab19, SZ20].

In Groth et al. [GKM+18] they are proven via a very expensive subspace argument that requires a SRS quadratic in the circuit size and a preprocessing step that is cubic. Sonic [MBKM19], the first efficient, universal, and updatable SNARK, gives two different ways to prove the affine constraints, a fully succinct one (not so efficient) and another one in the amortized setting (very efficient). Follow-up work (most notably, Marlin [CHM+20], Plonk [GWC19], Lunar [CFF+21]) has significantly improved the efficiency in the fully succinct mode.

There is an important trend in cryptography, that advocates for constructing protocols in a modular way. One reason for doing so is the fact that, by breaking complicated protocols into simpler steps, they become easier to analyze. Ishai [Ish20] mentions comparability as another fundamental motive. Specially in the area of

zero-knowledge, given the surge of interest in practical constructions, it is hard not to lose sight of what each proposal achieves. As Ishai puts it: "*one reason such comparisons are difficult is the multitude of application scenarios, implementation details, efficiency desiderata, cryptographic assumptions, and trust models*".

Starting from Sonic, all the aforementioned works on universal and updatable zk-SNARKs follow this trend. More concretely, they first build an information-theoretic proof system, that is then compiled into a full argument under some computational assumptions in bilinear groups. The main ingredient of the compiler is a polynomial commitment [KZG10, BFS20, KPV19]. However, the information theoretic component is still very complex and comparison among these works remains difficult, for precisely the same reasons stated by Ishai. In particular, it is hard to extract the new ideas in each of them in the complex description of the arguments, that use sophisticated tricks for improving efficiency, as well as advanced properties of multiplicative subgroups of a finite field or bivariate Lagrange interpolation. Further, it is striking that all fully succinct arguments are for restricted types of constraints (sums of permutations in Sonic, sparse matrices in Marlin, and Lunar[1]) or pay a price for additive gates (Plonk). A modular, unified view of these important works seems essential for a clearer understanding of the techniques. In turn, this should allow for a better comparison, more flexibility in combining the different methods, and give insights on current limitations.

3.1.1 Related Work

Bivariate Polynomial Evaluation Arguments. As mentioned before, the complexity of building updatable and universal zkSNARKs protocols is mainly caused by proving affine constraints. A natural way to encode them is through a bivariate public polynomial $P(X, Y)$; in order to avoid having a quadratic SRS, this polynomial can only be given to the verifier evaluated or partially evaluated in the field. The common approach is to let the verifier chose arbitrary field elements x, y and having the prover evaluate and send $\sigma = P(y, x)$. The challenge is to prove that the evaluation has been performed correctly. In Sonic [MBKM19], this last step is called a *signature of correct computation* [PST13] and can be performed by the prover or by the verifier with some help from an untrusted third party. The drawback of the first construction is that, while still linear, prover's work is considerably costly; also, linear constraints are assumed to be sparse and the protocol works exclusively for a very particular polynomial $P(X, Y)$. The second construction is interesting only in some restricted settings where the same verifier checks a linear amount of proofs for one circuit. Marlin [CHM$^+$20] bases its construction on the univariate sum-check protocol of Aurora [BCR$^+$19] and presents a novel way to navigate from the naive quadratic representation $P(X, Y)$ to a linear one. This approach results in succinct prover and verifier work, but restricts their protocol to the case where the num-

[1]The number of non-zero entries of the matrices that encode linear constraints cannot exceed the size of some multiplicative group of the field of definition.

ber of non-zero entries of matrix $\mathbf{W}$ is bounded by the size of some multiplicative subgroup of the field of definition. Lunar [CFF+21] uses the same representation as Marlin but improves on it, among other tweaks by introducing a new language (R1CS-lite) that can also encode arithmetic circuit satisfiability, but has a lighter representation than other constraint systems. Plonk [GWC19] does not use bivariate polynomials or require sparse matrices but the SRS size depends on the number of both multiplicative and additive gates. As we do, Plonk, Marlin and Lunar use the Lagrange interpolation basis to commit to vectors, while Claymore [SZ20] presents a modular construction for zkSNARKs based on similar algebraic building blocks but in the monomial basis: inner product, Hadamard product and matrix-vector product arguments.

Work		$\lvert\mathsf{srs_u}\rvert$	$\lvert\mathsf{srs_W}\rvert$	$\lvert\pi\rvert$	KeyGen	Derive	Prove	Verifier
Sonic [MBKM19]	$\mathbb{G}_1$	$4M$	-	20	$4M$	$36m$	$273m$	7P
	$\mathbb{G}_2$	$4M$	3	-	$4M$	-	-	
	$\mathbb{F}$	-	-	16	-	$O(K \log K)$	$O(K \log K)$	$O(l + \log K)$
Marlin [CHM+20]	$\mathbb{G}_1$	$3\hat{K}$	12	13	$3\hat{K}$	$12K$	$14m + 8K$	2P
	$\mathbb{G}_2$	2	2	-	-	-	-	
	$\mathbb{F}$	-	-	8	-	$O(K \log K)$	$O(K \log K)$	$O(l + \log K)$
Plonk [GWC19]	$\mathbb{G}_1$	$3N$	8	7	$3N$	$8n$	$11n$	2P
	$\mathbb{G}_2$	1	1	-	-	-	-	
	$\mathbb{F}$	-	-	7	-	$O(n \log n)$	$O(n \log n)$	$O(l + \log n)$
LunarLite2x[2] [CFF+21]	$\mathbb{G}_1$	$\hat{K}$	16	11	$\hat{K}$	$16K$	$8m + 4K$	2P
	$\mathbb{G}_2$	1	1	-	1	-	-	-
	$\mathbb{F}$	-	-	3	-	$O(K \log K)$	$O(K \log K)$	$O(l + \log K)$
Vampire [LSZ22]	$\mathbb{G}_1$	$12M + \hat{K}$	–	4	$\hat{K}$	$16K$	$20M + 2\hat{K}$	6P
	$\mathbb{G}_2$	$4M + \hat{K}$	22	-	1	$120M + 18\hat{K}$	-	$5\mathbb{G}_1 + 21\mathbb{G}_2$
	$\mathbb{F}$	-	-	1	-	$O(\hat{K} \log \hat{K})$	$O(K \log K)$	$O(l + \log M)$
This work Sec. 3.5.3	$\mathbb{G}_1$	$\hat{K}$	4	10	$\hat{K}$	$6K$	$6m + 4K$	2P
	$\mathbb{G}_2$	1	1	-	-	-	-	
	$\mathbb{F}$	-	-	3	-	$O(K \log K)$	$O(K \log K)$	$O(l + \log K)$
This work Fig. 3.10	$\mathbb{G}_1$	N	11	8	N	$11n$	$8n$	2P
	$\mathbb{G}_2$	1	1	-	-	-	-	
	$\mathbb{F}$	-	-	4	-	$O(n \log n)$	$O(n \log n)$	$O(l + \log n)$
Basilisk Sec. 3.7.3	$\mathbb{G}_1$	M	$3V + 1$	6	M	$(3v + 1)m$	$6m$	2P
	$\mathbb{G}_2$	1	1	-	-	-	-	
	$\mathbb{F}$	-	-	2	-	$O(m \log m)$	$O(m \log m)$	$O(l + \log m)$

Table 3.1: Comparison with state of the art universal and updatable zkSNARKs. m: number of multiplicative gates, n: number of total gates, v : bounded fan-out, K: non-zero elements of the matrix that describe the circuit ($|\mathbf{F} + \mathbf{G}|$ in our case). $N, \hat{K}, V, M$: maximum supported values for n, K, m, v. $N^* = M + A$. The numbers for our work take into account the trick to eliminate non-trivial degree checks in Section 3.7.2.

Information Theoretic Proof Systems. All these previous works follow the two step process described in the introduction and build their succinct argument by compiling an information theoretically secure one. Marlin introduces Algebraic Holographic proofs, that are variation of interactive oracle proofs (IOPs) [BCS16]. Holographic refers to the fact that the verifier never receives the input explicitly

[3]Among all schemes with different trade-offs presented in Lunar, we highlight this one as it is the most directly comparable to our scheme.

(otherwise, succinctness would be impossible), but rather its encoding as an oracle computed by an indexer or encoder. The term algebraic refers to the fact that oracles are low degree polynomials, and malicious provers are also bound to output low degree polynomials. This notion is similar to the one of Idealised Low Degree protocols of Plonk. Lunar refines this model by introducing Polynomial Holographic IOPs (Definition 2.3.7), which generalize these works mostly by allowing for a fine grained analysis of the zero-knowledge property, including degree checks, and letting prover and verifier send field elements.

Polynomial Commitments. Polynomial commitments allow to commit to a polynomial $p(X) \in \mathbb{F}[X]$, and open it at any point $x \in \mathbb{F}$. As it is common, we will use an adaptation of the polynomial commitment by Kate et al. [KZG10], which is presented in Section 2.5.1. Sonic gave a proof of extractability of the latter in the Algebraic Group Model [FKL18], and Marlin completed the proof to make the commitments usable as a standalone primitive, and also have an alternative construction under knowledge assumptions. Both Marlin and Plonk considered versions of polynomial commitments where queries in the same point can be batched together. For this work, we use the same definitions and construction as these works. We formally introduced polynomial commitments in Definition 3.

Untrusted Setup. The original constructions of pairing-based zkSNARKs crucially depend for soundness on a trusted setup, although, as was shown in [ABLZ17, Fuc18], the zero-knowledge property is still easy to achieve when the setup is subverted. Groth et al. introduced the updatable SRS model in [GKM+18] to address the issue of trust in SRS generation. There are several alternatives to achieve transparent setup and constant-size proofs, but all of them have either linear verifier [BCC+16, BBB+18, BCR+19, AHIV17], or work only for very structured types of computation [BBHR18, WTs+18]. An exception is the work of Setty [Set20]. Concretely, the approach is less efficient in terms of proof size and verification complexity compared to recent constructions of updatable and universal pairing-based SNARKs.

After [RZ21]. The results in this chapter have been published in 2021. Using similar techniques as the one in our work, Lipmaa Siim and Zajac introduce Vampire ([LSZ22]), a universal zkSNARK that achieves the smallest proof size. Their construction implicitly uses our framework, and the efficiency relies mostly on a sumcheck protocol with a 1 element proof. Still, their derived SRS is more than five times the one for our work, and need to be derived from a non-reusable (but still universal) SRS.

3.1.2 Contributions

We propose an algebraic framework that takes a step further in achieving modular constructions of universal and updatable SNARKs. We identify the technical core

of previous work as instances of a *Checkable Subspace Sampling* (CSS) Argument. In this information-theoretic proof system, two parties, prover and verifier, on input a field $\mathbb{F}$ and a matrix $\mathbf{M} \in \mathbb{F}^{Q \times m}$, agree on a polynomial $D(X)$ encoding a vector $\mathbf{d}$ in the row space of $\mathbf{M}$. The interesting part is that, even though the coefficients of the linear combination that define $\mathbf{d}$ are sampled with the verifier's coins, the latter does not need to perform a linear (in Q, the number of rows) number of operations to verify that $D(X)$ is correct. Instead, this must be demonstrated by the prover.

With this algebraic formulation, it is immediate to see that a CSS argument can be used as a building block for an argument of membership in linear spaces. Basically, given a matrix $\mathbf{M}$, we can prove that some vector $\mathbf{y}$ is orthogonal to the row vectors of $\mathbf{M}$ by sampling after $\mathbf{y}$ is declared, a *sufficiently random* vector $\mathbf{d}$ in the row space of $\mathbf{M}$ and checking an inner product relation, namely, whether $\mathbf{d} \cdot \mathbf{y} = 0$. The purpose of a CSS argument is to guarantee that the sampling process can be checked by the verifier in sublinear time without sacrificing soundness.

Naturally, for building succinct proofs, instead of $\mathbf{y}, \mathbf{d}$, the argument uses polynomial encodings $Y(X)$ and $D(X)$ (which are group elements after the compilation step). To compute the inner product of this encoded vectors, we introduce a new argument in Section 3.3, which is specific to the case where the polynomials are encoded in the Lagrange polynomial basis but can be easily generalized to the monomial basis (as we do later in Section 4.3.2). The argument is a straightforward application of the univariate sumcheck of Aurora [BCR$^+$19]. However, we contribute a generalized sumcheck (that works not only for multiplicative subgroups of finite fields), with a completely new proof that relates it with polynomial evaluation at some fixed point v.

These building blocks can be put together as an argument for the language of Rank1 constraint systems. For efficiency and generality, we introduce W-R1CS, a variant that captures all constraint systems used by existing universal and updatable snarks. Our final construction can be instantiated with any possible choice of CSS scheme, so in particular, it can essentially recover the construction of Plonk, Marlin and Lunar by isolating the CSS argument implicit in these works, or the amortized construction of Sonic. We hope that this serves to better identify the challenge behind building updatable and universal SNARKs, and allow for new steps in improving efficiency, as well as more easily combining the techniques.

In summary, we reduce R1CS constraint systems to three algebraic relations: an inner product, a Hadamard product, and a CSS argument. We think this algebraic formulation is very clear, and also makes it easier to relate advances in universal and updatable SNARKS with other works that have used a similar language, for example, the arguments for inner product of [BCC$^+$16], of membership in linear spaces [JR13], or for linear algebra relations [Gro09].

Finally, we give several constructions of CSS arguments. In Section 3.5, we start from the representation of a matrix $\mathbf{W}$ as bivariate polynomial introduced

in [CHM+20] to construct an alternative CSS argument for sparse matrices. Our contribution is to introduce a new linearization step that allows us to build it modularly from an argument for what we refer to as *simple matrices*, i.e., those with at most one non-zero element per column. Compared to [CHM+20, CFF+21], at a minimal increase in communication cost, our argument significantly reduces the size of the derived SRS. We generalize these arguments to sums of simple and sparse matrices, without increasing the communication complexity.

In Section 3.5.7, we show the helped Sonic mode works as a CSS argument in the amortized setting. In Section 3.5.6, we introduce a CSS argument that works for arbitrary matrices and, even though it requires quadratic indexer work and linear verifier memory, can be combined with other schemes to increase efficiency or expressivity, as we show in Section 3.4.4.

We study the performance trade-offs resulting from the different possible choices of CSS argument of Section 3.5. In particular. we observe that the argument for simple matrices and sums of simple matrices is useful on its own, and not only to achieve modularity. We study the efficiency of our zkSNARK when: a) the CSS argument is our argument for sparse matrices of Section 3.5.3, b) when the Plonk constraint system is used and the matrix of constraints is a permutation, which is a special case of a simple matrix, and c) when the circuit has bounded fan-out, which results in a matrix of constraints that is a sum of simple matrices, which are detailed in Table 3.1.

3.2 A Generalized Constraint System

Formally, we will construct an argument for the universal relation $\mathcal{R}_{\mathsf{W\text{-}R1CS}}$, that we present and discuss in this chapter. This definition comes as a generalization that captures several previous constraint systems introduced below.

Definition 14. *(R1CS) Let $\mathbb{F}$ be a finite field and $m, l, s \in \mathbb{N}$. We define the universal relation R1CS as:*

$$\mathcal{R}_{R1CS} = \left\{ \begin{array}{c} (\mathsf{R}, \mathsf{x}, \mathsf{w}) := \big((\mathbb{F}, s, m, l, \mathbf{F}, \mathbf{G}, \mathbf{O}), \mathbf{x}, \mathbf{w}\big) : \\ \mathbf{F}, \mathbf{G}, \mathbf{O} \in \mathbb{F}^{m \times m}, \mathbf{x} \in \mathbb{F}^{l-1}, \mathbf{w} \in \mathbb{F}^{m-l}, s = \max\{|\mathbf{F}|, |\mathbf{G}|, |\mathbf{O}|\}, \\ \textit{and for } \mathbf{z} := (1, \mathbf{x}, \mathbf{w}), (\mathbf{F}\mathbf{z}) \circ (\mathbf{G}\mathbf{z}) = \mathbf{z} \end{array} \right\}.$$

Campanelli et al. introduced in [CFF+21] an optimized version of Rank 1 Constraint Systems that is still NP complete and encodes circuit satisfiability in a natural way:

Definition 15. *(R1CS-lite) Let $\mathbb{F}$ be a finite field and $n, m, l \in \mathbb{N}$. We define the universal relation R1CS-lite as:*

$$\mathcal{R}_{R1CS\text{-}\mathsf{lite}} = \left\{ \begin{array}{c} (\mathsf{R}, \mathsf{x}, \mathsf{w}) := \big((\mathbb{F}, s, m, l, \mathbf{F}, \mathbf{G}), \mathbf{x}, \mathbf{w}\big) : \\ \mathbf{F}, \mathbf{G} \in \mathbb{F}^{m \times m}, \mathbf{x} \in \mathbb{F}^{l-1}, \mathbf{w} \in \mathbb{F}^{m-l}, s = \max\{|\mathbf{F}|, |\mathbf{G}|\}, \\ \textit{and for } \mathbf{c} := (1, \mathbf{x}, \mathbf{w}), (\mathbf{F}\mathbf{c}) \circ (\mathbf{G}\mathbf{c}) = \mathbf{c} \end{array} \right\}.$$

Definition 16. *(Plonk's constraint system) Let $\mathbb{F}$ be a finite field and $m, l \in \mathbb{N}$. Plonk's universal relation can be denoted as:*

$$\mathcal{R}_{\mathsf{Plonk}} = \left\{ \begin{array}{c} (\mathsf{R}, \mathsf{x}, \mathsf{w}) := \big((\mathbb{F}, m, l, \mathbf{P}, \mathbf{q}_L, \mathbf{q}_R, \mathbf{q}_O, \mathbf{q}_M, \mathbf{q}_C), \mathbf{x}, (\mathbf{a}', \mathbf{b}, \mathbf{c})\big) : \\ \mathbf{P} \in \mathbb{F}^{3m \times 3m} \text{ a permutation matrix}, \\ \mathbf{q}_\gamma \in \mathbb{F}^m \text{ for } \gamma \in \{L, R, O, M, C\}, \mathbf{x} \in \mathbb{F}^{l-1}, \\ \mathbf{a}' \in \mathbb{F}^{m-l}, \mathbf{a} = (1, \mathbf{x}, \mathbf{a}'), \\ (1)\ \mathbf{P} \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{pmatrix} = \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{pmatrix}, \text{ and} \\ (2)\ \mathbf{q}_L \circ \mathbf{a} + \mathbf{q}_R \circ \mathbf{b} + \mathbf{q}_O \circ \mathbf{c} + \mathbf{q}_M \circ \mathbf{a} \circ \mathbf{b} + \mathbf{q}_C = \mathbf{0} \end{array} \right\}.$$

The first equation is a "copy constraint" approach that takes care of consistency among wires. The second equation represents the different types of gates. When writing Circuit Satisfiability as satisfiability of this constraint system, m represents the number of additive and multiplicative gates. This relation is a rewriting of the one considered in Plonk. Indeed, the only difference is that we require $\mathbf{a}$ to have the public input in the first positions, instead of forcing this in Eq.(2). Still, this is only a reformulation and does not modify the constraint system itself.

In this work we go a step further and present a constraint system that generalizes R1CS and R1CS-lite as introduced above, while contemplating the Plonk constraint system as well. Consider the following universal relation, that depends on a set $\mathcal{M}$ of admissible matrices:

Definition 17. *(weighted-R1CS) Let $\mathbb{F}$ be a finite field and $m, l, l_b \in \mathbb{N}$. We define the universal relation* W-R1CS *as:*

$$\mathcal{R}_{\text{W-R1CS}} = \left\{ \begin{array}{c} (\mathsf{R}, \mathsf{x}, \mathsf{w}) := \big((\mathbb{F}, m, l, l_b, \mathbf{W}, \mathbf{q}_M, \mathbf{q}_L, \mathbf{q}_R, \mathbf{q}_C), \mathbf{x}, (\mathbf{a}', \mathbf{b}')\big) : \\ \mathbf{W} \in \mathcal{M} \subset \mathbb{F}^{Q \times 3m}, \\ \mathbf{q}_\gamma \in \mathbb{F}^m \text{ for } \gamma \in \{L, R, M, C\}, \mathbf{x} \in \mathbb{F}^{l-1}, \mathbf{a}' \in \mathbb{F}^{m-l}, \mathbf{b}' \in \mathbb{F}^{m-l_b} \\ \text{and for } \mathbf{a} := (1, \mathbf{x}, \mathbf{a}'), \mathbf{b} = (\mathbf{1}_{l_b}, \mathbf{b}') \\ \mathbf{W} \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{q}_M \circ \mathbf{a} \circ \mathbf{b} + \mathbf{q}_L \circ \mathbf{a} + \mathbf{q}_R \circ \mathbf{b} + \mathbf{q}_C \end{pmatrix} = \mathbf{0} \end{array} \right\}.$$

Note that, with this arithmetization, the vector $\mathbf{c}$ of outputs is implicitly set to be a weighted combination of $\mathbf{a}, \mathbf{b}, \mathbf{a} \circ \mathbf{b}$ and the constant 1.

Below, we show how $\mathcal{R}_{\text{W-R1CS}}$ captures existing universal relations, for different types of admissible matrices.

R1CS-lite. Note that for the case where $l = l_b$, $\mathbf{q}_M = \mathbf{1}$, $\mathbf{q}_L = \mathbf{q}_R = \mathbf{q}_C = \mathbf{0}$, and $\mathbf{W} = \begin{pmatrix} \mathbf{I} & \mathbf{0} & -\mathbf{F} \\ \mathbf{0} & \mathbf{I} & -\mathbf{G} \end{pmatrix}$ for matrices $\mathbf{F}, \mathbf{G}$ in $\mathbb{F}^{m \times m}$ containing the coefficients for

the linear constraints of the circuit, the relation described above corresponds to R1CS-lite.

Plonk. $\mathcal{R}_{\mathsf{W\text{-}R1CS}}$ can encode $\mathcal{R}_{\mathsf{Plonk}}$ when the set of admissible matrices includes $\mathbf{P} - \mathbf{I}$, where $\mathbf{P}$ is a permutation and $\mathbf{I}$ the identity matrix, and when we restrict ourselves to relations R such that $\mathbf{q}_O = -\mathbf{1}$. Indeed, it suffices to define $\mathbf{W} = \mathbf{P} - \mathbf{I}$, and observe that if Eq.(2) in $\mathcal{R}_{\mathsf{Plonk}}$ is satisfied, this means that

$$\mathbf{q}_L \circ \mathbf{a} + \mathbf{q}_R \circ \mathbf{b} + \mathbf{q}_M \circ \mathbf{a} \circ \mathbf{b} + \mathbf{q}_C = \mathbf{c}.$$

The restriction that $\mathbf{q}_O = -\mathbf{1}$ limits the expressiveness of the constraint system, but still captures all the constraint types described in Plonk, as we argue next.

First, observe that given some encoding of a relation R as in $\mathcal{R}_{\mathsf{Plonk}}$ with vectors $\mathbf{q}'_L, \mathbf{q}'_R, \mathbf{q}'_O, \mathbf{q}'_M, \mathbf{q}'_C$, such that $(q'_O)_i \neq 0$ for all i, we can rewrite the constraints for some vectors $\mathbf{q}_L, \mathbf{q}_R, \mathbf{q}_O, \mathbf{q}_M, \mathbf{q}_C$ such that $\mathbf{q}_O = -1$ by a normalization process. On the other hand, $(q'_O)_i = 0$ in Plonk only for the case where i corresponds to public inputs or to a boolean constraint. As explained above, we enforce public input constraints separately by including them in $\mathbf{a}$; also, boolean constraints can be easily written enforcing $(q_O)_i = -1$: instead of requiring $a_j = b_j$, and $a_j - a_j b_j = 0$ for some b_j such that $\sigma(a_j) = b_j$ as suggested in Plonk, they can be written as $a_j = b_j = c_j$, $a_j b_j - c_j = 0$.

Bounded fan-out. Circuits with fan-out bounded by some constant V can naturally be encoded as an instance of $\mathcal{R}_{\mathsf{W\text{-}R1CS}}$ for the set $\mathcal{M}$ of matrices $\mathbf{W} = \begin{pmatrix} \mathbf{I} & \mathbf{0} & -\mathbf{F} \\ \mathbf{0} & \mathbf{I} & -\mathbf{G} \end{pmatrix}$ with $\mathbf{F}, \mathbf{G} \in \mathbb{F}^{m \times m}$ such that there are at most V non-zero elements per column in each. For circuits with bounded fan-out, we can set $l = l_b$, $\mathbf{q}_M = \mathbf{1}$, $\mathbf{q}_L = \mathbf{q}_R = \mathbf{q}_C = \mathbf{0}$. Note that any circuit can be transformed to a circuit with bounded fan-out by artificially augmenting the vector $\mathbf{c}$ and adding constraints of the form $c_i = c_j$ that ensure consistency. To express satisfiability of this system as an instance of $\mathcal{R}_{\mathsf{W\text{-}R1CS}}$, the additional constraints $c_i = c_j$ can rewritten as an equation involving left (or right) wires, i.e. $a_i = c_j$, that is encoded in the matrix $\mathbf{W}$, and a gate, i.e. $a_i = c_i$ (setting $(q_M)_i = (q_R)_i = (q_C)_i = 0$, $(q_L)_i = 1$). If the fan-out of a certain gate is κ, this requires extending $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ by approximately $3\kappa/V$ dummy variables.

3.3 Generalized Univariate Sumcheck

In this section, we revisit the sumcheck of Aurora [BCR+19]. As presented there, this argument allows to prove that the sum of the evaluations of a polynomial

in some multiplicative subgroup[3] $\mathbb{H}$ of a finite field $\mathbb{F}$ sum to some value σ. We generalize the argument to arbitrary sets $\mathbb{H} \subset \mathbb{F}$, solving an open problem posed there. Additionally, we give a simpler proof of the same result by connecting the sumcheck to polynomial evaluation and other basic properties of polynomials.

We prove a generalized sumcheck theorem below, and derive the sumcheck of Aurora as a corollary for the special case where $\mathbb{H}$ is a multiplicative subgroup. The intuition is simple: let $P_1(X)$ be a polynomial of arbitrary degree in $\mathbb{F}[X]$, and $P_2(X) = \sum_{i=1}^{m} \lambda_i(X) P_1(\mathsf{h}_i)$. Note that $P_1(X), P_2(X)$ are congruent modulo $z_H(X)$, and the degree of $P_2(X)$ is at most $m-1$. Then, when $P_2(X)$ is evaluated at an arbitrary point $v \in \mathbb{F}$, $v \notin \mathbb{H}$, $P_2(v) = \sum_{i=1}^{m} \lambda_i(v) P_1(\mathsf{h}_i)$. Thus, $P_2(v)$ is "almost" (except for the constants $\lambda_i(v)$) the sum of the evaluations of $P_1(\mathsf{h}_i)$. Multiplying by a normalizing polynomial, we get rid of the constants and obtain a polynomial that evaluated at v is the sum of any set of evaluations of interest. The sum will be zero if this product polynomial has a root at v.

Theorem 1 (Generalized Sumcheck)**.** *Let $\mathbb{H}$ be an arbitrary subset of some finite field $\mathbb{F}$ and $z_H(X)$ the vanishing polynomial of $\mathbb{H}$. For any $P(X) \in \mathbb{F}[X]$, $\mathcal{S} \subset \mathbb{H}$, and any $v \in \mathbb{F}, v \notin \mathbb{H}$, $\sum_{s \in \mathcal{S}} P(s) = \sigma$ if and only if there exist polynomials $H(X) \in \mathbb{F}[X]$, $R(X) \in \mathbb{F}_{\leq m-2}[X]$ such that*

$$P(X) N_{\mathcal{S},v}(X) - \sigma = (X - v) R(X) + z_H(X) H(X),$$

where $N_{\mathcal{S},v}(X) = \sum_{s \in \mathcal{S}} \lambda_s(v)^{-1} \lambda_s(X)$ and $\lambda_s(X)$ is the Lagrange polynomial associated to s and the set $\mathbb{H}$.

Proof. Observe that $P(X) = \sum_{\mathsf{h} \in \mathbb{H}} P(\mathsf{h}) \lambda_{\mathsf{h}}(X) \mod z_H(X)$. Therefore,

$$\begin{aligned} P(X) N_{\mathcal{S},v}(X) - \sigma &= \Big(\sum_{\mathsf{h} \in \mathbb{H}} P(\mathsf{h}) \lambda_{\mathsf{h}}(X) \Big) \Big(\sum_{s \in \mathcal{S}} \lambda_s(v)^{-1} \lambda_s(X) \Big) - \sigma \\ &= \Big(\sum_{s \in \mathcal{S}} P(s) \lambda_s(v)^{-1} \lambda_s(X) \Big) - \sigma \mod z_H(X). \end{aligned}$$

Let $Q(X) = \Big(\sum_{s \in \mathcal{S}} P(s) \lambda_s(v)^{-1} \lambda_s(X) \Big) - \sigma$. Note that $Q(v) = \sum_{s \in \mathcal{S}} P(s) - \sigma$. Thus, $\sum_{s \in \mathcal{S}} P(s) = \sigma$ if and only if $Q(X)$ is divisible by $X - v$. The claim follows from this observation together with the fact that $Q(X)$ is the unique polynomial of degree $m-1$ that is congruent with $P(X) N_{\mathcal{S},v}(X) - \sigma$. □

Lemma 2. *If $\mathcal{S} = \mathbb{H}$ is a multiplicative subgroup of $\mathbb{F}$, $N_{\mathbb{H},0}(X) = m$.*

Proof. Recall that, as $\mathbb{H}$ is a multiplicative subgroup, $\lambda_i(0) = 1/m$ for all $i = 1, \ldots, m$. Therefore, $N_{\mathbb{H},0}(X) = \sum_{i=1}^{m} \lambda_i(0)^{-1} \lambda_i(X) = m \sum_{i=1}^{m} \lambda_i(X) = m$.

□

[3] In fact, the presentation is more general as they also consider additive cosets, but we stick to the multiplicative case which is the one that has been used in other constructions of zkSNARKs.

As a corollary of Lemma 2 and the Generalized Sumcheck, we recover the univariate sumcheck: if $\mathbb{H}$ is a multiplicative subgroup, $\sum_{\mathsf{h}\in\mathbb{H}} P(\mathsf{h}) = \sigma$ if and only if there exist polynomials $R(X), H(X)$ with $\deg(R(X)) \leq m-2$ such that $P(X)m - \sigma = XR(X) + z_H(X)H(X)$.

3.3.1 Application to Linear Algebra Arguments

Several works [BCR+19, CHM+20, CFF+21] have observed that R1CS languages can be reduced to proving a Hadamard product relation and a linear relation, where the latter consists on showing that two vectors $\mathbf{x}, \mathbf{y}$ are such that $\mathbf{y} = \mathbf{M}\mathbf{x}$, or equivalently, that the inner product of $(\mathbf{y}, \mathbf{x})$ with all the rows of $(\mathbf{I}, -\mathbf{M})$ is zero. When matrices and vectors are encoded as polynomials for succinctness, constructing a PHP requires to express these linear algebra operations as polynomial identities.

For the Hadamard product relation, the basic observation is that, for any polynomials $A(X), B(X), C(X)$, the equation

$$A(X)B(X) - C(X) = H(X)z_H(X), \tag{3.1}$$

holds for some $H(X)$ if and only if it is the case that $(A(\mathsf{h}_1), \ldots, A(\mathsf{h}_m)) \circ (B(\mathsf{h}_1), \ldots, B(\mathsf{h}_m)) - (C(\mathsf{h}_1), \ldots, C(\mathsf{h}_m)) = 0$. In particular, if $A(X) = \mathbf{a}^\top \boldsymbol{\lambda}(X)$, $B(X) = \mathbf{b}^\top \boldsymbol{\lambda}(X)$ encode vectors $\mathbf{a}, \mathbf{b}$, then $C(X) \mod z_H(X)$ encodes $\mathbf{a} \circ \mathbf{b}$. This Hadamard product argument is one of the main ideas behind the zkSNARK of Gentry et al. [GGPR13] and follow-up work.

For linear relations, the following Theorem explicitly derives a polynomial identity that encodes the inner product relation from the univariate sumcheck. This connection in a different formulation is implicit in previous works [BCR+19, CHM+20, CFF+21].

Theorem 2 (Inner Product Polynomial Relation)**.** *For some $k \in \mathbb{N}$, let $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_k)$, $\mathbf{d} = (\mathbf{d}_1, \ldots, \mathbf{d}_k)$ be two vectors in $\mathbb{F}^{km}$, $\mathbf{y}_i, \mathbf{d}_i \in \mathbb{F}^m$ where y_{ij}, d_{ij} denote the jth element of vector $\mathbf{y}_i$ and $\mathbf{d}_i$, respectively. Consider also a multiplicative subgroup $\mathbb{H}$ of $\mathbb{F}$ of order m. Then, $\mathbf{y} \cdot \mathbf{d} = \sigma$ if and only if there exist $H(X), R(X) \in \mathbb{F}[X]$, $R(X)$ of degree at most $m-2$ such that the following relation holds:*

$$\mathbf{Y}(X) \cdot \mathbf{D}(X) - \frac{\sigma}{m} = XR(X) + z_H(X)H(X), \tag{3.2}$$

where $\mathbf{Y}(X) = (Y_1(X), \ldots, Y_k(X))$ is a vector of polynomials of arbitrary degree such that $Y_i(\mathsf{h}_j) = y_{ij}$ for all $i = 1, \ldots, k$, $j = 1, \ldots, m$, and $\mathbf{D}(X) = (D_1(X), \ldots, D_k(X))$ is such that $D_i(X) = \mathbf{d}_i^\top \boldsymbol{\lambda}(X)$.

Proof. Since $Y_i(\mathsf{h}_j) = y_{ij}$, for all i, j, $Y_i(X) = \mathbf{y}_i^\top \boldsymbol{\lambda}(X) \mod z_H(X)$. Therefore, $Y_i(X)D_i(X) = (\mathbf{y}_i^\top \boldsymbol{\lambda}(X))(\mathbf{d}_i^\top \boldsymbol{\lambda}(X)))$, and by the aforementioned properties of the Lagrange basis, this is also congruent modulo $z_H(X)$ to $(\mathbf{y}_i \circ \mathbf{d}_i)^\top \boldsymbol{\lambda}(X)$. Therefore,

$$\mathbf{Y}(X)\cdot\mathbf{D}(X)=\sum_{i=1}^{k}Y_i(X)D_i(X)=\sum_{i=1}^{k}(\mathbf{y}_i\circ\mathbf{d}_i)^\top\boldsymbol{\lambda}(X)\mod z_H(X)$$
$$=\left(\sum_{i=1}^{k}(\mathbf{y}_i\circ\mathbf{d}_i)^\top\right)\boldsymbol{\lambda}(X)\mod z_H(X).$$

By Theorem 1, $\big(\big(\sum_{i=1}^{k}(\mathbf{y}_i\circ\mathbf{d}_i)^\top\big)\boldsymbol{\lambda}(X)\big)N_{\mathbb{H},0}(X)-\sigma$ is divisible by X if and only if the sum of the coordinates of $\sum_{i=1}^{k}(\mathbf{y}_i\circ\mathbf{d}_i)$ is σ. The implication is also true after dividing by $N_{\mathbb{H},0}(X)=m$. The jth coordinate of $\sum_{i=1}^{k}(\mathbf{y}_i\circ\mathbf{d}_i)$ is $\sum_{i=1}^{k}y_{ij}d_{ij}$, thus the sum of all coordinates is $\sum_{j=1}^{m}\sum_{i=1}^{k}y_{ij}d_{ij}=\mathbf{y}\cdot\mathbf{d}$, which concludes the proof. □

In the rest of this work $\mathbb{H}$ will always be a multiplicative subgroup, both for simplicity (as $N_{\mathbb{H},0}=m$), and efficiency (due to the properties that Lagrange and vanishing polynomials associated to multiplicative subgroups have). However, Theorem 2 can be easily generalized to arbitrary sets $\mathbb{H}$ (just multiplying the left side of Eq. (3.2) by the polynomial $N_{\mathbb{H},0}(X)$ corresponding to set $\mathbb{H}$).

3.4 Algebraic Framework for W-R1CS

3.4.1 Checkable Subspace Sampling: Definition and Implications

In a *Checkable Subspace Sampling* (CSS) argument prover and verifier interactively agree on a polynomial $D(X)$ representing a vector $\mathbf{d}$ in the row space of a matrix $\mathbf{M}$. The fiber of the protocol is that $D(X)$ is calculated as a linear combination of encodings of the rows of $\mathbf{M}$ with some coefficients determined by the verifier, but the verifier does not need to calculate $D(X)$ itself (this would require the verifier to do linear work in the number of rows of $\mathbf{M}$). Instead, the prover can calculate this polynomial and then convince the verifier that it has been correctly computed.

Below we give the syntactical definition of Checkable Subspace Sampling. Essentially, a CSS scheme is similar to a PHP for a relation $\mathsf{R}_{\mathbf{M}}$, except that the statement $(\mathsf{cns}, D(X))$ is decided interactively, and the verifier has only oracle access to the polynomial $D(X)$. A CSS scheme can be used as a building block in a PHP, and the result is also a PHP.

Definition 18 (Checkable Subspace Sampling, CSS)**.** *A checkable subspace sampling argument over a field $\mathbb{F}$ defines some $Q, m\in\mathbb{N}$, a set of admissible matrices $\mathcal{M}$, a vector of polynomials $\beta(X)\in(\mathbb{F}[X])^m$, a coinspace $\mathcal{C}$, a sampling function* Smp :

$\mathcal{C} \to \mathbb{F}^Q$, *and a relation:*

$$\mathsf{R}_{\mathsf{CSS},\mathbb{F}} = \left\{ \begin{array}{ll} \big(\mathbf{M}, \mathsf{cns}, D(X)\big) \ : & \mathbf{M} \in \mathcal{M} \subset \mathbb{F}^{Q \times m}, D(X) \in \mathbb{F}[X], \mathsf{cns} \in \mathcal{C}, \\ & \mathbf{s} = \mathsf{Smp}(\mathsf{cns}), \text{ and } D(X) = \mathbf{s}^\top \mathbf{M} \boldsymbol{\beta}(X) \end{array} \right\}.$$

For any $\mathbf{M} \in \mathcal{M}$*, it also defines:*

$$\mathsf{R}_{\mathbf{M}} = \left\{ \big(\mathsf{cns}, D(X)\big) \ : \ \big(\mathbf{M}, \mathsf{cns}, D(X)\big) \in \mathsf{R}_{\mathsf{CSS},\mathbb{F}} \right\}.$$

It consists of three algorithms:

- $\mathcal{I}_{\mathsf{CSS}}$ *is the indexer: in an offline phase, on input* $(\mathbb{F}, \mathbf{M})$ *returns a set* $\mathcal{W}_{\mathsf{CSS}}$ *of* $\mathsf{n}(0)$ *polynomials* $\{p_{0,j}(X)\}_{j=1}^{n(0)} \in \mathbb{F}[X]$*. This algorithm is run once for each* $\mathbf{M}$*.*
- *Prover and Verifier proceed as in a PHP, namely, the verifier sends field elements to the prover and has oracle access to the polynomials outputted by both the indexer and the prover; this phase is run in two different stages:*
  - Sampling*:* $\mathcal{P}_{\mathsf{CSS}}$ *and* $\mathcal{V}_{\mathsf{CSS}}$ *engage in an interactive protocol. In some round, the verifier sends* $\mathsf{cns} \leftarrow \mathcal{C}$*, and the prover replies with* $D(X) = \mathbf{s}^\top \mathbf{M} \boldsymbol{\beta}(X)$*, for* $\mathbf{s} = \mathsf{Smp}(\mathsf{cns})$*.*
  - ProveSampling*:* $\mathcal{P}_{\mathsf{CSS}}$ *and* $\mathcal{V}_{\mathsf{CSS}}$ *engage in another interactive protocol to prove that* $(\mathsf{cns}, D(X)) \in \mathsf{R}_{\mathbf{M}}$*.*
- *When the proving phase is concluded, the verifier outputs a bit indicating acceptance or rejection.*

The vector $\boldsymbol{\beta}(X) = (\beta_1(X), \ldots, \beta_m(X))$ defines an encoding of vectors as polynomials: vector $\mathbf{v}$ is mapped to the polynomial $\mathbf{v}^\top \boldsymbol{\beta}(X) = \sum_{i=1}^m v_i \beta_i(X)$. When using a CSS for constructing an argument of membership in linear spaces as in the next section, we choose a characterization of inner product that is compatible with Lagrange polynomials. Thus, in this work, $\beta_i(X)$ is defined as $\lambda_i(X)$, the ith Lagrange polynomial associated to some multiplicative subgroup $\mathbb{H}$ of $\mathbb{F}$. Still, it also makes sense to consider CSS arguments for other polynomial encodings, e.g. the monomial basis or Laurent polynomials. In fact, the CSS argument in the amortized setting described in Section 3.5.7 is an abstraction of the helped mode of Sonic, that was presented for the encoding with Laurent polynomials.

We require a CSS argument to satisfy the following security definitions:

Perfect Completeness. If both prover and verifier are honest the output of the protocol is 1:

$$\Pr\left[\langle \mathcal{P}_{\mathsf{CSS}}(\mathbb{F}, \mathbf{M}, \mathsf{cns}), \mathcal{V}_{\mathsf{CSS}}^{\mathcal{W}_{\mathsf{CSS}}}(\mathbb{F}) \rangle = 1\right] = 1.$$

where the probability is taken over the random coins of prover and verifier.

Soundness. A checkable subspace sampling argument $(\mathcal{I}_{\mathsf{CSS}}, \mathcal{P}_{\mathsf{CSS}}, \mathcal{V}_{\mathsf{CSS}})$ is ϵ-sound if for all $\mathbf{M}$ and any polynomial time prover $\mathcal{P}^*_{\mathsf{CSS}}$, the following probability is smaller than ϵ:

$$\mathsf{Pr}\left[D^*(X) \neq \mathbf{s}^\top \mathbf{M}\boldsymbol{\beta}(X) \;\middle|\; \begin{array}{c}(\mathsf{cns}, D^*(X)) \leftarrow \mathsf{Sampling}\langle \mathcal{P}^*_{\mathsf{CSS}}(\mathbb{F}, \mathbf{M}, \mathsf{cns}), \mathcal{V}^{\mathcal{W}_{\mathsf{CSS}}}(\mathbb{F})\rangle; \\ \mathbf{s} = \mathsf{Smp}(\mathsf{cns});\ \langle \mathcal{P}^*_{\mathsf{CSS}}(\mathbb{F}, \mathbf{M}, \mathsf{cns}), \mathcal{V}^{\mathcal{W}_{\mathsf{CSS}}}_{\mathsf{CSS}}(\mathbb{F})\rangle = 1\end{array}\right]$$

The soundness of the CSS argument will ensure that the vector is sampled as specified by the coins of the verifier so the prover cannot influence its distribution. For a CSS argument to be useful, we additionally need that distribution induced by the sampling function is sufficiently "good". This is a geometric property that can be captured in the Elusive Kernel property defined below.

Definition 19. *A CSS argument is ϵ-elusive kernel*[4] *if*

$$\max_{\mathbf{t}\in\mathbb{F}^Q, \mathbf{t}\neq\mathbf{0}} \Pr\left[\mathbf{s}\cdot\mathbf{t} = 0 \,\middle|\, \mathbf{s} = \mathsf{Smp}(\mathsf{cns}); \mathsf{cns} \leftarrow \mathcal{C}\right] \leq \epsilon.$$

In practice, for most schemes, $\mathbf{s}$ is a vector of monomials or Lagrange basis polynomials evaluated at some point $x = \mathsf{cns}$, and this property is an immediate application of Schwartz-Zippel lemma, so we will not explicitly prove it for most of our CSS arguments. An exception is the argument of Section 3.5.6.

It is useful in some contexts to generalize the definition of CSS arguments to block matrices, that is, to extend the relation to tuples $\big(\mathbf{M}, \mathsf{cns}, \mathbf{D}(X)\big)$, where $\mathbf{M} = (\mathbf{M}_1, \ldots, \mathbf{M}_k)$ and $\mathbf{D}(X) = (D_1(X), \ldots, D_k(X))$ and $D_i(X) = \mathbf{s}^\top \mathbf{M}_i \boldsymbol{\beta}(X)$, and $\mathbf{M}_i \in \mathbb{F}^{Q\times m}$. This generalization is not necessary if correct sampling is proven for each block individually, but to save on proof size the proofs might be aggregated in some cases. This generalization is useful to formalize this technique.

3.4.2 Linear Arguments from Checkable Subspace Sampling

In this section we build a PHP for the universal relation of membership in linear subspaces:

$$\mathcal{R}_{\mathsf{LA}} = \left\{(\mathbb{F}, \mathbf{W}, \mathbf{y}) : \mathbf{W} \in \mathbb{F}^{Q\times km}, \mathbf{y} \in \mathbb{F}^{km} \text{ s.t. } \mathbf{W}\mathbf{y} = \mathbf{0}\right\},$$

using a CSS scheme as building block. That is, given a vector $\mathbf{y}$, the argument allows to prove membership in the linear space $\mathbf{W}^\perp = \{y \in \mathbb{F}^{km} : \mathbf{W}\mathbf{y} = \mathbf{0}\}$. Although relation $\mathcal{R}_{\mathsf{LA}}$ is polynomial-time decidable, it is not trivial to construct a polynomial holographic proof for it, as the verifier has only an encoding of $\mathbf{W}$ and $\mathbf{y}$.

A standard way to prove that some vector $\mathbf{y}$ is in $\mathbf{W}^\perp$ is to let the verifier sample a *sufficiently random* vector $\mathbf{d}$ in the row space of matrix $\mathbf{W}$, and prove $\mathbf{y}\cdot\mathbf{d} = 0$.

[4]The name is inspired by the property of t-elusiveness of [MRV16].

Naturally, the vector $\mathbf{y}$ must be declared before $\mathbf{d}$ is chosen. We follow this strategy to construct a PHP for $\mathcal{R}_{\mathsf{LA}}$, except that the vector $\mathbf{d}$ is sampled by the prover itself on input the coins of the verifier through a CSS argument.

As we have seen in Section 3.2, it is natural in our application to proving different variants of R1CS to consider matrices in blocks (We can think of a matrix consisting of one $3m \times 3m$ block in the case of Plonk). Thus, in this section we prove membership in $\mathbf{W}^{\perp}$ where the matrix is written in k blocks of columns, that is, $\mathbf{W} = (\mathbf{W}_1, \ldots, \mathbf{W}_k)$. The vectors $\mathbf{y}, \mathbf{d} \in \mathbb{F}^{km}$ are also written in blocks as $\mathbf{y}^{\top} = (\mathbf{y}_1^{\top}, \ldots, \mathbf{y}_k^{\top})$ and $\mathbf{d}^{\top} = (\mathbf{d}_1^{\top}, \ldots, \mathbf{d}_k^{\top})$.

Each block of $\mathbf{W}$, as well as the vectors $\mathbf{y}, \mathbf{d}$ can be naturally encoded, respectively, as a vector of polynomials or a single polynomial multiplying on the right by $\boldsymbol{\lambda}(X)$. However, we allow for additional flexibility in the encoding of $\mathbf{y}$: our argument is parameterized by a set of valid witnesses W_Y and a function $\mathcal{E}_Y : W_Y \to (\mathbb{F}[X])^k$ that determines how $\mathbf{y}$ is encoded as a polynomial. Thanks to this generalization we can use the argument as a black-box in our W-R1CS construction. There, valid witnesses are of the form $(\mathbf{a}, \mathbf{b}, \mathbf{q}_M \circ \mathbf{a} \circ \mathbf{b} + \mathbf{q}_L \circ \mathbf{a} + \mathbf{q}_R \circ \mathbf{b} + \mathbf{q}_C)$ and, for efficiency, its encoding will be $(A(X) = \mathbf{a}^{\top}\boldsymbol{\lambda}(X), B(X) = \mathbf{b}^{\top}\boldsymbol{\lambda}(X), q_M(X)A(X)B(X) + q_L(X)A(X) + q_R(X)B(X) + q_C(X)))$, for public polynomails $q_\gamma(X) = \mathbf{q}_\gamma^{\top}\boldsymbol{\lambda}(X)$, $\gamma \in \{M, L, R, C\}$, which in practice means that the last element does not need to be sent.

The argument goes as follows. The prover sends a vector of polynomials $\mathbf{Y}(X)$ encoding $\mathbf{y}$. The CSS argument is used to delegate to the prover the sampling of $\mathbf{d}_i^{\top}$, $i = 1, \ldots, k$ in the row space of $\mathbf{W}_i$. Then, the prover sends $\mathbf{D}(X)$ together with a proof that $\mathbf{y} \cdot \mathbf{d} = 0$. For this inner product argument to work, we resort to Theorem 2 that guarantees that, if $\mathcal{E}_Y$ is an encoding such that if $\mathcal{E}_Y(\mathbf{y}) = \mathbf{Y}(X)$, then $Y_i(\mathsf{h}_j) = y_{ij}$, the inner product relation holds if and only if the verification equation is satisfied for some $H(X), R(X)$.

Because of the soundness property of the CSS argument, the prover cannot influence the distribution of $\mathbf{d}$, which is sampled according to the verifier's coins. Therefore, if $\mathbf{Y}(X)$ passes the test of the verifier, $\mathbf{y}$ is orthogonal to $\mathbf{d}$. By the Elusive Kernel property of the CSS argument, $\mathbf{d}$ will be sufficiently random. As it is sampled after $\mathbf{y}$ is declared, this will imply that $\mathbf{y}$ is in $\mathbf{W}^{\perp}$.

Theorem 3. *When instantiated using a CSS scheme with perfect completeness, and when the encoding $\mathcal{E}_Y : W_Y \to \mathbb{F}[X]^k$ satisfies that, if $\mathcal{E}_Y(\mathbf{y}) = \mathbf{Y}(X)$, then $Y_i(\mathsf{h}_j) = y_{ij}$, the PHP of Fig. 3.1 has perfect completeness.*

Proof. By definition, $\mathbf{D}(X) = (\mathbf{s}^{\top}\mathbf{W}_1\boldsymbol{\lambda}(X), \ldots, \mathbf{s}^{\top}\mathbf{W}_k\boldsymbol{\lambda}(X))$, for $\mathbf{s} = \mathsf{Samp}(\mathsf{cns})$. Note that this is because the k instances of the CSS scheme are run in parallel and the same coins are used to sample each of the $\mathbf{d}_i$. Thus, $\mathbf{D}(X)$ is the polynomial encoding of $\mathbf{d} = (\mathbf{s}^{\top}\mathbf{W}_1, \ldots, \mathbf{s}^{\top}\mathbf{W}_k) = \mathbf{s}^{\top}\mathbf{W}$. Therefore, if $\mathbf{y}$ is in $\mathbf{W}^{\perp}$, $\mathbf{d} \cdot \mathbf{y} = \mathbf{s}^{\top}\mathbf{W}\mathbf{y} = \mathbf{0}$. By the characterization of inner product, as explained in Section 3.3, this implies that polynomials $H(X), R(X)$ satisfying the verification equation exist. □

Offline Phase: $\mathcal{I}_{\mathsf{LA}}(\mathbb{F}, \mathbf{W})$: For $i = 1, \ldots, k$, run the indexer $\mathcal{I}_{\mathsf{CSS}}$ on input $(\mathbb{F}, \mathbf{W}_i)$ to obtain the set $\mathcal{W}_{\mathsf{CSS}i}$ and output $\mathcal{W}_{\mathsf{LA}} = \bigcup_{i=1}^{k} \mathcal{W}_{\mathsf{CSS}i}$.

Online Phase: $\mathcal{P}_{\mathsf{LA}}$: On input a witness $\mathbf{y} \in W_Y \subset (\mathbb{F}^m)^k$, output $\mathbf{Y}(X) = \mathcal{E}_Y(\mathbf{y})$.
$\mathcal{P}_{\mathsf{LA}}$ and $\mathcal{V}_{\mathsf{LA}}$ run in parallel k instances of the CSS argument, with inputs $(\mathbb{F}, \mathbf{W}_i)$ and $\mathbb{F}$, respectively, and where the verifier is given oracle access to $\mathcal{W}_{\mathsf{CSS}i}$. The output is a set $\{(\mathsf{cns}, D_i(X))\}_{i=1}^{k}$, where cns are the same for all k instances. Define $\mathbf{D}(X) = (D_1(X), \ldots, D_k(X))$.
$\mathcal{P}_{\mathsf{LA}}$: Outputs $R(X) \in \mathbb{F}_{\leq m-2}[X], H(X)$ such that

$$\mathbf{Y}(X) \cdot \mathbf{D}(X) = XR(X) + z_H(X)H(X). \tag{3.3}$$

Decision Phase: Accept if and only if (1) $\deg(R) \leq m - 2$, (2) $\mathcal{V}^i_{\mathsf{CSS}}$ accepts $(\mathsf{cns}, D_i(X))$, and (3) the following equation holds:

$$\mathbf{Y}(X) \cdot \mathbf{D}(X) = XR(X) + z_H(X)H(X).$$

Figure 3.1: Argument for proving membership in $\mathbf{W}^{\perp}$, parameterized by the polynomial encoding $\mathcal{E}_Y : W_Y \to \mathbb{F}[X]^k$, and the set $W_Y \subset \mathbb{F}^{km}$.

Theorem 4. *Let* CSS *be* ϵ*-sound and* ϵ'*-Elusive Kernel, and* $\mathcal{E}_Y : W_Y \to \mathbb{F}[X]^k$ *an encoding such that if* $\mathcal{E}_Y(\mathbf{y}) = \mathbf{Y}(X)$, $Y_i(\mathsf{h}_j) = y_{ij}$. *Then, for any polynomial time adversary* $\mathcal{A}$ *against the soundness of PHP of Fig. 3.1:*

$$\mathsf{Adv}(\mathcal{A}) \leq \epsilon' + k\epsilon.$$

Further, the PHP satisfies 0*-knowledge soundness.*

Proof. Let $\mathbf{Y}^*(X) = (Y_1^*(X), \ldots, Y_k^*(X))$ be the output of a cheating $\mathcal{P}^*_{\mathsf{LA}}$ and $\mathbf{y}^* = (\mathbf{y}_1^*, \ldots, \mathbf{y}_k^*)$ the vector such that $Y_i^*(\mathsf{h}_j) = y_{ij}^*$. As a direct consequence of Theorem 2, $\mathbf{Y}^*(X) \cdot \mathbf{D}(X) = XR(X) + z_H(X)H(X)$ only if $\mathbf{y}^* \cdot \mathbf{d} = 0$, where $\mathbf{d}$ is the unique vector $\mathbf{d}$ such that $\mathbf{D}(X) = (\mathbf{d}_1^{\top}\boldsymbol{\lambda}(X), \ldots, \mathbf{d}_k^{\top}\boldsymbol{\lambda}(X))$.

On the other hand, the soundness of the CSS scheme guarantees that, for each i, the result of sampling $D_i(X)$ corresponds to the sample coins sent by the verifier, except with probability ϵ. Thus, the chances that the prover can influence the distribution of $\mathbf{D}(X)$ so that so that $\mathbf{y}^* \cdot \mathbf{d} = 0$ are at most $k\epsilon$. Excluding this possibility, a cheating prover can try to craft $\mathbf{y}^*$ in the best possible way to maximize the chance that $\mathbf{y}^* \cdot \mathbf{d} = 0$. Since $\mathbf{d}^{\top} = \mathbf{s}^{\top}\mathbf{W}$, and in a successful attack $\mathbf{y}^* \notin \mathbf{W}^{\perp}$, we can see that this possibility is bounded by the probability:

$$\max_{\mathbf{y}^* \notin \mathbf{W}^{\perp}} \Pr\left[\mathbf{d} \cdot \mathbf{y}^* = 0 \;\middle|\; \begin{array}{c} \mathsf{cns} \leftarrow \mathcal{C}; \\ \mathbf{s} = \mathsf{Smp}(\mathsf{cns}); \\ \mathbf{d} = \mathbf{s}^{\top}\mathbf{W} \end{array}\right] = \max_{\mathbf{y}^* \notin \mathbf{W}^{\perp}} \Pr\left[\mathbf{s}^{\top}\mathbf{W}\mathbf{y}^* = 0 \;\middle|\; \begin{array}{c} \mathsf{cns} \leftarrow \mathcal{C}; \\ \mathbf{s} = \mathsf{Smp}(\mathsf{cns}) \end{array}\right]$$

Since $\mathbf{s}^\top \mathbf{W}\mathbf{y}^* = \mathbf{s} \cdot (\mathbf{W}\mathbf{y}^*)$, and $\mathbf{W}\mathbf{y}^* \neq \mathbf{0}$, this can be bounded by ϵ', by the elusive kernel property of the CSS scheme.

For knowledge soundness, define the extractor $\mathcal{E}$ as the algorithm that runs the prover and, by evaluating $Y_i(X)$ in $\{\mathsf{h}_j\}_{j=1}^m$ for all $i \in [k]$, recovers $\mathbf{y}$. If the verifier accepts with probability greater than $\epsilon' + k\epsilon$, then $\mathbf{y}$ is such that $\mathbf{W}\mathbf{y} = \mathbf{0}$ with the same probability.

□

3.4.3 W-R1CS from Linear Arguments

Offline Phase: $\mathcal{I}_{\mathsf{W\text{-}R1CS}}(\mathbb{F}, m, l, l_b, \mathbf{W}, \mathbf{q}_M, \mathbf{q}_L, \mathbf{q}_R, \mathbf{q}_C)$ parses $\mathbf{W}$ as $(\mathbf{W}_a, \mathbf{W}_b, \mathbf{W}_c)$ and runs the indexer $\mathcal{I}_{\mathsf{LA}}$ on input $(\mathbb{F}, \mathbf{W}_a, \mathbf{W}_b, \mathbf{W}_c)$ to obtain the set $\mathcal{W}_{\mathsf{LA}} = \mathcal{W}_{\mathsf{LA}a} \cup \mathcal{W}_{\mathsf{LA}b} \cup \mathcal{W}_{\mathsf{LA}c}$.
For $S \in \{L, R, M, C\}$, the indexer computes polynomials $q_S(X) = \mathbf{q}_S^\top \boldsymbol{\lambda}(X)$.
Outputs $\mathcal{W}_{\mathsf{W\text{-}R1CS}} = \mathcal{W}_{\mathsf{LA}} \cup \{q_L(X), q_R(X), q_M(X), q_C(X)\}$

Online Phase:

- $\mathcal{P}_{\mathsf{W\text{-}R1CS}}$ Computes and outputs

$$A'(X) = \left(\sum_{j=l+1}^{m} a_j \lambda_j(X)\right) / t_l(X), B'(X) = \left(\left(\sum_{j=1}^{m} b_j \lambda_j(X)\right) - 1\right) / t_{l_b}(X),$$

  where $t_l(X) = \prod_{j=1}^{l}(X - \mathsf{h}_j)$, $t_{l_b}(X) = \prod_{j=1}^{l_b}(X - \mathsf{h}_j)$.

- $\mathcal{P}_{\mathsf{W\text{-}R1CS}}$ and $\mathcal{V}_{\mathsf{W\text{-}R1CS}}$ instantiate $\mathcal{P}_{\mathsf{LA}}(\mathbb{F}, \mathbf{W}, (\mathbf{a}, \mathbf{b}, \mathbf{q}_M \circ \mathbf{a} \circ \mathbf{b} + \mathbf{q}_L \circ \mathbf{a} + \mathbf{q}_R \circ \mathbf{b} + \mathbf{q}_C))$ and $\mathcal{V}_{\mathsf{LA}}^{\mathcal{W}_{\mathsf{LA}}}(\mathbb{F})$. Let $\mathbf{Y}(X) = (A(X), B(X), q_M(X)A(X)B(X) + q_L(X)A(X) + q_R(X)B(X) + q_C(X))$ be the polynomials $\mathcal{P}_{\mathsf{LA}}$ outputs in the first round.

Decision Phase: Defines $C_l(X) = \lambda_1(X) + \sum_{j=1}^{l-1} x_j \lambda_{j+1}(X)$ and accepts if and only if (1) $A(X) = A'(X)t_l(X) + C_l(X)$, (2) $B(X) = B'(X)t_{l_b}(X) + 1$, and (3) $\mathcal{V}_{\mathsf{LA}}$ accepts.

Figure 3.2: PHP for the universal relation $\mathcal{R}_{\mathsf{W\text{-}R1CS}}$.

In Fig. 3.2 we present a PHP for $\mathcal{R}_{\mathsf{W\text{-}R1CS}}$ that uses as building block a linear argument as in Section 3.4.2. The set of admissible matrices must coincide with the set of admissible matrices of the linear argument, which in fact depends on the admissible matrices of the CSS argument. For better flexibility, we present it for a matrix of three blocks. The PHP for $\mathcal{R}_{\mathsf{LA}}$ should be instantiated for $W_Y =$

$\{(\mathbf{a}, \mathbf{b}, \mathbf{q}_M \circ \mathbf{a} \circ \mathbf{b} + \mathbf{q}_L \circ \mathbf{a} + \mathbf{q}_R \circ \mathbf{b} + \mathbf{q}_C) : \mathbf{a}, \mathbf{b} \in \mathbb{F}^m\}$, $\mathcal{E}(\mathbf{a}, \mathbf{b}, \mathbf{q}_M \circ \mathbf{a} \circ \mathbf{b} + \mathbf{q}_L \circ \mathbf{a} + \mathbf{q}_R \circ \mathbf{b} + \mathbf{q}_C) = (\mathbf{a}^\top \boldsymbol{\lambda}(X), \mathbf{b}^\top \boldsymbol{\lambda}(X), (\mathbf{q}_M^\top \boldsymbol{\lambda}(X))(\mathbf{a}^\top \boldsymbol{\lambda}(X))(\mathbf{b}^\top \boldsymbol{\lambda}(X)) + (\mathbf{q}_L^\top \boldsymbol{\lambda}(X))(\mathbf{a}^\top \boldsymbol{\lambda}(X)) + (\mathbf{q}_R^\top \boldsymbol{\lambda}(X))(\mathbf{b}^\top \boldsymbol{\lambda}(X)) + (\mathbf{q}_C^\top \boldsymbol{\lambda}(X))$.

Theorem 5. *When instantiated with a complete, sound and knowledge soundness linear argument, the PHP of Fig. 3.2 satisfies completeness, soundness and knowledge-soundness.*

Proof. Completeness follows directly from the definition of $A'(X)$, $B'(X)$, $A(X)$, $B(X)$ and completeness of the linear argument. Soundness and knowledge soundness hold if the linear argument is sound as well, because $\mathcal{V}_{\mathsf{W\text{-}R1CS}}$ accepts if $\mathcal{V}_{\mathsf{LA}}$ accepts, meaning $\dot{\mathbf{W}}(\mathbf{a}, \mathbf{b}, \mathbf{q}_M \circ \mathbf{a} \circ \mathbf{b} + \mathbf{q}_L \circ \mathbf{a} + \mathbf{q}_R \circ \mathbf{b} + \mathbf{q}_C)^\top = 0$ and $\mathcal{R}_{\mathsf{W\text{-}R1CS}}$ holds, while for extraction it suffices to use the extractor of the linear argument. □

3.4.4 Adding Zero Knowledge

To achieve zero-knowledge, it is common to several works on pairing-based zk-SNARKS [CFF+21, CHM+20, GGPR13] to randomize the polynomial commitment to the witness with a polynomial that is a multiple of the vanishing polynomial. That is, the commitment to a vector $\mathbf{a}$ is $A(X) = \sum a_i \lambda_i(X) + z_H(X)h(X)$, where $z_H(X), \lambda_i(X)$ are defined as usual, and the coefficients of $h(X)$ are the randomness. In [GGPR13], $h(X)$ can be constant, since the commitment $A(X)$ in the final argument is evaluated at a single point. In other works where the commitment needs to support queries at several point values, $h(X)$ needs to be of higher degree. In Marlin, it is suggested to choose the degree according to the number of oracle queries to maximimize efficiency, and in Lunar this idea is developed into a fine-grained analysis and a vector with query bounds is specified for the compiler. Additionally, for this technique, the prover needs to send a masking polynomial to randomize the polynomial $R(X)$ of the inner product check. The reason is that this polynomial leaks information about $(A(X), B(X), q_M(X)A(X)B(X) + q_L(X)A(X) + q_R(X)B(X) + q_C(X)) \cdot \mathbf{D}(X) \mod z_H(X)$.

In this section, we show how to add zero-knowledge to the PHP for W-R1CS of Section 3.4.3 without sending additional polynomials. The approach is natural and a similar technique has also been used in [SZ20]. Let $(\mathsf{b}_A, \mathsf{b}_B, \mathsf{b}_R, \mathsf{b}_H)$ be the tuple of bounds on the number of polynomial evaluations seen by the verifier after compiling for the polynomials $A(X), B(X), R(X), H(X)$. To commit to a vector $\mathbf{y} \in \mathbb{F}^m$, we sample some randomness $\mathbf{r} \in \mathbb{F}^n$, where n is a function of $(\mathsf{b}_A, \mathsf{b}_B, \mathsf{b}_R, \mathsf{b}_H)$ to be specified (a small constant when compiling). The cardinal of $\mathbb{H}$ is denoted by $\tilde{m}$ in this section. A commitment is defined in the usual way for the vector $(\mathbf{y}, \mathbf{r})$, i.e. $\sum_{i=1}^{m} y_i \lambda_i(X) + \sum_{i=m+1}^{m+n} r_i \lambda_i(X)$, and, naturally, we require $m + n \leq \tilde{m}$. Our idea is to consider related randomness for $A(X), B(X)$ so that the additional randomness sums to 0 and does not interfere with the inner product argument. The novel approach is to enforce this relation of the randomness by adding one additional

Offline Phase: For $\tilde{m} = m + n$, the matrix of constraints is:

$$\tilde{\mathbf{W}} = \begin{pmatrix} \mathbf{W}_a & \mathbf{0}_{2m\times n} & \mathbf{W}_b & \mathbf{0}_{2m\times n} & \mathbf{W}_c & \mathbf{0}_{2m\times n} \\ \mathbf{0}_m^\top & \mathbf{1}_n^\top & \mathbf{0}_m^\top & \mathbf{1}_n^\top & \mathbf{0}_m^\top & \mathbf{0}_n^\top \end{pmatrix}$$

and polynomials $q_S(X)$, $S \in \{M, L, R, C\}$ are constructed from $\tilde{\mathbf{q}}_M = (\mathbf{q}_M, \mathbf{1}_n)$, $\tilde{\mathbf{q}}_L = (\mathbf{q}_L, \mathbf{0}_n)$, $\tilde{\mathbf{q}}_R = (\mathbf{q}_R, \mathbf{0}_n)$, $\tilde{\mathbf{q}}_C = (\mathbf{q}_C, \mathbf{0}_n)$.
Online Phase: $\mathcal{P}_{\mathsf{W\text{-}R1CS}}$ samples $\mathbf{r}_a \leftarrow \mathbb{F}^n, \mathbf{r}_b \leftarrow \mathbb{F}^n$ conditioned on $\sum_{i=1}^n r_{a,i} + r_{b,i} = 0$ and uses $\tilde{\mathbf{a}} := (1, \mathbf{x}, \mathbf{a}', \mathbf{r}_a)$, $\tilde{\mathbf{b}} := (\mathbf{1}_{l_b}, \mathbf{b}', \mathbf{r}_b)$, to construct $\tilde{A}(X)$ and $\tilde{B}(X)$, $\tilde{A}'(X)$ and $\tilde{B}'(X)$ as before.

Figure 3.3: Modification of the PHP for $\mathcal{R}_{\mathsf{W\text{-}R1CS}}$ to achieve zero-knowledge.

constraint to $\mathbf{W}$. The marginal cost of this for the prover is minimal. Starting from the PHP of Fig. 3.2 we introduce the changes described in Fig. 3.3, note that the omitted parts in the latter are identical to the ones in the former.

Theorem 6. *With the modification described in Fig. 3.3 the PHP of Fig. 3.2 is perfectly complete, sound, knowledge-sound, perfect zero-knowledge and* $(\mathsf{b}_A, \mathsf{b}_B, \mathsf{b}_R, \mathsf{b}_H)$*-bounded honest-verifier zero-knowledge if* $n \geq \big(\mathsf{b}_A + \mathsf{b}_B + \mathsf{b}_R + \mathsf{b}_H + 1\big)/2$*, and* $n \geq \max(\mathsf{b}_A, \mathsf{b}_B)$.

Proof. The only difference with the previous argument is the fact that the matrix of constraints has changed, which is now $\tilde{\mathbf{W}}$. For completeness, observe that the additional constraint makes sure that $\sum_{i=1}^n r_{a,i} + r_{b,i} = 0$, and an honest prover chooses the randomness such that this holds. On the other hand, the sumcheck theorem together with this equation guarantee that the randomness does not affect the divisibility at 0 of $(\tilde{A}(X), \tilde{B}(X), q_M(X)\tilde{A}(X)\tilde{B}(X) + q_L(X)\tilde{A}(X) + q_R(X)\tilde{B}(X) + q_C(X))) \cdot \mathbf{D}(X) \mod z_H(X)$.

For soundness, note that $\tilde{\mathbf{W}}\left(\tilde{\mathbf{a}}^\top, \tilde{\mathbf{b}}^\top, (\tilde{\mathbf{q}}_M \circ \tilde{\mathbf{a}} \circ \tilde{\mathbf{b}} + \tilde{\mathbf{q}}_L \circ \tilde{\mathbf{a}} + \tilde{\mathbf{q}}_R \circ \tilde{\mathbf{b}} + \tilde{\mathbf{q}}_C)^\top\right) = \mathbf{0}$, is equivalent to 1) $\mathbf{W}_a\mathbf{a} + \mathbf{W}_b\mathbf{b} + \mathbf{W}_c(\mathbf{q}_M \circ \mathbf{a} \circ \mathbf{b} + \mathbf{q}_L \circ \mathbf{a} + \mathbf{q}_R \circ \mathbf{b} + \mathbf{q}_C) = 0$, and 2) $\sum_{i=1}^n r_{a,i} + r_{b,i} = 0$, for $\mathbf{a} := (1, \mathbf{x}, \mathbf{a}')$, $\mathbf{b} := (\mathbf{1}_{l_b}, \mathbf{b}')$. This is because the first two blocks of constraints have 0s in the columns corresponding to $\mathbf{r}_a, \mathbf{r}_b$, and the other way around for the last constraint. Therefore, by the soundness of the linear argument $\sum_{i=1}^n r_{a,i} + r_{b,i} = 0$, and the randomness does not affect divisibility at 0 of $(\tilde{A}(X), \tilde{B}(X), q_M(X)\tilde{A}(X)\tilde{B}(X) + q_L(X)\tilde{A}(X) + q_R(X)\tilde{B}(X) + q_C(X)))^\top \cdot \mathbf{D}(X)$ mod $z_H(X)$, so the same reasoning used for the argument of Fig. 3.2 applies.

Perfect zero-knowledge of the PHP is immediate, as all the messages in the CSS procedure contain only public information and the rest of the information exchanged are oracle polynomials.

We now prove honest-verifier bounded zero-knowledge. The simulator is similar to [CFF+21](Th. 4.7), but generalized to the distribution of $\mathbf{D}(X)$ induced by the underlying CSS scheme. The simulator gets access to the random tape of the honest

verifier and receives x and the coins of the CSS scheme, as well as a list of its checks. It creates honestly all the polynomials of the CSS argument, since these are independent of the witness.

For an oracle query at point γ, the simulator samples uniform random values $A'_\gamma, B'_\gamma, R_{\gamma,t}$ in $\mathbb{F}$ and declares them, respectively, as $A'(\gamma), B'(\gamma), R(\gamma)$. It then defines the rest of the values to be consistent with them. More precisely, let $\mathbf{D}(X)^\top = \mathbf{s}^\top \mathbf{W} \lambda(X) = (D_a(X), D_b(X), D_{ab}(X))$ be the output of the CSS argument, which the simulator can compute with the CSS coins. Then, the simulator sets:

$$A_\gamma = A'_\gamma t_l(\gamma) + \sum_{i=1}^{l} x_i \lambda_i(\gamma), \qquad B_\gamma = B'_\gamma t_l(\gamma) + 1,$$

$$p_\gamma = D_a(\gamma) A_\gamma + D_b(\gamma) B_\gamma + D_{ab}(\gamma)\big(q_M(\gamma) A_\gamma B_\gamma + q_L(\gamma) A_\gamma + q_R(\gamma) B_\gamma + q_C(\gamma)\big)$$

$$H_{t\gamma} = (p_\gamma - \gamma R_{t,\gamma})/t(\gamma),$$

where Q_γ for $Q \in \{A', B', R, H\}$ is declared as $Q(\gamma)$. The simulator keeps a table of the computed values to answer consistently the oracle queries.

We now argue that the queries have the same distribution as the evaluations of the prover's polynomials if all the queries γ are in $\mathbb{F} \setminus \mathbb{H}$. Since the verifier is honest, and $|\mathbb{H}|$ is assumed to be a negligible fraction of the field elements, we can always assume this is the case. In this case, the polynomial encoding of $\mathbf{r}_a, \mathbf{r}_b$ acts as a masking polynomial for $A'(X), B'(X), R(X), H(X)$ and taking into account that $\sum_{i=1}^{n} r_{a,i} + r_{b,i} = 0$ to have the same distribution it is sufficient that $2n - 1 \geq \mathsf{b}_A + \mathsf{b}_B + \mathsf{b}_R + \mathsf{b}_H$, and $n \geq \max(\mathsf{b}_A + \mathsf{b}_B)$, as stated in the theorem. Therefore, bounded zero-knowledge is proven. □

Combining CSS schemes

Since a CSS scheme outputs a linear combination of the rows of a matrix $\mathbf{M}$, different instances of a CSS scheme can be easily combined with linear operations. More precisely, given a matrix $\mathbf{M}$ that can be written as $\begin{pmatrix}\mathbf{M}_1 \\ \mathbf{M}_2\end{pmatrix}$, we can use a different CSS arguments for each $\mathbf{M}_i$[5] Since all current constructions of CSS arguments have limitations in terms of the types of matrices they apply to, this opens the door to decomposing the matrix of constraints into blocks that admit different efficient CSS arguments. That is, one reason to divide the matrix $\mathbf{M}$ into blocks is to have a broader class of admissible matrices. Another reason is efficiency, since if a block that is either $\mathbf{0}$ or the identity matrix, the verifier can open the polynomial $D(X)$ itself, saving on the number of polynomials that need to be sent. More specifically, for our final construction, we will often split a matrix into two blocks of m rows,

[5]The naive approach would run both CSS arguments in parallel, but savings are possible by batching the proofs.

$\mathbf{M} = \begin{pmatrix}\mathbf{M}_1\\ \mathbf{M}_2\end{pmatrix}$, use the same CSS argument for each matrix with the same coins, and combine them to save on communication. More precisely, if $\mathbf{s} = \mathsf{Smp}(\mathsf{cns})$, and $D_1(X) = \mathbf{s}^\top \mathbf{M}_1 \lambda(X)$ and $D_2(X) = \mathbf{s}^\top \mathbf{M}_2 \lambda(X)$ are the polynomials associated to $\mathbf{M}_1, \mathbf{M}_2$, we will modify the CSS argument so that it sends $D_1(X) + zD_2(X)$ for some challenge z chosen by the verifier, instead of $D_1(X)$ and $D_2(X)$ individually. Note that $D_1(X) + zD_2(X) = (\mathbf{s}^\top, z\mathbf{s}^\top)\mathbf{M}\lambda(X)$, that is, this corresponds to a CSS argument where the sampling coefficients depend on z also.

We note that this cannot be done generically. The success of this technique depends on the underlying CSS argument and the type of admissible matrices. Intuitively, this modification corresponds to implicitly constructing a CSS argument for the matrix $\mathbf{M}_1 + z\mathbf{M}_2$, so it is necessary that: a) the polynomials computed by the indexer of the CSS argument for $\mathbf{M}_1, \mathbf{M}_2$ can be combined, upon receiving the challenge z, to the CSS indexer polynomials of $\mathbf{M}_1 + z\mathbf{M}_2$, and b) that $\mathbf{M}_1 + z\mathbf{M}_2$ is an admissible matrix for this CSS argument. For instance, if $\mathbf{M}_1, \mathbf{M}_2$ has K non-zero entries each, and the admissible matrices of a CSS instance must have at most K non-zero entries, then $\mathbf{M}_1 + z\mathbf{M}_2$ is not generally an admissible matrix. We will be using this optimization for our final PHP for sparse matrices, and we will see there that these conditions are met in this case.

3.5 Instantiation of CSS Arguments

Given the results of the previous section, to construct a PHP for our W-R1Cs argument, it is sufficient to design a CSS scheme for matrices $\mathbf{M} \in \mathbb{F}^{m\times m}$ and then use it on all the blocks of $\mathbf{W}$. In this section, we give several novel CSS arguments for different types of square matrices.

On the same line, matrices $\mathbf{M} \in \mathbb{F}^{m\times m}$ can be naturally encoded as a bivariate polynomial as $P(X,Y) = \boldsymbol{\alpha}(Y)^\top \mathbf{M} \boldsymbol{\beta}(X)$, for some $\boldsymbol{\alpha}(Y) \in \mathbb{F}[Y]^m, \boldsymbol{\beta}(X) \in \mathbb{F}[X]^m$. Let $\mathbf{m}_i^\top$ be the ith row of $\mathbf{M}$, and $P_i(X) = \mathbf{m}_i^\top \boldsymbol{\beta}(X)$. Then,

$$P(X,x) = \boldsymbol{\alpha}(x)^\top \mathbf{M} \boldsymbol{\beta}(X) = \sum_{i=1}^{m} \alpha_i(x) P_i(X).$$

That is, the polynomial $P(X,x)$ is a linear combination of the polynomials associated to the rows of $\mathbf{M}$ via the encoding defined by $\boldsymbol{\beta}(X)$, with coefficients $\alpha_i(x)$. This suggests to define a CSS scheme where, in the sampling phase, the verifier sends the challenge x and the prover replies with $D(X) = P(X,x)$, and, in the proving phase, the prover convinces the verifier that $D(X)$ is correctly sampled from coins x. This approach appears, implicitly or explicitly, in Sonic and most follow-up work we are aware of.

In Sonic, $\boldsymbol{\alpha}(Y), \boldsymbol{\beta}(X)$ are vectors of Laurent polynomials. In Marlin, Lunar and most of the constructions in this work, $\boldsymbol{\alpha}(Y) = \boldsymbol{\lambda}(Y)$, and $\boldsymbol{\beta}(X) = \boldsymbol{\lambda}(X)$. The

choice of $\boldsymbol{\beta}(X)$ is to make the encoding compatible with the inner product defined by the sumcheck, and the choice of $\boldsymbol{\alpha}(Y)$ is necessary for the techniques used in the proving phase of the CSS schemes that will be detailed in this Section.

For the proving phase, the common strategy is to follow the general template introduced in Sonic: the verifier samples a challenge $y \in \mathbb{F}$, checks that $D(y)$ is equal to a value σ sent by the prover, and that $\sigma = P(y, x)$ (through what is called a signature of correct computation, as in [PST13]). This proves that $D(X) = P(X, x)$. The last one is the challenging step, and is in fact, the main technical novelty of each of the mentioned previous works. In all of them, this is achieved by restricting the sets of matrices $\mathbf{M}$ to have a special structure: in Sonic they need to be sums of permutation matrices, and in Marlin, as later also Lunar and Vampire, arbitrary matrices with at most K non-zero entries, while Plonk only considers sums of permutation matrices.

This section is organized as follows. We start by giving an overview of our new techniques below. In Section 3.5.1, we explain a basic CSS scheme, that works only for *simple matrices*, i.e., matrices with at most one non-zero element per column, followed by a scheme for matrices with at most V non-zero elements per column, for some small bound V, in Section 3.5.2. In Section 3.5.3, we see how to compose these checks to achieve a CSS argument for arbitrary sparse matrices $\mathbf{M}$ with at most K non-zero elements, where K is the size of a multiplicative subgroup $\mathbb{K} \subset \mathbb{F}$. A similar technique than the one for sum of basic matrices is used in Section 3.5.4 to generalize the latter argument to matrices that can be written as a sum of V matrices of sparsity K, resulting on a scheme for matrices with sparsity VK that uses the same multiplicative subgroup and does not increase the communication complexity with respect to the one for matrices with sparsity K. In Section 3.5.5 we observe that our results also apply to low tensor rank matrices. Finally, in Section 3.5.6 we provide a construction that, at the best of our knowledge, is the first efficient CSS that works for *arbitrary* matrices, and in Section 3.5.7 we show that the helped version of Sonic can be instantiated as a CSS as well.

Overview of New Techniques

As in previous works([CHM+20, CFF+21]), we consider two disjoint subgroups of roots of unity, $\mathbb{H} = \{\mathsf{h}_1, \ldots, \mathsf{h}_m\}$, $\mathbb{K} = \{\mathsf{k}_1, \ldots \mathsf{k}_K\}$ with Lagrange and vanishing polynomials $\{\lambda_j(X)\}_{j=1}^m, z_H(X)$ and $\{\mu_\ell(X)\}_{\ell=1}^K, z_K(X)$, respectively.

Most of the results introduced in this section are CSS schemes for matrices $\mathbf{M} = (m_{i,j}) \in \mathbb{F}^{m \times m}$ that are sparse, that is, have some bounded amount of non-zero elements. Assuming these non-zero entries are ordered, a sparse matrix can be represented, as proposed in Marlin, by three functions $\mathsf{v} : \mathbb{K} \to \mathbb{F}$, $\mathsf{r} : \mathbb{K} \to [m]$, $\mathsf{c} : \mathbb{K} \to [m]$ such that $P(X, Y) = \sum_{\ell=1}^K \mathsf{v}(\mathsf{k}_\ell)\lambda_{\mathsf{r}(\mathsf{k}_\ell)}(Y)\lambda_{\mathsf{c}(\mathsf{k}_\ell)}(X)$, where the ℓth non-zero entry is $\mathsf{v}(\mathsf{k}_\ell) = m_{\mathsf{r}(\mathsf{k}_\ell),\mathsf{c}(\mathsf{k}_\ell)}$. If the matrix has less than K non-zero entries $\mathsf{v}(\mathsf{k}_\ell) = 0$, for $\ell = |\mathbf{M}| + 1, \ldots, K$, and $\mathsf{r}(\mathsf{k}_\ell), \mathsf{c}(\mathsf{k}_\ell)$ are defined arbitrarily. We borrow

this representation but design our own CSS schemes by following a "linearization strategy".

To see that $P(y, x)$ is correctly evaluated, we observe that it can be written as:

$$P(y,x) = \big(\lambda_{\mathsf{r}(\mathsf{k}_1)}(x), \ldots, \lambda_{\mathsf{r}(\mathsf{k}_K)}(x)\big) \cdot \big(\mathsf{v}(\mathsf{k}_1)\lambda_{\mathsf{c}(\mathsf{k}_1)}(y), \ldots, \mathsf{v}(\mathsf{k}_K)\lambda_{\mathsf{c}(\mathsf{k}_K)}(y)\big).$$

We define low degree extensions of each of these vectors respectively as:

$$e_x(X) = \sum_{\ell=1}^{K} \lambda_{\mathsf{r}(\mathsf{k}_\ell)}(x)\mu_\ell(X), \qquad e_y(X) = \sum_{\ell=1}^{K} \mathsf{v}(\mathsf{k}_\ell)\lambda_{\mathsf{c}(\mathsf{k}_\ell)}(y)\mu_\ell(X).$$

If the prover can convince the verifier that $e_x(X), e_y(X)$ are correctly computed, then it can show that $P(y, x) = \sigma$ by using the inner product argument of Section 3.3 to prove that the sum of $e_x(X)e_y(X) \mod z_H(X)$ at $\mathbb{K}$ is σ.

Observe that $e_x(X) = \lambda(x)^\top \mathbf{M}_x \mu(X)$ and $e_y(X) = \lambda(y)^\top \mathbf{M}_y \mu(X)$, for some matrices $\mathbf{M}_x, \mathbf{M}_y$ with at most one non-zero element per column. To prove they are correctly computed it suffices to design a CSS argument for these simple matrices. This can be done in a much simpler way than in Marlin (and as in Lunar, that uses a similar technique), who prove directly that a low degree extension of $e_x(X)e_y(X)$ is correctly computed (intuitively, theirs is a quadratic check that requires the indexer to publish more information, as verifiers can only do linear operations in the polynomials output by it). Still, our technique is similar to theirs: given an arbitrary polynomial $e_x(X) = \sum_{\ell=1}^{K} \mathsf{v}(\mathsf{k}_\ell)\lambda_{\mathsf{f}(\mathsf{k}_\ell)}(x)\mu_\ell(X)$, for some function $\mathsf{f} : \mathbb{K} \to [m]$, we can "complete" $\lambda_{\mathsf{f}(\mathsf{k}_\ell)}(x)$ with the missing term $(x - \mathsf{h}_{\mathsf{f}(\mathsf{k}_\ell)})$ to get the vanishing polynomial $z_H(x)$. The key insight is that the low degree extension of these "completing terms" is $x - v_1(X)$, where $v_1(X) = \sum_{\ell=1}^{K} \mathsf{h}_{\mathsf{f}(\mathsf{k}_\ell)}\mu_\ell(X)$ can be computed by the indexer.

The encoding for sparse matrices requires K to be at least $|\mathbf{M}|$, and generating a field with this large multiplicative subgroup can be a problem. We consider a generalization to matrices $\mathbf{M}$ of a special form with sparsity KV, for any $V \in \mathbb{N}$. The interesting point is that communication complexity does not grow with V, and only the number of indexer polynomials grows (as $2V + 2$). This generalization is constructed from the argument for sums of basic matrices presented in Section 3.5.2.

We stress the importance of the linearization step: it not only allows for a simple explanation of underlying techniques for the proving phase, but also for generalizations such as the ones in Sections 3.5.2, 3.5.3 and 3.5.5. The argument for basic matrices is also the key to our most efficient construction.

3.5.1 Basic Matrices

Our basic building block is a CSS argument for what we call *Basic Matrices*, that is, matrices $\mathbf{M} = (m_{ij}) \in \mathbb{F}^{m \times K}$ with at most one non-zero value in each column,

$$\mathcal{M} = \{\mathbf{M} \in \mathbb{F}^{m\times K} : \forall\ j \in [m]\ \exists!\ \ell \in [K]\ s.t.\ m_{j\ell} \neq 0\}.$$

In particular, if $K > m$, $|\mathbf{M}| \leq K$. We define two functions associated to $\mathbf{M}$, $\mathsf{v} : \mathbb{K} \to \mathbb{F}$, $\mathsf{f} : \mathbb{K} \to [m]$. Given an element $\mathsf{k}_\ell \in \mathbb{K}$, $\mathsf{v}(\mathsf{k}_\ell) = m_{\mathsf{f}(\mathsf{k}_\ell),\ell} \neq 0$, i.e., function v outputs the only non zero value of column ℓ and f the corresponding row; if such a value does not exist set $\mathsf{v}(\mathsf{k}_\ell) = 0$ and $\mathsf{f}(\mathsf{k}_\ell)$ arbitrarily.

We define the polynomial $P(X, Y)$ such that $D(X) = P(X, x)$ as $P(X, Y) = \lambda(Y)^\top \mathbf{M}\rho(X)$. Observe that, by definition of v and f,

$$P(X, Y) = \sum_{\ell=1}^{K} \mathsf{v}(\mathsf{k}_\ell)\lambda_{\mathsf{f}(\mathsf{k}_\ell)}(Y)\rho_\ell(X).$$

Offline Phase: $\mathcal{I}_{\mathsf{CSS}}\big(\mathbb{F}, \mathbf{M}\big)$ outputs $\mathcal{W}_{\mathsf{CSS}} = \{v_1(X), v_2(X)\}$, where

$$v_1(X) = \sum_{\ell=1}^{K} \mathsf{h}_{\mathsf{f}(\mathsf{k}_\ell)}\rho_\ell(X), \qquad v_2(X) = m^{-1}\sum_{\ell=1}^{K} \mathsf{v}(\mathsf{k}_\ell)\mathsf{h}_{\mathsf{f}(\mathsf{k}_\ell)}\rho_\ell(X).$$

Online Phase: Sampling: $\mathcal{V}_{\mathsf{CSS}}$ outputs $x \leftarrow \mathbb{F}$ and $\mathcal{P}_{\mathsf{CSS}}$ sends $D(X) = P(X, x)$. ProveSampling: $\mathcal{P}_{\mathsf{CSS}}$ finds and outputs $H_k(X)$ such that

$$D(X)\big(x - v_1(X)\big) = z_H(x)v_2(X) + H_k(X)z_K(X)$$

Decision Phase: Accept if and only if (1) $\deg D(X) \leq K-1$, and (2) $D(X)\big(x - v_1(X)\big) = z_H(x)v_2(X) + H_k(X)z_K(X)$.

Figure 3.4: A simple CSS scheme for matrices with at most one non-zero element per column.

Theorem 7. *The argument of Fig. 3.4 satisfies completeness and perfect soundness.*

Proof. When evaluated in any $\mathsf{k}_\ell \in \mathbb{K}$, the right side of the verification equation is $z_H(x)v_2(\mathsf{k}_\ell) = z_H(x)\mathsf{v}(\mathsf{k}_\ell)\mathsf{h}_{\mathsf{f}(\mathsf{k}_\ell)}m^{-1}$. Completeness follows from the fact that the left side is:

$$D(\mathsf{k}_\ell)(x - v_1(\mathsf{k}_\ell)) = \big(\mathsf{v}(\mathsf{k}_\ell)\lambda_{\mathsf{f}(\mathsf{k}_\ell)}(x)\big)\big(x - \mathsf{h}_{\mathsf{f}(\mathsf{k}_\ell)}\big) = z_H(x)\mathsf{v}(\mathsf{k}_\ell)m^{-1}\mathsf{h}_{\mathsf{f}(\mathsf{k}_\ell)}.$$

For soundness, note that the degree of $D(X)$ is at most $K-1$ and that the left side of the verification is $D(\mathsf{k}_\ell)(x - v_1(\mathsf{k}_\ell))$, so $D(\mathsf{k}_\ell) = z_H(x)\mathsf{v}(\mathsf{k}_\ell)m^{-1}\mathsf{h}_{\mathsf{f}(\mathsf{k}_\ell)}(x - \mathsf{h}_{\mathsf{f}(\mathsf{k}_\ell)})^{-1} = \mathsf{v}(\mathsf{k}_\ell)\lambda_{\mathsf{f}(\mathsf{k}_\ell)}$, for all $\mathsf{k}_\ell \in \mathbb{K}$. Thus, $D(X) = \sum_{\ell=1}^{K} \mathsf{v}(\mathsf{k}_\ell)\lambda_{\mathsf{f}(\mathsf{k}_\ell)}\rho_\ell(X)$. □

3.5.2 Sums of Basic Matrices

In this section, we use $\mathbf{M}$ for a matrix in $\mathbb{F}^{m\times K}$ that can be written as $\sum_{i=1}^{V}\mathbf{M}_i$, with each $\mathbf{M}_i$ having at most one non-zero element in each column.

$$\mathcal{M}=\{\mathbf{M}\in\mathbb{F}^{m\times K}:\mathbf{M}=\sum_{i=1}^{V}\mathbf{M}_i \ s.t.\ \forall\ i\in[V]\ \mathbf{M}_i \text{ is a Basic Matrix}\}.$$

We define two functions associated to each $\mathbf{M}_i$, $\mathsf{v}_i:\mathbb{K}\to\mathbb{F}$, $\mathsf{f}_i:\mathbb{K}\to[m]$ as in Section 3.5.1. This type of matrices will be used to design a generalization of the CSS argument for sums of sparse matrices in Section 3.5.3. Also, in Section 3.5 we use this argument in the context where $\mathbf{M}$ is a matrix in $\mathbb{F}^{K\times m}$. In that case, the role of the multiplicative subgroups $\mathbb{K},\mathbb{H}$ should be inversed.

Define $P(X,Y)=\lambda(Y)^\top\mathbf{M}\rho(X)$, and $D(X)=P(X,x)$. Observe that

$$P(X,Y)=\sum_{i=1}^{V}\sum_{\ell=1}^{K}\mathsf{v}_i(\mathsf{k}_\ell)\lambda_{\mathsf{f}_{i(\mathsf{k}_\ell)}}(Y)\rho_\ell(X).$$

Let $S_\ell=\{\mathsf{f}_i(\mathsf{k}_\ell):i\in[V]\}$, and $S_\ell^c=[K]-S_\ell$. The intuition is that, since there are at most V non zero $\mathsf{v}_i(\mathsf{k}_\ell)$ for each ℓ, we can factor as:

$$P(\mathsf{k}_\ell,x)=\sum_{i=1}^{V}\mathsf{v}_i(\mathsf{k}_\ell)\lambda_{\mathsf{f}_{i(\mathsf{k}_\ell)}}(x)=\prod_{s\in S_\ell^c}(x-\mathsf{h}_s)R_\ell(x),$$

where $R_\ell(X)$ is a polynomial of degree V. So, to "complete" $P(\mathsf{k}_\ell,x)$ to be a multiple of $z_H(x)$, we need to multiply it by $\prod_{s\in S_\ell}(x-\mathsf{h}_s)$, and the result will be $z_H(x)R_\ell(x)$. The trick is that $\hat{I}_\ell(Y)=\prod_{s\in S_\ell}(Y-\mathsf{h}_s)$, and $R_\ell(X)$ are polynomials of degrees V, $V-1$, respectively. Thus, if the indexer publishes the coefficients of these polynomials in the monomial basis, they can be reconstructed by the verifier with coefficients $1,x,\ldots,x^V$.

Theorem 8. *The argument of Fig. 3.5 satisfies completeness and perfect soundness.*

Proof. When evaluated in any $\mathsf{k}_\ell\in\mathbb{K}$, the right side of the verification equation is:

$$\begin{aligned}z_H(x)\hat{R}_x(x)&=\frac{z_H(x)}{m}\sum_{i=1}^{V}\mathsf{v}_i(\mathsf{k}_\ell)\mathsf{h}_{\mathsf{f}_{i(\mathsf{k}_\ell)}}\prod_{s\in S_\ell-\{\mathsf{f}_i(\mathsf{k}_\ell)\}}(x-\mathsf{h}_s)\\&=\sum_{i=1}^{V}\mathsf{v}_i(\mathsf{k}_\ell)\frac{\mathsf{h}_{\mathsf{f}_{i(\mathsf{k}_\ell)}}}{m}\frac{z_H(x)}{x-\mathsf{h}_{\mathsf{f}_{i(\mathsf{k}_\ell)}}}\prod_{s\in S_\ell}(x-\mathsf{h}_s)=\prod_{s\in S_\ell}(x-\mathsf{h}_s)\sum_{i=1}^{V}\mathsf{v}_i(\mathsf{k}_\ell)\lambda_{\mathsf{f}_{i(\mathsf{k}_\ell)}}(x).\end{aligned}$$

The left side of the equation is $D(\mathsf{k}_\ell)\hat{I}_x(\mathsf{k}_\ell) = \big(\sum_{i=1}^{V} \mathsf{v}_i(\mathsf{k}_\ell)\lambda_{\mathsf{f}_i(\mathsf{k}_\ell)}(x)\big)\big(\prod_{s\in S_\ell}(x - \mathsf{h}_s)\big)$, so completeness is immediate.

Offline Phase: $\mathcal{I}_{\mathsf{CSS}}\big(\mathbb{F}, \mathbf{M}\big)$: Define the polynomials $\hat{R}_\ell(Y), \hat{I}_\ell(Y)$, and its coefficients $\hat{R}_{\ell j}, \hat{I}_{\ell j}$:

$$\hat{R}_\ell(Y) = \frac{1}{m}\sum_{i=1}^{V} \mathsf{v}_i(\mathsf{k}_\ell)\mathsf{h}_{f_i(\mathsf{k}_\ell)} \prod_{s\in S_\ell - \{f_i(\mathsf{k}_\ell)\}} (Y - \mathsf{h}_s) = \sum_{j=0}^{V-1} \hat{R}_{\ell j} Y^j,$$

$$\hat{I}_\ell(Y) = \prod_{s\in S_\ell}(Y - \mathsf{h}_s) = \sum_{j=0}^{V} \hat{I}_{\ell j} Y^j.$$

Define

$$v_j^{\hat{R}}(X) = \sum_{\ell=1}^{K} \hat{R}_{\ell j}\rho_\ell(X), \qquad v_j^{\hat{I}}(X) = \sum_{\ell=1}^{K} \hat{I}_{\ell j}\rho_\ell(X).$$

Output $\mathcal{W}_{\mathsf{CSS}} = \Big\{\{v_j^{\hat{I}}(X)\}_{j=0}^{V}, \{v_j^{\hat{R}}(X)\}_{j=0}^{V-1}\Big\}$.

Online Phase: Sampling: $\mathcal{V}_{\mathsf{CSS}}$ outputs $x \leftarrow \mathbb{F}$ and $\mathcal{P}_{\mathsf{CSS}}$ computes $D(X) = P(X, x)$.

ProveSampling: $\mathcal{P}_{\mathsf{CSS}}$ finds and outputs $H_k(X)$ such that, if $\hat{R}_x(X) = \sum_{j=0}^{V-1} x^j v_j^{\hat{R}}(X)$, and $\hat{I}_x(X) = \sum_{j=0}^{V} x^j v_j^{\hat{I}}(X)$,

$$D(X)\hat{I}_x(X) = z_H(x)\hat{R}_x(X) + H_k(X)z_K(X).$$

Decision Phase: Accept if and only if (1) $\deg(D) \le K - 1$, and (2) $D(X)\hat{I}_x(X) = z_H(x)\hat{R}_x(X) + H_k(X)z_K(X)$.

Figure 3.5: A CSS scheme for matrices with at most V non-zero elements per column.

For soundness, if the verifier accepts $D(X)$, then $D(\mathsf{k}_\ell)\hat{I}_x(\mathsf{k}_\ell) = z_H(x)\hat{R}_x(\mathsf{k}_\ell)$ and $\hat{I}_x(\mathsf{k}_\ell) = \hat{I}_\ell(x)$, therefore:

$$D(\mathsf{k}_\ell) = \hat{I}_\ell(x)^{-1} z_H(x)\hat{R}_\ell(x) = \big(\prod_{s\in S_\ell^c}(x - \mathsf{h}_s)\big)\hat{R}_x(x) = \sum_{i=1}^{V} \mathsf{v}_i(\mathsf{k}_\ell)\lambda_{\mathsf{f}_i(\mathsf{k}_\ell)}(x).$$

We conclude that $D(X) = P(X, x) \mod z_K(X)$. Since both have degree at most $K - 1$, soundness is proven. □

3.5.3 Sparse Matrices

In this section, we present a CSS argument for matrices $\mathbf{M}$ that are sparse but without any restriction on the non-zero entries per column.

$$\mathcal{M} = \{\mathbf{M} \in \mathbb{F}^{m \times K} \ s.t. \ |\mathbf{M}| \leq K\}.$$

This approach was introduced in Marlin, and is pursued in Lunar and Vampire. For functions v, c and r as defined in the overview at the beginning of this section,

$$P(X, Y) = \sum_{\ell=1}^{K} \mathsf{v}(\mathsf{k}_\ell) \lambda_{\mathsf{r}(\mathsf{k}_\ell)}(Y) \lambda_{\mathsf{c}(\mathsf{k}_\ell)}(X).$$

As explained in the overview, $P(y, x)$ can be written as the inner product of two vectors that depend only on x and y, and the low degree extensions of these vectors, $e_x(X), e_y(X)$, are nothing but the encodings of new matrices $\mathbf{M}_x$ and $\mathbf{M}_y$ in $\mathbb{F}^{m \times K}$ that have at most one non-zero element per column, so the basic CSS of Section 3.5.1 can be used to prove correctness.

Theorem 9. *The argument of Fig. 3.6 satisfies completeness and* $(2K + 1)/|\mathbb{F}|$*-soundness.*

Proof. Completeness follows immediately and thus we only prove soundness. Although it does so in a batched form, the prover is showing that the following equations are satisfied,

$$e_x(X)(x - v_{\mathsf{r}}(X)) = z_H(x) m^{-1} v_{\mathsf{r}}(X) + H_{k,x}(X) z_K(X)$$

$$e_y(X)(y - v_{1,\mathsf{c}}(X)) = z_H(y) v_{2,\mathsf{c}}(X) + H_{k,y}(X) z_K(X)$$

$$K e_x(X) e_y(X) - \sigma = X R_k(X) + z_K(X) H_{k,x,y}(X),$$

Now, since all the left terms of the equations are defined before the verifier sends δ, by the Schwartz-Zippel lemma, with all but probability $3/|\mathbb{F}|$, the verifier accepts if and only such $H_{k,x}(X), H_{k,y}(X), H_{k,x,y}(X), R_k(X)$ exist.

Assuming they do, the rest of the proof is a consequence of (1) soundness of the protocol in Fig. 3.4, which implies that $e_x(X), e_y(X)$ correspond to the correct polynomials modulo $z_K(X)$, and (2) Lemma 3 (see below) shows that if the last equation is satisfied, and $e_x(X), e_y(X)$ coincide with the honest polynomials modulo $z_K(X)$, then $\sigma = P(y, x)$. Because the prover sends $D(X)$sma before receiving y and $D(y) = \sigma$, from the Schwartz-Zippel lemma we have that, except with negligible probability, $P(X, x) = D(X)$ and the argument is sound. □

Lemma 3. *Given $e_x(X)$, $e_y(X)$ such that $e_x(X) = \sum_{\ell=1}^{K} \lambda_{\mathsf{r}(\mathsf{k}_\ell)}(x)\rho_\ell(X)$ and $e_y(X) = \sum_{\ell=1}^{K} \mathsf{v}(\mathsf{k}_\ell)\lambda_{\mathsf{c}(\mathsf{k}_\ell)}(y)\rho_\ell(X)$, $P(y,x) = \sum_{\ell=1}^{K} \mathsf{v}(\mathsf{k}_\ell)\lambda_{\mathsf{c}(\mathsf{k}_\ell)}(y)\lambda_{\mathsf{r}(\mathsf{k}_\ell)}(x) = \sigma$ if and only if there exist polynomials $R_k(X) \in \mathbb{F}_{\leq m-2}[X], H_{k,x,y}(X)$ such that:*

$$e_x(X)e_y(X) - \sigma/K = XR_k(X) + H_{k,x,y}(X)z_K(X).$$

Offline Phase: $\mathcal{I}_{\mathsf{CSS}}$ outputs $\mathcal{W}_{\mathsf{CSS}} = \big(v_{\mathsf{r}}(X), v_{1,\mathsf{c}}(X), v_{2,\mathsf{c}}(X)\big)$, where:

$$v_r(X) = \sum_{\ell=1}^{K} \mathsf{h}_{\mathsf{r}(\mathsf{k}_\ell)}\rho_\ell(X),$$

$$v_{1,\mathsf{c}}(X) = \sum_{\ell=1}^{K} \mathsf{h}_{\mathsf{c}(\mathsf{k}_\ell)}\rho_\ell(X), \qquad v_{2,\mathsf{c}}(X) = m^{-1}\sum_{\ell=1}^{K} \mathsf{v}(\mathsf{k}_\ell)\mathsf{h}_{\mathsf{c}(\mathsf{k}_\ell)}\rho_\ell(X).$$

Online Phase: Sampling: $\mathcal{V}_{\mathsf{CSS}}$ sends $x \leftarrow \mathbb{F}$, and $\mathcal{P}$ outputs $D(X) = P(X,x)$, for $P(X,Y) = \sum_{\ell=1}^{K} \mathsf{v}(\mathsf{k}_\ell)\lambda_{\mathsf{r}(\mathsf{k}_\ell)}(Y)\lambda_{\mathsf{c}(\mathsf{k}_\ell)}(X)$.

ProveSampling: $\mathcal{V}_{\mathsf{CSS}}$ sends $y \leftarrow \mathbb{F}$ and $\mathcal{P}_{\mathsf{CSS}}$ outputs $\sigma = D(y)$ and $e_x(X), e_y(X)$, where $e_x(X) = \sum_{\ell=1}^{K} \lambda_{\mathsf{r}(\mathsf{k}_\ell)}(x)\rho_\ell(X)$, $e_y(X) = \sum_{\ell=1}^{K} \mathsf{v}(\mathsf{k}_\ell)\lambda_{\mathsf{c}(\mathsf{k}_\ell)}(y)\rho_\ell(X)$, $\mathcal{V}_{\mathsf{CSS}}$ sends $\delta \leftarrow \mathbb{F}$ and $\mathcal{P}_{\mathsf{CSS}}$ computes $H_{k,x}(X)$, $H_{k,y}(X)$, $R_k(X)$, $H_{k,x,y}(X)$ such that:

$$e_x(X)(x - v_{\mathsf{r}}(X)) = m^{-1}z_H(x)v_{\mathsf{r}}(X) + H_{k,x}(X)z_K(X)$$

$$e_y(X)(y - v_{1,\mathsf{c}}(X)) = z_H(y)v_{2,\mathsf{c}}(X) + H_{k,y}(X)z_K(X)$$

$$Ke_x(X)e_y(X) - \sigma = XR_k(X) + H_{k,x,y}(X)z_K(X),$$

It also defines $H_k(X) = H_{k,x,y}(X) + \delta H_{k,x}(X) + \delta^2 H_{k,y}(X)$, and outputs $\big(R_k(X), H_k(X)\big)$.

Decision Phase: Accept if and only if (1) $\deg(R_k) \leq K-2$, (2) $D(y) = \sigma$, and (3) for $i_x(X) = (x - v_{\mathsf{r}}(X))$, $i_y(X) = (y - v_{1,\mathsf{c}}(X))$

$$\begin{aligned}&(e_x(X) + \delta^2 i_y(X))(e_y(X) + \delta i_x(X)) - \delta^3 i_x(X)i_y(X)\\ &\quad - \delta^2 z_H(y)v_{2,\mathsf{c}}(X) - \sigma/K - \delta z_H(x)m^{-1}v_{\mathsf{r}}(X) = XR_k(X) + H_k(X)z_K(X).\end{aligned}$$

Figure 3.6: CSS argument for $\mathbf{M}$, with $\mathbb{K}$ such that $|\mathbf{M}| \leq |\mathbb{K}|$.

Proof. Note that $e_x(X)e_y(X) = \sum_{\ell=1}^{K} \mathsf{v}(\mathsf{k}_\ell)\lambda_{\mathsf{c}(\mathsf{k}_\ell)}(y)\lambda_{\mathsf{r}(\mathsf{k}_\ell)}(x)\rho_\ell(X) \mod z_K(X)$. By the univariate sumcheck (Lemma 2), $e_x(X)e_y(X) - \sigma/K$ is divisible by X if and only if $P(y,x) = \sigma$, which concludes the proof. □

Sums of Sparse Matrices

The argument for general sparse matrices of last section can be easily generalized without increasing the communication complexity to any matrix

$$\mathcal{M} = \{\mathbf{M} \in \mathbb{F}^{m\times K} : \mathbf{M} = \sum_{i=1}^{V} \mathbf{M}_i \ s.t. \ \forall\ i \in [V]\ |\mathbf{M}_i| \leq K\}.$$

We consider one function $\mathsf{r} : \mathbb{K} \to [m]$, and, for each i, two functions $\mathsf{c}_i : \mathbb{K} \to [m]$, and $\mathsf{v}_i : \mathbb{K} \to \mathbb{F}$, such that:

$$\lambda(X)^\top \mathbf{M}_i \lambda(Y) = P_i(X,Y) = \sum_{\ell=1}^{K} \mathsf{v}_i(\mathsf{k}_\ell)\lambda_{\mathsf{r}(\mathsf{k}_\ell)}(Y)\lambda_{\mathsf{c}_i(\mathsf{k}_\ell)}(X).$$

Choosing the row and the column function smartly, this can cover many sparse matrices with KV non-zero entries, considerably increasing the expressiveness of the CSS argument. For this generalization, we observe that if $P(X,Y) = \sum_{i=1}^{V} P_i(X,Y)$, then

$$P(y,x) = \big(\lambda_{\mathsf{r}(\mathsf{k}_1)}(x), \ldots, \lambda_{\mathsf{r}(\mathsf{k}_K)}(x)\big) \cdot \sum_{i=1}^{V} \big(\mathsf{v}_i(\mathsf{k}_1)\lambda_{\mathsf{c}_i(\mathsf{k}_1)}(y), \ldots, \mathsf{v}_i(\mathsf{k}_K)\lambda_{\mathsf{c}_i(\mathsf{k}_K)}(y)\big).$$

We can define $e_x(X)$ as Section 3.5.3, and $e_y(X) = \sum_{i=1}^{V}\sum_{\ell=1}^{K} \mathsf{v}_i(\mathsf{k}_\ell)\lambda_{\mathsf{c}_i(\mathsf{k}_\ell)}(y)\rho_\ell(X)$. Thus, $e_y(X) = \lambda(Y)^\top \mathbf{M}_y \rho(X)$, where $\mathbf{M}_y$ is a matrix with at most V non-zero entries in each column. The CSS is constructed as in the one for sparse matrices of Section 3.5.3, except that to prove that $e_y(X)$ is correctly sampled, we use the CSS for sums of basic matrices of Section 3.5.2. Note that this change does not represent an increase in the communication complexity with respect to Section 3.5.3, only in the SRS size.

3.5.4 Linear Combination of Sparse Matrices

Below, we present a CSS argument for the case where a matrix in $\mathbb{F}^{2m\times m}$ is split into two blocks of m rows. This construction corresponds to R1CS-lite, where the encoding of $\mathbf{W}_a$ and $\mathbf{W}_b$ can be opened and checked by the verifier, so we only need to run a CSS argument for $\mathbf{W}_c$. We define K_1, K_2 such that $|\mathbf{F}| \leq K_1, |\mathbf{G}| \leq K_2$ and $K = K_1 + K_2$. Technically, we construct a CSS argument for the matrix $\mathbf{W}_c$ where the coefficients depend on x, δ. As mentioned before, this corresponds to implicitly applying the results in Section 3.5.3 to a matrix $\hat{\mathbf{W}}_\mathbf{c} = \mathbf{W}_c^1 + \delta\mathbf{W}_c^2$ that depends on the verifier's challenge δ. Formally, the set of admissible matrices for the CSS argument of this section is,

$$\mathcal{M} = \{\mathbf{M} \in \mathbb{F}^{m\times m} : \mathbf{M} = \mathbf{M}_1 + \delta\mathbf{M}_2,\ |\mathbf{M}_1| \leq K_1\ \wedge\ |\mathbf{M}_2| \leq K_2 \text{ for given } K_1, K_2\}.$$

Offline Phase: $\mathcal{I}_{\mathsf{CSS}}$ outputs $\mathcal{W}_{\mathsf{CSS}} = \big(v_{\mathsf{r}}(X), v_{1,\mathsf{c}}(X), v^1_{2,\mathsf{c}}(X), v^2_{2,\mathsf{c}}(X)\big)$, where:

$$v_r(X) = \sum_{\ell}^{K} \mathsf{h}_{\mathsf{r}(\mathsf{k}_\ell)}\rho_\ell(X) \qquad v_{1,\mathsf{c}}(X) = \sum_{\ell=1}^{K} \mathsf{h}_{\mathsf{c}(\mathsf{k}_\ell)}\rho_\ell(X).$$

$$v^1_{2,\mathsf{c}}(X) = m^{-1}\sum_{\ell=1}^{K_1} \mathsf{v}(\mathsf{k}_\ell)\mathsf{h}_{\mathsf{c}(\mathsf{k}_\ell)}\rho_\ell(X), \qquad v^2_{2,\mathsf{c}}(X) = m^{-1}\sum_{\ell=K_1+1}^{K} \mathsf{v}(\mathsf{k}_\ell)\mathsf{h}_{\mathsf{c}(\mathsf{k}_\ell)}\rho_\ell(X).$$

Online Phase: Sampling: $\mathcal{V}_{\mathsf{CSS}}$ sends $x, \delta_1 \leftarrow \mathbb{F}$, and $\mathcal{P}$ outputs $D(X) = P(X, x)$, for $P(X, Y) = \sum_{\ell=1}^{K_1} \mathsf{v}(\mathsf{k}_\ell)\lambda_{\mathsf{r}(\mathsf{k}_\ell)}(Y)\lambda_{\mathsf{c}(\mathsf{k}_\ell)}(X) + \delta_1 \sum_{\ell=K_1+1}^{K} \mathsf{v}(\mathsf{k}_\ell)\lambda_{\mathsf{r}(\mathsf{k}_\ell)}(Y)\lambda_{\mathsf{c}(\mathsf{k}_\ell)}(X)$.

ProveSampling: $\mathcal{V}_{\mathsf{CSS}}$ sends $y \leftarrow \mathbb{F}$ and $\mathcal{P}_{\mathsf{CSS}}$ outputs $\sigma = D(y)$ and $e_x(X), e_y(X)$, where $e_x(X) = \sum_{\ell=1}^{K} \lambda_{\mathsf{r}(\mathsf{k}_\ell)}(x)\rho_\ell(X)$, $e_y(X) = \sum_{\ell=1}^{K_1} \mathsf{v}(\mathsf{k}_\ell)\lambda_{\mathsf{c}(\mathsf{k}_\ell)}(y)\rho_\ell(X) + \delta_1 \sum_{\ell=K_1+1}^{K} \mathsf{v}(\mathsf{k}_\ell)\lambda_{\mathsf{c}(\mathsf{k}_\ell)}(y)\rho_\ell(X)$,
$\mathcal{V}_{\mathsf{CSS}}$ sends $\delta_2 \leftarrow \mathbb{F}$ and $\mathcal{P}_{\mathsf{CSS}}$ computes $H_{k,x}(X)$, $H_{k,y}(X)$, $R_k(X)$, $H_{k,x,y}(X)$ such that:

$$e_x(X)(x - v_{\mathsf{r}}(X)) = m^{-1}z_H(x)v_{\mathsf{r}}(X) + H_{k,x}(X)z_K(X)$$
$$e_y(X)(y - v_{1,\mathsf{c}}(X)) = z_H(y)(v^1_{2,\mathsf{c}}(X) + \delta_1 v^1_{2,\mathsf{c}}(X)) + H_{k,y}(X)z_K(X)$$
$$Ke_x(X)e_y(X) - \sigma = XR_k(X) + H_{k,x,y}(X)z_K(X),$$

It also defines $H_k(X) = H_{k,x,y}(X) + \delta_2 H_{k,x}(X) + \delta_2^2 H_{k,y}(X)$, and outputs $\big(R_k(X), H_k(X)\big)$.

Decision Phase: Accept if and only if (1) $\deg(R_k) \leq K - 2$, (2) $D(y) = \sigma$, and (3) for $i_x(X) = (x - v_{\mathsf{r}}(X))$, $i_y(X) = (y - v_{1,\mathsf{c}}(X))$

$$(e_x(X) + \delta_2^2 i_y(X))(e_y(X) + \delta_2 i_x(X)) - \delta_2^3 i_x(X)i_y(X)$$
$$-\delta_2^2 z_H(y)(v^1_{2,\mathsf{c}}(X) + \delta_1 v^2_{2,\mathsf{c}}(X)) - \sigma/K - \delta_2 z_H(x)m^{-1}v_{\mathsf{r}}(X) = XR_k(X) + H_k(X)z_K(X).$$

Figure 3.7: CSS Argument for matrices with at most K non-zero entries.

Let $\mathsf{v} : \mathbb{K} \to \mathbb{F}$ be the function that maps an element $\mathsf{k}_\ell \in \mathbb{K}$ to the value of the ℓth non-zero element of matrix $\mathbf{F}$, if $\ell \leq K_1$, and to the value of the $(\ell - K_1)$th element of $\mathbf{G}$ if $\ell > K_1$. Define also $\mathsf{c}, \mathsf{r} : \mathbb{K} \to [m]$ as the functions that output its

row and column position in the corresponding matrix, we define

$$P_1(X,Y) = \sum_{\ell=1}^{K_1} \mathsf{v}(\mathsf{k}_\ell)\lambda_{\mathsf{r}(\mathsf{k}_\ell)}(Y)\lambda_{\mathsf{c}(\mathsf{k}_\ell)}(X)$$

the sparse encoding of $\mathbf{F}$ and

$$P_2(X,Y) = \sum_{\ell=K_1+1}^{K_2} \mathsf{v}(\mathsf{k}_\ell)\lambda_{\mathsf{r}(\mathsf{k}_\ell)}(Y)\lambda_{\mathsf{c}(\mathsf{k}_\ell)}(X)$$

the sparse encoding of $\mathbf{G}$. Our argument implicitly constructs the sparse encoding of $\mathbf{F} + \delta\mathbf{G}$ as $P(X,Y) = P_1(X,Y) + zP_2(X,Y)$.

As explained in Section 3.5, $P(y,x)$ can be written as the inner product of two vectors that depend only on x and y, and the low degree extensions of these vectors, $e_x(X), e_y(X)$, are nothing but the encodings of new matrices $\mathbf{M}_x$ and $\mathbf{M}_y$ in $\mathbb{F}^{m\times K}$ that have at most one non-zero element per column, so the basic CSS argument of Section 3.5.1 can be used to prove correctness. We present this scheme in Fig. 3.7.

3.5.5 Extension to Low Tensor Rank Matrices

Similar techniques to the ones in Section 3.5 can be used to construct a CSS scheme for matrices that are not sparse but for which a representation of low tensor rank is known. A matrix $\mathbf{M} \in \mathbb{F}^{m\times m}$ has tensor rank r if there exist vectors $\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i \in \mathbb{F}^m$, $i \in [r]$ such that $\mathbf{M} = \sum_{i=1}^r \boldsymbol{\alpha}_i\boldsymbol{\beta}_i^\top$. The main observation is that, in this case, $P(y,x) = \boldsymbol{\lambda}(x)^\top\mathbf{M}\boldsymbol{\lambda}(y) = \sum_i(\boldsymbol{\lambda}(x)^\top\boldsymbol{\alpha}_i)\cdot(\boldsymbol{\beta}_i^\top\boldsymbol{\lambda}(y))$. For each i, we can compute low degree extensions of $(\boldsymbol{\lambda}(x)^\top\boldsymbol{\alpha}_i)$ and $(\boldsymbol{\lambda}(y)\boldsymbol{\beta}_i^\top)$ as before (but taking $\mathbb{K} = \mathbb{H}$), and prove correctness with the basic CSS scheme of Section 3.5.1. Then, we can use the sumcheck theorem to see that $\sigma_{x,i} = \boldsymbol{\lambda}(x)^\top\boldsymbol{\alpha}_i$, and $\sigma_{y,i} = \boldsymbol{\beta}_i^\top\boldsymbol{\lambda}(y)$, and check $P(y,x) = \sum_{i=1}^r \sigma_{x,i}\sigma_{y,i}$. Naturally, the communication complexity depends on the tensor rank.

There is no reason to expect that in practice the tensor rank will be low and, further, in general it is hard to compute. But we think it is of theoretical value to note that sparsity is not always the key for building efficient CSS schemes.

3.5.6 Extended Vandermonde Sampling

The constructions discussed in the previous section impose (once the finite field is fixed) some conditions of the type of admissible matrices considered by the CSS scheme. For many practical use cases, this does not seem to be a limitation. However, regardless of the types of constraints that appear in applications so far, we

Offline Phase: $\mathcal{I}_{\mathsf{CSS}}(\mathbb{F}, \mathbf{M}, J, \ell)$: For all $i \in [Q]$, defines the polynomials $P_i(X) = \sum_{j=1}^{m} m_{ij}\lambda_j(X)$. For $i \in [\ell]$, it defines $P_{Q+i}(X) = \sum_{j=1}^{Q} i^{j-1}P_j(X)$. It outputs $\mathcal{W}_{\mathsf{CSS}} = \{P_1(X), \ldots, P_{Q+\ell}(X)\}$.
Online Phase: $\mathcal{V}_{\mathsf{CSS}}$ samples $x \leftarrow \mathbb{F}$ and a set of J indices $\mathcal{J} \subset [Q+\ell]$. $\mathcal{P}_{\mathsf{CSS}}$ computes and outputs $D(X) = \sum_{i_j \in \mathcal{J}} x^{j-1}P_{i_j}(X)$.

Figure 3.8: CSS argument with verifier sampling

think it is interesting to explore ways of constructing CSS arguments for more general matrices both for future applications and for theoretical understanding.

The most trivial CSS scheme for a matrix $\mathbf{M} \in \mathbb{F}^{Q\times m}$ works as follows: indexer sends Q oracle polynomials, one for each row, as $P_i(X) = \sum_{j=1}^{m} m_{ij}\lambda_j(X)$. The verifier samples $x \leftarrow \mathbb{F}$, and both prover and verifier compute the same polynomial $D(X) = \sum_{i=1}^{Q} x^{i-1}P_i(X)$, the verifier only accepts if the prover sends the same $D(X)$ it computed itself. This "Vandermonde Sampling" of polynomials associated with the row space of $\mathbf{M}$ requires $\mathcal{W}_{\mathsf{CSS}}$ size and prover work to be linear in Q. When using this argument as part of a zkSNARK, the verifier will be linear in the circuit size, which is completely impractical in most scenarios.

In Fig. 3.8, we introduce a simple extension of the "Vandermonde sampling" technique, but trading memory for verifier work. This is impractical if $\mathbf{M}$ is the matrix that encodes the circuit's affine constraints, as $Q \approx m$. However, since this CSS scheme works for any arbitrary $\mathbf{M}$, it is interesting to combine it as explained in Section 3.4.4 with other approaches: for example, this CSS argument can be used to encode a few very dense constraints, and the approach in Section 3.5.3 can be used for the rest.

The argument depends on two parameters J, ℓ: $J = |\mathcal{J}|$ is the number of exponentiations that the verifier does, and ℓ defines the size of the SRS. As we will prove, the argument is Elusive Kernel with probability $\epsilon = \left(\frac{Q}{Q+\ell}\right)^J$. Fixing the soundness error to some λ, one can derive a trade-off between the size of J, ℓ. Taking ℓ as some constant multiple of Q, for having low verifier work, indexer work would be $O(Qm + Q^2)$ and verifier memory $O(Q)$. Again, this only makes sense when Q represents some small set of constraints.

The prover does not need to send the polynomial $D(X)$ as it is computed by the verifier, and in the decision phase the verifier will always accept, so we omit it.

Theorem 10. *The argument of Fig. 3.8 is perfectly complete, perfectly sound and ϵ-Elusive Kernel, for $\epsilon = \frac{J}{|\mathbb{F}|} + \left(\frac{Q}{Q+\ell}\right)^J$.*

Proof. The verifier samples $D(X)$ on its own and thus completeness and soundness follow immediately. On the other hand, the probability that $\mathbf{y}^*$ is not orthogonal to $\mathbf{M}$ but it is orthogonal to $\sum_{i_j \in \mathcal{J}} x^{j-1} P_{i_j}(X)$ can be upper bounded by standard techniques by $\frac{J}{|\mathbb{F}|} + \left(\frac{Q}{Q+\ell}\right)^J$. Indeed, there are two options, a) either it is orthogonal to all the vectors encoded in $\{P_{i_j}(X)\}_{i_j \in \mathcal{J}}$, or b) it is not. The probability of b) is at most $\frac{J}{|\mathbb{F}|}$ by Schwartz-Zippel. For a), note that if $\mathbf{y}^*$ is not orthogonal to $\mathbf{M}$, it can satisfy at most $Q-1$ constraints out of $Q+\ell$. Since the set $\mathcal{J}$ is chosen independently of $\mathbf{y}^*$, the probability that the set $\mathcal{J}$ coincides with constraints $\mathbf{m}_{i_j}$ such that $\mathbf{y} \cdot \mathbf{m}_{i_j} = 0$ is at most:

$$\frac{\binom{Q-1}{J}}{\binom{Q+\ell}{J}} \leq \left(\frac{Q}{Q+\ell}\right)^J.$$

□

3.5.7 Amortized CSS argument

In this section we present a CSS argument that works only in the *amortized* setting as considered in Sonic [MBKM19]. The construction is basically the protocol in the named work, but for a bivariate polynomial in the Lagrange basis rather than Laurent polynomials.

In the amortized setting, the same verifier aims to check the output of different provers $\mathcal{P}_{\mathsf{CSS}}$ in Sampling. The cost of the verification is linear in m and thus the scheme is only recommended when the number of proofs is linear in m as well. The construction is not holographic due to the fact that the verifier needs to read the matrix $\mathbf{M}$ that describes the relation and thus the indexer is trivial. Note that in this case, we consider a single block $\mathbf{M} \in \mathbb{F}^{m \times m}$.

Online Phase: $\mathcal{V}_{\mathsf{CSS}}$ samples $x_s \leftarrow \mathbb{F}$. $\mathcal{P}_{\mathsf{CSS}}$ defines $P(X,Y) = \sum_{i=1}^{m} \lambda_i(Y) P_i(X)$, for $P_i(X) = \sum_{j=1}^{m} m_{ij} \lambda_j(X)$. It outputs $D_s(X) = P(X, x_s)$.

Online Helped Phase: $\mathcal{V}_{\mathsf{CSS}}$ chooses $u_1 \leftarrow \mathbb{F}$. $\mathcal{P}_{\mathsf{CSS}}$ outputs $\tilde{D}(X) = P(u_1, X)$.

Decision Phase: Chooses $u_2 \leftarrow \mathbb{F}$, and calculate $P(u_1, u_2)$. Accept if and only if $\tilde{D}(u_2) = P(u_1, u_2)$ and, for every $\{x_s\}_{s=1}^{t}$, $\tilde{D}(x_s) = D_s(u_1)$.

Figure 3.9: Amortized CSS scheme from [MBKM19].

Still, in the ProveSampling algorithm, the verifier has oracle access to a set $\mathcal{D} = \{D_1(X), \ldots, D_t(X)\}$ of polynomials where each $D_s(X)$ is the output of a different

execution of Sampling with verifier's challenge x_s. Following the original definition, the verifier also has oracle access to the polynomials outputted by $\mathcal{P}_{\mathsf{CSS}}$ (instantiated by what in Sonic is called a helper) in ProveSampling.

3.6 CSS for Specific Relations

In this section we present several instantiations of CSS arguments for different sets of admissible matrices $\mathbf{W} \in \mathbb{F}^{Q \times 3m}$ that represent the relations described in Section 3.2, as opposite to the previous section where we considered its $m \times m$ blocks, that is, square matrices $\mathbf{M} \in \mathbb{F}^{m \times m}$. As before, by using the CSS constructions of this section as a starting point, we can construct linear arguments that can be used as a building block in the PHP of Fig. 3.2 for the corresponding families of weighted R1CS relations. The final goal is to study the efficiency trade-offs that result from the different approaches. We start by describing the general approach and some techniques that allow for better efficiency. We then describe the particular cases separately, presenting a full description of each scheme.

A fundamental observation derived from the description of the relation W-R1CS in Section 3.2 is that the matrix $\mathbf{W}$ that describes it can be seen as a matrix with three blocks $\big(\mathbf{W}_a, \mathbf{W}_b, \mathbf{W}_c\big)$, and sampling in each of these blocks must be done separately, as the prover needs $(D_a(X), D_b(X), D_c(X))$ to do the inner product with $(A(X), B(X), C(X))$, where $C(X) = q_M(X)A(X)B(X) + q_L(X)A(X) + q_R(X)B(X) + q_C(X)$. Naively, the polynomials $D_a(X), D_b(X)$, and $D_c(X)$ are obtained by running *one* CSS scheme for each matrix $\mathbf{W}_a, \mathbf{W}_b$, and $\mathbf{W}_c$, but more careful approaches can save elements in communication complexity.

Before we flesh out the different options of CSS arguments for $\mathbf{W}$ we note some general principles to improve efficiency and possible trade-offs:

a) Each of the column blocks $\mathbf{W}_a, \mathbf{W}_b, \mathbf{W}_c$ is a matrix of Q rows, where $Q = 2m$ or $Q = 3m$. One possibility is to use one CSS argument directly for each one of these matrices of Q rows (assuming they belong to the set of admisible matrices). Another possibility is to cut each $\mathbf{W}_a, \mathbf{W}_b, \mathbf{W}_c$ into blocks of m rows. For instance, when $\mathbf{W}_c = \begin{pmatrix} \mathbf{F} \\ \mathbf{G} \end{pmatrix}$, we can actually use a CSS argument for the matrix $\mathbf{F} + \delta\mathbf{G} \in \mathbb{F}^{m \times m}$, where δ is an element chosen by the verifier. Technically, this is in fact a CSS argument for the matrix $\mathbf{W}_c$ where the sampling coefficients depends also on δ.

b) When a block of size $m \times m$ is trivial, that is, either $\mathbf{0}$ or $\mathbf{I}$, the corresponding $D(X)$ is either zero or can be opened by the verifier. Indeed, when the block is $\mathbf{0}$ so is the resulting polynomial, and when it is $\mathbf{I} \in \mathbb{F}^{m \times m}$, we define $P(X, Y) = \boldsymbol{\lambda}(Y)^\top \mathbf{I} \boldsymbol{\lambda}(X)^\top = \boldsymbol{\lambda}(Y)^\top \boldsymbol{\lambda}(X)$, and the value $P(y, x) = \big(z_H(x)y - xz_H(y)\big)/(x - y)$ can be calculated by the verifier with $O(\log m)$ field operations (a proof of

this can be found, for example, in Lemma 3 of Lunar [CFF+21]). Thus, when $\mathbf{W}_a, \mathbf{W}_b$ consist of trivial blocks of size $m \times m$, as in R1CS-lite, it makes sense to use the approach described in a) and cut these matrices in blocks of m rows. The verifier can then open $D_a(X), D_b(X)$ itself (as they are linear combination of the polynomials corresponding to trivial blocks), so there is no need to use a CSS argument to prove correct sampling.

c) The proofs that $D_a(X), D_b(X), D_c(X)$ are correctly sampled (in case neither of the matrices has a simple form and none of these polynomials can be sampled by the verifier) can be aggregated.

In Section 3.6.1 we introduce a CSS scheme for the case where $\mathbf{W}$ is a matrix of permutations, in Section 3.6.2 an argument for matrices representing circuits with bounded fan-out, and in Section 3.6.3 our most efficient CSS construction, corresponding to a mix of the last two.

3.6.1 Permutation Matrix

As explained in Section 3.2, in order to instantiate $\mathcal{R}_{\mathsf{W\text{-}R1CS}}$ following Plonk, we consider matrices of the form $\mathbf{W} = \mathbf{P} - \mathbf{I}$, where $\mathbf{P}$ is a matrix of permutations in $\mathbb{F}^{3m \times 3m}$. For simplifying notation, we define the mapping $\iota : \{1,2,3\} \to \{a,b,c\}$ as $\iota(1) = a, \iota(2) = b$ and $\iota(3) = c$.

There are several possible ways of proving correct sampling in the rows of $\mathbf{P} - \mathbf{I}$. For instance, we could consider $\mathbf{P}$ as a matrix of three column blocks $\mathbf{P}_a, \mathbf{P}_b$ and $\mathbf{P}_c$, and define the polynomial encoding of each block as $\boldsymbol{\rho}(Y)^\top \mathbf{P}_\gamma \boldsymbol{\lambda}(X)$, where $\boldsymbol{\rho}(X)^\top = (\rho_1(X), \ldots, \rho_{3m}(X))$ are Lagrange interpolation polynomials associated a multiplicative subgroup of size at least $3m$. However, the simplest and most efficient one, splits this matrix into 9 blocks $m \times m$. Since all the blocks of m rows of $\mathbf{I} \in \mathbb{F}^{3m \times 3m}$ are either $\mathbf{0}$ or $\mathbf{I}$, the verifier can open the polynomial associated to $\mathbf{I} \in \mathbb{F}^{3m \times 3m}$ on its own, and a CSS argument is necessary only to sample in the rows of $\mathbf{P}$.

For this approach, parse $\mathbf{P}$ as $(\mathbf{P}_a, \mathbf{P}_b, \mathbf{P}_c)$ and, for $i = 1,2,3$, each $\mathbf{P}_{\iota(i)}$ as three matrices $\mathbb{F}^{m \times m}$ corresponding to blocks of m rows and denoted as $\mathbf{P}^1_{\iota(i)}, \mathbf{P}^2_{\iota(i)}$, and $\mathbf{P}^3_{\iota(i)}$. For $i = 1,2,3$, define the function $\mathsf{r}_i : \mathbb{H} \to [3m]$ that, given an element $\mathsf{h}_\ell \in \mathbb{H}$ outputs the row corresponding to the only non-zero element in column ℓ of matrix $\mathbf{P}_{\iota(i)}$. For $k = 1,2,3$, the polynomial encoding of $\mathbf{P}^k_{\iota(i)}$ is $P^k_{\iota(i)}(X,Y) = \boldsymbol{\lambda}(Y)^\top \mathbf{P}^k_{\iota(i)} \boldsymbol{\lambda}(X) = \sum_{\ell:(k-1)m+1 \le \mathsf{r}_i(\mathsf{h}_\ell) \le km} \lambda_{\mathsf{r}_i(\mathsf{h}_\ell)-(k-1)m}(Y) \lambda_\ell(X)$. The polynomial $D_{\iota(i)}(X)$ is $P_{\iota(i)}(X,x) = P^1_{\iota(i)}(X,x) + zP^2_{\iota(i)}(X,x) + z^2 P^3_{\iota(i)}(X,x)$. The two key elements for efficiency are: 1) the observation that each column block $\mathbf{P}_{\iota(i)}$ is a simple matrix, since it has at most one non-zero element per columnn, and 2) the fact that the proofs for each of these blocks can be batched together.

Offline Phase: $\mathcal{I}_{\mathsf{CSS}}(\mathbb{F}, \mathbf{M})$: For $i = 1, 2, 3$, $k = 1, 2, 3$ define $\mathcal{V}^{i,k} = \{\ell \in [m] : (k-1)m + 1 \leq \mathsf{r}_i(\mathsf{h}_\ell) \leq km\}$, and

$$v^{i,k}(X) = \sum_{\ell \in \mathcal{V}^{i,k}} \mathsf{h}_{\mathsf{r}_i(\mathsf{h}_\ell)-(i-1)m} \lambda_\ell(X).$$

Output $\mathcal{W}_{\mathsf{CSS}} = \left\{\{v^{i,k}(X)\}_{i,k=1}^{3}\right\}$.
Online Phase: Sampling: $\mathcal{V}_{\mathsf{CSS}}$ outputs $x \leftarrow \mathbb{F}$ and $\mathcal{P}_{\mathsf{CSS}}$ computes and outputs $D_{\iota(i)}(X) = P_{\iota(i)}(X, x)$ for $i = 1, 2, 3$.
ProveSampling: $\mathcal{V}_{\mathsf{CSS}}$ outputs δ. For $i = 1, 2, 3$, $v^i(X) = \sum_{k=1}^{3} v^{i,k}(X)$ and $v_z^i(X) = \sum_{k=1}^{3} z^{k-1} v^{i,k}(X)$, the prover $\mathcal{P}_{\mathsf{CSS}}$ finds and outputs $H(X)$ such that

$$\begin{aligned} D_a(X)(x - v^1(X)) + \delta D_b(X)(x - v^2(X)) + \delta^2 D_c(X)(x - v^3(X)) = \\ z_H(x)(v_z^1(X) + \delta v_z^2(X) + \delta^2 v_z^3(X)) + H(X) z_H(X). \end{aligned}$$

Decision Phase: Accept if and only if (1) $\deg(D_{\iota(i)}) \leq m - 1$, for $i = 1, 2, 3$ and (2)

$$\begin{aligned} D_a(X)(x - v^1(X)) + \delta D_b(X)(x - v^2(X)) + \delta^2 D_c(X)(x - v^3(X)) \\ = z_H(x)(v_z^1(X) + \delta v_z^2(X) + \delta^2 v_z^3(X)) + H(X) z_H(X) \end{aligned}$$

Figure 3.10: CSS Argument for $\mathbf{P} \in \mathbb{F}^{3m \times 3m}$.

We propose our construction in Fig. 3.10. The approach is less efficient in terms of proof size than PLONK, but we think the additional flexibility of the CSS argument is a plus. We argue that combining this approach with the bounded fan-out approach presented next, the SRS size does not need to depend on the total number of gates (additive plus multiplicative), as it will be discussed.

3.6.2 Bounded Fan-out

Circuits with fan-out bounded by some constant V can naturally be encoded as an instance of $\mathcal{R}_{\text{W-R1CS}}$ for the set $\mathcal{M}$ of matrices $\mathbf{W} = \begin{pmatrix} \mathbf{I} & \mathbf{0} & -\mathbf{F} \\ \mathbf{0} & \mathbf{I} & -\mathbf{G} \end{pmatrix}$ with $\mathbf{F}, \mathbf{G} \in \mathbb{F}^{m \times m}$ such that there are at most V non-zero elements per column in each. As we shall see in Fig. 3.11, there exists a very efficient proof system for this relation, since the structure of the matrices allows to use basic CSS arguments that cannot be used in the general case.

For circuits with bounded fan-out, we can set $l = l_b$, $\boldsymbol{q}_M = \mathbf{1}$, $\boldsymbol{q}_L = \boldsymbol{q}_R = \boldsymbol{q}_C = \mathbf{0}$. This choice also gives very short specific SRS, since these vectors do not need to be computed by the indexer. We present the rolled out zkSNARK for such matrices in Section 3.7.3.

However, we note that a more flexible choice of these values can be helpful to encode general circuits. Indeed, any circuit can be transformed to a circuit with bounded fan-out by artificially augmenting the vector $\boldsymbol{c}$ and adding constraints of the form $c_i = c_j$ that ensure consistency. To express satisfiability of this system as an instance of $\mathcal{R}_{\mathsf{W\text{-}R1CS}}$, the additional constraints $c_i = c_j$ can rewritten as an equation involving left (or right) wires, i.e. $a_i = c_j$, that is encoded in the matrix $\mathbf{W}$, and a gate, i.e. $a_i = c_i$ (setting $(q_M)_i = (q_R)_i = (q_C)_i = 0$, $(q_L)_i = 1$). If the fan-out of a certain gate is κ, this requires extending $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ by approximately $3\kappa/V$ dummy variables, and include $\boldsymbol{q}_M$ and $\boldsymbol{q}_L$ in the SRS (the rest are trivial). The construction of Section 3.7.3 can be easily modified for that case and we omit further details.

Finally, we consider the case of circuits with bounded fan-out, that is, the case where the circuit can be represented with a matrix $\mathbf{W} = \begin{pmatrix} \mathbf{I} & \mathbf{0} & -\mathbf{F} \\ \mathbf{0} & \mathbf{I} & -\mathbf{G} \end{pmatrix}$ that is a sum of at most V simple matrices, i.e. it has at most V non-zero elements per column.

As before, and since the other blocks of m rows are the identity matrix or the zero matrix, it suffices to use a CSS argument to sample in the image of $\mathbf{W}_c = -\begin{pmatrix} \mathbf{F} \\ \mathbf{G} \end{pmatrix}$. For that, we first write the matrix $\mathbf{W}_c = \sum_{i=1}^{V} \begin{pmatrix} \mathbf{F}_i \\ \mathbf{G}_i \end{pmatrix}$, where each $\begin{pmatrix} \mathbf{F}_i \\ \mathbf{G}_i \end{pmatrix}$ is a simple matrix. Once more, we will implicitly construct the scheme for $\hat{\mathbf{W}} = \mathbf{F} + \delta\mathbf{G}$, that can be written as $\sum_{i=1}^{V} \mathbf{F}_i + \delta\mathbf{G}_i$, with each $\mathbf{F}_i + \delta\mathbf{G}_i$ having at most one non-zero element in each column. We define two functions associated to each $\mathbf{W}_{c,i} = \begin{pmatrix} \mathbf{F}_i \\ \mathbf{G}_i \end{pmatrix}$. The function $\mathsf{r}_i : \mathbb{H} \to [2m]$ that, given an element $\mathsf{h}_\ell \in \mathbb{H}$ outputs the row corresponding to the only non-zero element in column ℓ of matrix $\mathbf{W}_{c,i}$ and the function $\mathsf{v}_i : \mathbb{H} \to \mathbb{F}$ that outputs the value of this non-zero entry. The details of the scheme are given in Fig. 3.11 and for simplicity in the notation, we define the sets $\mathcal{V}_\ell^1 = \{i \in [V] : 1 \leq \mathsf{r}_i(\mathsf{h}_\ell) \leq m\}$, $\mathcal{V}_\ell^2 = \{i \in [V] : m+1 \leq \mathsf{r}_i(\mathsf{h}_\ell) \leq 2m\}$, $S_\ell = \{\{\mathsf{r}_i(\mathsf{h}_\ell) : i \in \mathcal{V}_\ell^1\} \cup \{\mathsf{r}_i(\mathsf{h}_\ell) - m : i \in \mathcal{V}_\ell^2\}\}$ and $\hat{\mathcal{V}}_i^1 = \{\ell \in [m] : 1 \leq \mathsf{r}_i(\mathsf{h}_\ell) \leq m\}$, $\hat{\mathcal{V}}_i^2 = \{\ell \in [m] : m+1 \leq \mathsf{r}_i(\mathsf{h}_\ell) \leq 2m\}$.

3.6.3 Mixing the Bounded Fan-out and the Permutation Approach.

Bayer and Groth [BG12] introduce techniques to prove that a vector is a permutation of another one. This approach is useful for many applications, but for the ones discussed in this section it has the drawback that it is not easy to extend it to sums of permutations without increasing the communication complexity. This is exactly the issue in the fully succinct mode of Sonic, where complexity grows with the number of permutation matrices into which the constraint matrix can be decomposed.

Offline Phase: $\mathcal{I}_{\mathsf{CSS}}\big(\mathbb{F}, \mathbf{M}\big)$: Define the polynomials $\hat{R}^1_\ell(Y), \hat{R}^2_\ell(Y), \hat{I}_\ell(Y)$, and its coefficients $\hat{R}^1_{\ell j}, \hat{R}^2_{\ell j}, \hat{I}_{\ell j}$:

$$\hat{R}^1_\ell(Y) = \frac{1}{m} \sum_{i \in \mathcal{V}^1_\ell} \mathsf{v}_i(\mathsf{h}_\ell) \mathsf{h}_{\mathsf{r}_i(\mathsf{h}_\ell)} \prod_{s \in S_\ell - \{\mathsf{r}_i(\mathsf{h}_\ell)\}} (Y - \mathsf{h}_s) = \sum_{j=0}^{V-1} \hat{R}^1_{\ell j} Y^j,$$

$$\hat{R}^2_\ell(Y) = \frac{1}{m} \sum_{i \in \mathcal{V}^2_\ell} \mathsf{v}_i(\mathsf{h}_\ell) \mathsf{h}_{\mathsf{r}_i(\mathsf{h}_\ell)-m} \prod_{s \in S_\ell - \{\mathsf{r}_i(\mathsf{h}_\ell)-m\}} (Y - \mathsf{h}_s) = \sum_{j=0}^{V-1} \hat{R}^2_{\ell j} Y^j,$$

$$\hat{I}_\ell(Y) = \prod_{s \in S_\ell} (Y - \mathsf{h}_s) = \sum_{j=0}^{V} \hat{I}_{\ell j} Y^j.$$

Define

$$v^{\hat{R},1}_j(X) = \sum_{\ell=1}^{m} \hat{R}^1_{\ell j} \lambda_\ell(X), \qquad v^{\hat{R},2}_j(X) = \sum_{\ell=1}^{m} \hat{R}^2_{\ell j} \lambda_\ell(X).$$

$$v^{\hat{I}}_j(X) = \sum_{\ell=1}^{m} \hat{I}_{\ell j} \lambda_\ell(X).$$

Output $\mathcal{W}_{\mathsf{CSS}} = \Big\{ \{v^{\hat{I}}_j(X)\}_{j=0}^{V}, \{v^{\hat{R},1}_j(X), v^{\hat{R},2}_j(X)\}_{j=0}^{V-1} \Big\}$.

Online Phase: Sampling: $\mathcal{V}_{\mathsf{CSS}}$ outputs $x, \delta \leftarrow \mathbb{F}$ and $\mathcal{P}_{\mathsf{CSS}}$ computes $D(X) = P(X, x)$, for $P(X, Y) = \sum_{i=1}^{V} \left(\sum_{\ell \in \hat{\mathcal{V}}^1_i} \mathsf{v}_i(\mathsf{h}_\ell) \lambda_{\mathsf{r}_i(\mathsf{h}_\ell)}(Y) \lambda_\ell(X) \right) + \delta \left(\sum_{\ell \in \hat{\mathcal{V}}^2_i} \mathsf{v}_i(\mathsf{h}_\ell) \lambda_{\mathsf{r}_i(\mathsf{h}_\ell)-m}(Y) \lambda_\ell(X) \right)$.

ProveSampling: $\mathcal{P}_{\mathsf{CSS}}$ finds and outputs $H(X)$ such that, if $\hat{R}_x(X) = \sum_{j=0}^{V-1} x^j \big(v^{\hat{R},1}_j(X) + \delta v^{\hat{R},2}_j(X) \big)$, and $\hat{I}_x(X) = \sum_{j=0}^{V} x^j v^{\hat{I}}_j(X)$,

$$D(X)\hat{I}_x(X) = z_H(x)\hat{R}_x(X) + H(X)z_H(X).$$

Decision Phase: Accept if and only if (1) $\deg(D) \leq m - 1$, and (2) $D(X)\hat{I}_x(X) = z_H(x)\hat{R}_x(X) + H(X)z_H(X)$.

Figure 3.11: CSS Argument for a matrix $\mathbf{W}_c$ with two blocks $\mathbf{F}, \mathbf{G}$ where $\mathbf{F}, \mathbf{G}$ have at most V non-zero elements per column.

To counter this issue, Plonk proposes to define the permutation $\mathbf{P} \in \mathbb{F}^{3m \times 3m}$. As mentioned before, the idea is to create a vector of copy constraints. With this approach, the fan-out can be unlimited. Values that are repeated are encoded as a cycle of the permutation and the price to pay is that additive gates are no longer for free.

It is worth investigating if these ideas can be mixed. Namely, since in our case we can increase the fan-out to V without paying in terms of proof size, one could follow the copy constraint approach only for the wires exceeding the fan-out bound. The result would be that additive gates involving only output wires that are input of less than V multiplication gates would be for free.

3.7 zkSNARKs from CSS arguments

In this Section, we focus on the most practical side of the contributions presented so far. In Section 3.7.1 we will discuss on the result of applying the compiler in [CFF+21] to the PHPs for W-R1CS that come out when instantiating the protocol in Fig. 3.2 with the CSS schemes of Section 3.6. Then, in Section 3.7.2 we show a practical result that saves several proof elements as well as extra pairings in our constructions. Finally, we rolled-out Basilisk, our most efficient zkSNARK in Section 3.7.3.

3.7.1 Compiler

The universal SRS of the zkSNARK will be $\mathsf{srs_u} = \left(\{[\tau^i]_1\}_{i=1}^{d}, [\tau]_2\right)$, where d is the maximum degree among all polynomials in $\mathcal{W}_{\mathsf{CSS}}$ or sent by the prover. $\mathsf{srs_W}$ consists of the evaluation in τ of the polynomials that $\mathcal{I}_{\mathsf{LA}}$ outputs, thus, $|\mathsf{srs_W}| = |\mathcal{W}_{\mathsf{CSS}}|+4$, due to polynomials $q_L(X), q_r(X), q_M(X)$ and $q_c(X)$. Still, in all schemes but the one of Fig. 3.10 these polynomials are zero and then the size of $\mathsf{srs_W}$ is the size of $\mathcal{W}_{\mathsf{CSS}}$.

Prover and Verifier instantiate $\mathcal{P}_{\mathsf{W\text{-}R1CS}}$ and $\mathcal{V}_{\mathsf{W\text{-}R1CS}}$ for the PHP of Fig. 3.2 that achieves zero-knowledge through the changes presented in Fig. 3.3, as presented below.

All oracle polynomials sent by $\mathcal{P}_{\mathsf{W\text{-}R1CS}}$ are translated into polynomials evaluated (in the source group) at τ. For degree checks with $deg(p) < \mathsf{dg}$, $\mathsf{dg} < d$, the prover sends a single extra polynomial and field element(see Definition 3), while checks for $\mathsf{dg} = d$ are for free.

For each polynomial equation, prover sends extra field elements corresponding to evaluations (or openings) of some of the polynomials involved on it (maximum one per quadratic term, due to the procedure stated in [GWC19] attributed to M. Maller). There are several ways to do this compilation check, but to optimize efficiency the choices are quite standard (for instance, only $A'(X)$ or $B'(X)$, should be opened).

All the openings at one point, as well as the degrees of the opened polynomials, can be proven with one group element and verified with two pairings, which sets

proof size (in terms of group elements) as the amount of oracles sent by $\mathcal{P}_{\text{W-R1CS}}$ plus one element for degree check of $R_t(X)$ in the linear argument and one for each polynomial equation. The number of field elements sent by the prover changes depending on the amount of terms included in the final polynomial equation as explained above, but always include one element for each polynomial commitment opening.

Prover's work includes running $\mathcal{P}_{\text{W-R1CS}}$ as well as the computation of the polynomial commitment opening procedures. Verifier work is also $\mathcal{V}_{\text{W-R1CS}}$ plus the (batched) verification procedure of the polynomial commitments. The vector of queries is $(\mathsf{b}_A, \mathsf{b}_B, \mathsf{b}_{R_t}, \mathsf{b}_{H_t}) = (1, 0, 1, 0)$.

On the other hand, we write the matrix $\mathbf{W}$ that expresses the constraints as:

$$\mathbf{W} = \begin{pmatrix} \mathbf{I}_m & \mathbf{0}_{m\times n} & \mathbf{0}_{m\times m} & \mathbf{0}_{m\times n} & -\mathbf{F} & \mathbf{0}_{m\times n} \\ \mathbf{0}_{m\times m} & \mathbf{0}_{m\times n} & \mathbf{I}_m & \mathbf{0}_{m\times n} & -\mathbf{G} & \mathbf{0}_{m\times n} \\ \mathbf{0}_m^\top & \mathbf{1}_n^\top & \mathbf{0}_m^\top & \mathbf{1}_n^\top & \mathbf{0}_{m\times m}^\top & \mathbf{0}_{m\times n}^\top \end{pmatrix} = \begin{pmatrix} \mathbf{I}' & \mathbf{0} & \mathbf{F}' \\ \mathbf{0} & \mathbf{I}' & \mathbf{G}' \\ \mathbf{w} & \mathbf{w} & \mathbf{0} \end{pmatrix},$$

where $\mathbf{I}', \mathbf{F}', \mathbf{G}'$ are of size $m \times (m+n)$, $\mathbf{w}$ is a row vector of length $m+n$.

Our PHP is built generically for any CSS argument, but concrete efficiency depends on the specifics of it and also how the blocks of rows of $\mathbf{W}$ are combined. The last constraint will always be treated separately (to exploit the symmetry of the other blocks), and because of its simple form, the verifier can compute the corresponding $\mathbf{D}(X) = (\sum_{i=m+1}^{m+n} \lambda_i(x), \sum_{i=m+1}^{m+n} \lambda_i(x), 0)$ itself, and combine it with the rest (see Section 3.4.4).

For the sparse matrix construction of Fig. 3.6, we assume that $K \geq 2m$, which sets $d = K-1$. This eliminates the degree checks for $e_x(X), e_y(X), R_k(X)$. Assuming $K \geq |\mathbf{F}| + |\mathbf{G}|$, the indexer is run for a matrix $\mathbf{F} + Z\mathbf{G}$, where Z is a variable and thus outputs one polynomial $v_r(X)$, one polynomial $v_{1,c}(X)$ but two polynomials $v_{2,c}^F(X), v_{2,c}^G(X)$ that will let the verifier construct $v_{2,c}(X) = v_{2,c}^F(X) + \delta v_{2,c}^G(X)$ after choosing δ, as shown in Fig.3.7. This set the size of the universal srs to $K-1$ and the size of $\mathsf{srs}_{\mathbf{W}}$ to 4. Prover sends 11 polynomials, 8 of degree up to $m-1$ and the rest $(e_x(X), e_y(X), R_k(X), H_k(X))$ of degree $K-1$. Verifier performs two pairings and the field operations to compute $D_a(y), D_b(y), z_l(y), z_K(y)$, and $z_H(y)$.

The zkSNARK that uses the CSS argument of Fig. 3.10 has a universal SRS of size $n+5$, for n the *total* number of gates of the circuit, i.e., multiplicative and additive gates as well, while the relation dependent has 11 group elements. The proof has 9 group elements (this time the prover has to send $D_a(X)$ and $D_b(X)$) and 5 field elements (as it also sends its evaluations on challenge y). Prover work depends only on these polynomials and consists of $9m$ group operations. Similar to other constructions, verifier work includes 2 pairings and $O(l + \log m)$ field operations.

Finally, the zkSNARK that builds on the CSS scheme of Fig. 3.11 (*Basilisk*) is given in Fig. 3.12, 3.13: the universal SRS has size $m+6$, for m the number

of multiplicative gates of the circuit, and the one describing the relation includes $3V + 1$ group elements. The proof includes 7 group and 3 field elements. Prover work is dominated by the generation of these polynomials, and consists of $7m$ group operations. Verifier performs $O(l + \log m)$ operations to compute $\sum_{i=1}^{l} x_i \lambda_i(y)$ from the public input $\mathbf{x}$ and challenge y and to evaluate $z_H(x), z_H(y), z_l(y), D_a, D_b$ from its challenges x and y. It also does two pairings to check the final equation. As explained in Section 3.4.3, to generalize the construction to arbitrary circuits we can add dummy variables (the exact number depends on the number of gates that exceed the fan-out bound). The SRS will grow accordingly, and the derived SRS needs to include two additional polynomials.

If the extended Vandermonde technique of (Fig. 3.8) is used for some set of Q rows, we set $d = 2m - 1$. $\mathsf{srs}_{\mathbf{W}}$ outputs $Q + \ell$ vectors of three polynomials. Prover only sends commitments to $A'(X)$ and $B'(X)$ and the polynomials $R(X), H(X)$ of the linear argument (Fig.3.1). Verifier checks degree of $R(X)$ and one polynomial equation of three terms, two of which include polynomials it can evaluate itself (X and $z_H(X)$).

3.7.2 Eliminating Non-Trivial Degree Checks

As explained in Apendix 3.7, checking that a polynomial is of degree at most $m-1$ is for free, as the srs includes only powers of τ up to this bound. On the other hand, checks for smaller degrees require the prover to send two extra elements, one in the field and one in the group. In all our constructions such degree check is required for the linear argument, since Theorem 2 states that the degree of polynomial $R(X)$ has to be at most $m - 2$. Also, in the CSS of Fig. 3.6, it must be the case that $\deg(R_u) \leq m - 2$.

Below, we present a simple corollary of Theorem 2 to augment the degree of $R(X)$ by 1. This trick allows to save the aforementioned elements in the proof.

Corollary 1. *Let $k, m, \mathbf{y}, \mathbf{d}, \mathbb{F}, \mathbb{H}$ be as in Theorem 2 and let $u \in \mathbb{F}^*$, $u \notin \mathbb{H}$. Then, $\mathbf{y} \cdot \mathbf{d} = \sigma$ if and only if there exist $H(X), R(X) \in \mathbb{F}[X]$, $R(X)$ of degree at most $m - 1$, such that the following relation holds:*

$$\mathbf{Y}(X) \cdot \mathbf{D}(X)(X - u) - \frac{\sigma}{m}(X - u) = XR(X) + z_H(X)H(X)(X - u), \qquad (3.4)$$

where $\mathbf{Y}(X) = (Y_1(X), \ldots, Y_k(X))$ is a vector of polynomials of arbitrary degree such that $Y_i(\mathsf{h}_j) = y_{ij}$ for all $i = 1, \ldots, k$, $j = 1, \ldots, m$, and $\mathbf{D}(X) = (D_1(X), \ldots, D_k(X))$ is such that $D_i(X) = \mathbf{d}_i^\top \lambda(X)$.

Proof. By Theorem 2, $\mathbf{y} \cdot \mathbf{d} = \sigma$ if and only if there exists some $R'(X)$ of degree at most $m - 2$ such that $\mathbf{Y}(X) \cdot \mathbf{D}(X) - \frac{\sigma}{m} = XR'(X) + z_H(X)H(X)$.

If $\mathbf{y} \cdot \mathbf{d} = \sigma$ then such $R'(X)$ exists and the polynomial $R(X) = (X-u)R'(X)$ is of degree at most $m-1$ and satisfies Eq. (3.4).

We now prove the reciprocal. Suppose there exists $R(X)$ of degree at most $m-1$ such that Eq. (3.4) holds. Since all the sum terms in the equation, except $XR(X)$, are divisible by $X-u$, and $u \neq 0$, then $(X-u)$ divides $R(X)$. Define $R'(X) = R(X)/(X-u)$. Dividing Eq. (3.4) by $X-u$, it follows that $R'(X)$ satisfies Eq. (3.2) and is of degree at most $m-2$, so by Theorem 2 it follows that $\mathbf{y}\cdot\mathbf{d} = \sigma$. □

3.7.3 Rolled-out zkSNARK for Circuits with Bounded Fan-Out

Below we present the most efficient zkSNARK we can achieve from the presented CSS schemes. It works for relations $\mathsf{R} \in \mathcal{R}_{\mathsf{W\text{-}R1CS}}$ that represent a circuit with bounded fan-out[6]. We have $\mathbf{W} = (\mathbf{W}_a, \mathbf{W}_b, \mathbf{W}_c)$, where $\mathbf{W}_c$ has at most V non-zero elements per column. We present the scheme for the case where $l = l_b$, $\mathbf{q}_M = \mathbf{1}$, $\mathbf{q}_L = \mathbf{q}_R = \mathbf{q}_C = \mathbf{0}$ but it is straightforward to modify the argument for other values. In blue we highlight the modifications to the PHP of Fig. 3.2 in order to make it zero knowledge. The functions $\{\mathsf{v}_i, \mathsf{r}_i\}_{i=1}^{V}$, $P(X,Y)$, and the sets $\{S_\ell, \mathcal{V}_\ell^1, \mathcal{V}_\ell^2\}_{\ell=1}^{m}$ are defined as above. If $\iota(1) = a, \iota(2) = b, \iota(3) = c$, and for $i = 1,2$, $k = 1,2$, let $(P')^k_{\iota(i)}(X,Y) = \lambda(Y)^\top(\mathbf{W}^1_{\iota(i)} + \delta_1 \mathbf{W}^2_{\iota(i)})\lambda(X)$, and $(D')^k_{\iota(i)}(X) = (P')^k_{\iota(i)}(X,x)$.

$\mathsf{KeyGen}(\mathcal{R})$: Sample $\tau \leftarrow \mathbb{F}$ and output $\tau, \mathsf{srs_u} = \left(\{[\tau^i]_1\}_{i=0}^{m-1}, \{[\tau^i]_1\}_{i=m}^{m+5}, [\tau]_2\right)$. Choose an arbitrary $u \in \mathbb{F}^*$, $u \notin \mathbb{H}$.
$\mathsf{KeyGenD}(\mathsf{srs_u}, \mathbf{W}', \mathbf{w}')$: Parse $\mathbf{W}' = \left(\mathbf{W}'_a, \mathbf{W}'_b, \mathbf{W}'_c\right)$ and $\mathbf{W}'_c$ as $\mathbf{W}'_c = \begin{pmatrix} \mathbf{F} & \mathbf{0}_{m\times 6} \\ \mathbf{G} & \mathbf{0}_{m\times 6} \end{pmatrix}$, $\mathbf{F}, \mathbf{G} \in \mathbb{F}^{m\times m}$. For $i \in [V], k = 1,2$ define $\hat{R}^k_\ell(Y)$, and its coefficients $\hat{R}^k_{\ell j}$ as:

$$\hat{R}^k_\ell(Y) = \frac{1}{m} \sum_{i \in \mathcal{V}^k_\ell} \mathsf{v}_i(\mathsf{h}_\ell) \mathsf{h}_{\mathsf{r}_i(\mathsf{h}_\ell)-(k-1)m} \prod_{s \in S_\ell - \{\mathsf{r}_i(\mathsf{h}_\ell)-(k-1)m\}} (Y - \mathsf{h}_s) = \sum_{j=0}^{V-1} \hat{R}^k_{\ell j} Y^j,$$

Also, let $\hat{I}_\ell(Y)$ and $\hat{I}_{\ell j}$ be such that $\hat{I}_\ell(Y) = \prod_{s\in S_\ell}(Y - \mathsf{h}_s) = \sum_{j=0}^{V} \hat{I}_{\ell j} Y^j$.
Finally, for $j = 0, \ldots, V-1$ define $v_j^{\hat{R},1}(X) = \sum_{\ell=1}^m \hat{R}^1_{\ell j}\lambda_\ell(X)$, $v_j^{\hat{R},2}(X) = \sum_{\ell=1}^m \hat{R}^2_{\ell j}\lambda_\ell(X)$, and, for $j = 0, \ldots, V$ $v_j^{\hat{I}}(X) = \sum_{\ell=1}^m \hat{I}_{\ell j}\lambda_\ell(X)$. Compute $[v_j^{\hat{I}}]_1 = [v_j^{\hat{I}}(\tau)]_1, [v_j^{\hat{R},1}]_1 = [v_j^{\hat{R},1}(\tau)]_1, [v_j^{\hat{R},2}]_1 = [v_j^{\hat{R},2}(\tau)]_1$.
Output $\mathsf{srs_W} = \left(\mathsf{srs_u}, \left\{[v_j^{\hat{I}}]_1\right\}_{j=0}^{V}, \left\{[v_j^{\hat{R},1}]_1, [v_j^{\hat{R},2}]_1\right\}_{j=0}^{V-1}\right)$.

Figure 3.12: Basilisk's KeyGen and KeyGenD algorithms.

[6]Circuits can be transformed into this form by adding additional dummy constraints.

$\mathsf{Prove}(\mathbf{W}, \mathsf{srs}_{\mathbf{W}}, (\mathbf{x}, (\mathbf{a}', \mathbf{b}')))$: Sample $\mathbf{r}_a \leftarrow \mathbb{F}^4, \mathbf{r}_b \leftarrow \mathbb{F}^2$ and define $\mathbf{a} = (\mathbf{x}, \mathbf{a}', \mathbf{r}_a, \mathbf{1})$, $\mathbf{b} = (\mathbf{1}, \mathbf{b}', \mathbf{1}, \mathbf{r}_b)$. Then compute $A(X) = \sum_{j=1}^{m+6} a_j \lambda_j(X)$, $B(X) = \sum_{j=1}^{m+6} b_j \lambda_j(X)$, $B'(X) = \big(B(X) - 1\big)/\big(t_l(X)\prod_{i=1}^{4}(X - \mathsf{h}_{m+i})\big)$, and

$$A'(X) := \left(\left(\sum_{j=l+1}^{m+6} a_j \lambda_j(X)\right) - t_l(X)\right) / \left(t_l(X)(X - \mathsf{h}_{m+5})(X - \mathsf{h}_{m+6})\right).$$

Output $\pi_1 = \big([A']_1 = [A'(\tau)]_1, [B']_1 = [B'(\tau)]_1\big)$.

$\mathsf{Verify}(\mathsf{srs}_{\mathbf{W}}, \mathbf{x}, \pi_1)$: Send $x, \delta_1, \delta_2 \leftarrow \mathbb{F}$.

$\mathsf{Prove}(\mathbf{W}, \mathsf{srs}_{\mathbf{W}}, (\mathbf{x}, (\mathbf{a}', \mathbf{b}')), x, \delta_1, \delta_2)$: For each $i = 1, 2, 3$, define $D'_{\iota(i)}(X) = (D')^1_{\iota(i)}(X) + \delta_1 (D')^2_{\iota(i)}(X)$. Let $D_a(X) = D'_a(X) + \delta_1^2 \sum_{j=m+1}^{m+6} \lambda_j(X)$, $D_b(X) = D'_b(X) + \delta_1^2 \sum_{j=m+1}^{m+6} \lambda_j(X)$ and $D_c(X) = D'_c(X)$.

Find $R(X), H_1(X), H_2(X)$ such that:

$$\begin{aligned} A(X)D_a(X)(X-u) + B(X)D_b(X)(X-u) - D_c(X)A(X)B(X)(X-u) \\ = XR(X) + z_H(X)H_1(X)(X-u) \end{aligned}$$

and, if $\hat{R}_x(X) = \sum_{j=0}^{V-1} x^j \big(v_j^{\hat{R},1}(X) + \delta_1 v_j^{\hat{R},2}(X)\big)$ and $\hat{I}_x(X) = \sum_{j=0}^{V} x^j v_j^{\hat{I}}(X)$,

$$D_c(X)\hat{I}_x(X) = z_H(x)\hat{R}_x(X) + H_2(X)z_H(X).$$

Output $\pi_2 = \big([D_c]_1 = [D_c(\tau)]_1,\ [H]_1 = [H_1(\tau)]_1 + \delta_2[H_2(\tau)]_1, [R]_1 = [R(\tau)]_1$.

$\mathsf{Verify}(\mathsf{srs}_{\mathbf{W}}, \mathbf{x}, \pi_1, \pi_2)$: Send $y, \gamma \leftarrow \mathbb{F}$.

$\mathsf{Prove}(\mathbf{W}, \mathsf{srs}_{\mathbf{W}}, (\mathbf{x}, (\mathbf{a}', \mathbf{b}')), x, \delta_1, \delta_2, y, \gamma)$: Define $\sigma = D_c(y)$ and, for

$$\begin{aligned} E(X) = A(y)D_a(y)(y-u) + B(X)D_b(y)(y-u) + \sigma\big(-A(y)B(X)(y-u) + \delta_2\hat{I}_x(X)\big) \\ -yR(X) - \delta_2 z_H(x)\hat{R}_x(X) - z_H(y)H(X)(y-u), \end{aligned}$$

vector of polynomials $\mathbf{p}(X) = (A(X), D_c(X), E(X))$, and degree bounds set by $\mathbf{d} = (m-1, m-1, m-1)$, calculate $([w]_1, (a, \sigma, 0)) \leftarrow \mathsf{PC.Open}\,(\mathsf{srs}_{\mathsf{u}}, \mathbf{p}(X), \mathbf{d}, y, \gamma)$ and output $\pi_3 = ([w]_1, (a, \sigma))$.

$\mathsf{Verify}(\mathsf{srs}_{\mathbf{W}}, \mathbf{x}, \pi_1, \pi_2, \pi_3)$: Define $s = a + \gamma\sigma$ and compute constants $D_a = D_b = \big(z_H(x)y - xz_H(y)\big)/(x-y) - \sum_{j=m+1}^{m+6} \lambda_j(x)\lambda_j(y)$. Also, set

$$[A]_1 = \big([A']_1(y - \mathsf{h}_{m+5})(y - \mathsf{h}_{m+6}) + 1\big)t_l(y) + \sum_{i=1}^{l} x_i \lambda_i(y), \qquad [\hat{I}_x]_1 = \sum_{j=0}^{V} x^j [v_j^{\hat{I}}]_1,$$

$$[B]_1 = \Big([B']_1 t_l(y)\prod_{i=1}^{4}(y - \mathsf{h}_{m+i}) + 1\Big), \qquad [\hat{R}_x]_1 = \sum_{j=0}^{V-1} x^j \big([v_j^{\hat{R},1}]_1 + \delta_1 [v_j^{\hat{R},2}]_1\big) \text{ and}$$

$$\begin{aligned} [p]_1 = [A]_1 + \gamma[D_c]_1 + \gamma^2\big(aD_a + \delta_1 D_b[B]_1 + \sigma\big(-a[B]_1(y-u) + \delta_2[\hat{I}_x]_1\big) \\ -y[R]_1 - \delta_2 z_H(x)[\hat{R}_x]_1 - z_H(y)[H]_1(y-u)\big) \end{aligned}$$

Output 1 if and only if

$$e\big([p]_1 - [s]_1, [1]_2\big) = e\big([w]_1, [\tau - y]_2\big).$$

Figure 3.13: Basilisk's Prove and Verify algorithms.

Chapter 4

Linear-map Vector Commitments

This chapter is based on the paper 'Linear-map Vector Commitments and their Practical Applications' [CNR+22], which is a joint work with Matteo Campanelli, Anca Nitulescu, Carla Ràfols, and Alexandros Zacharakis.

4.1 Introduction

Vector commitment schemes (VC) were first introduced by Libert and Young [LY10] and Catalano and Fiore [CF13] as commitment schemes with capability to shrink ordered sequences of data, stored in a vector $\mathbf{v}$, and later open specific positions, as an efficiency improvement over commitment schemes to single elements.

A vector commitment scheme is asked to satisfy *position binding*, meaning that no prover should be able to produce valid proofs for opening at a position i to two different values $v_i \neq v_i'$. Additionally, we can require such commitments to be *hiding* so no entity can distinguish between commitments to different vectors.

One key property mentioned in [CF13] is *conciseness*, both the proof of opening and commitment should be independent of m, the size of the committed vector. Also, when working with individual proofs of openings, *updatability* plays a central role: the commitment and the potentially pre-computed and stored proofs of openings should be fast to update when vector $\mathbf{v}$ is changed. In fact, if one piece of data stored in $\mathbf{v}$ changes, the scheme should allow to update the commitment C and all proofs π_i in less time than the required to freshly re-compute them.

In the last years, vector commitments have been discovered to be useful in a plethora of applications, and their study has lead to new capabilities and more general notions.

In 2016, Libert, Ramanna and Yung defined in [LRY16] the concept of Functional Vector Commitments, where the prover has the ability to compute commitments to vectors and later perform openings of linear functions (inner-products) $f : \mathbb{F}^m \to \mathbb{F}$ of these vectors, for some field $\mathbb{F}$. Lai and Malavolta [LM19] and also Boneh, Bünz, and Fisch [BBF19] have defined the property of *subvector openings* as the additional possibility for vector commitments to open to arbitrary subsets of positions I rather than individual ones. In this case, the opening should be of size independent, not only of m, but of $|I|$.

Both vector commitments with subvector openings and functional commitments for inner-products can be captured as vector commitments with openings for a more general class of function families, linear-maps. This fact has been exposed by Lai and Malavolta ([LM19]) and captured in the definiton of Linear Map Commitments (LMC). In such a scheme, the prover is able to open the commitment to some vector $\mathbf{v}$ to the output of multiple linear functions or, equivalently, to the output of one linear-map $f : \mathbb{F}^m \to \mathbb{F}^n$, by producing a single short proof. In this work, we borrow Lai and Malavolta [LM19] LMC notion for full-featured vector commitment generic definition and refer to it as Linear Map Vector Commitment (LVC)[1].

4.1.1 Motivation

Vector commitments are very useful to scale highly decentralized networks of large size and whose content is dynamic [CPZ18, BBF19, CFG+20, GRWZ20](such dynamic content can be the state of a blockchain, amount stored on a wallet, the value of a file in a decentralized storage network, etc.). Beyond the basic requirement that openings should be efficient, in this work we also discuss some of the most prominent applications of LVC to motivate and justify the importance of some additional properties in practice.

In *Verifiable Databases*, a client outsources the storage of a database to a server while keeping the ability to access and change some of its records, i.e. query functions of the data, update some of it, and ensure the server does not tamper with the data. For a VC scheme to be the ideal solution, we require it to support efficient updates and verify linear-map queries. A popular instantiation that achieves efficient updatability are Merkle trees [Mer88], but these are not expressive enough to allow for functional openings.

Stateless Cryptocurrencies are payment systems based on a distributed ledger where neither validators of transactions nor system users need to store the full ledger state. A vector commitment scheme for such applications must have small commitment size, short proofs, efficient computation for openings, and it should allow for aggregation to minimize communication in the transactions and *maintainability* for

[1] We use LVC rather than initially proposed LMC in order to emphasize the Vector Commitment aspect of our notion.

the proofs, as proofs are pre-computed and stored, and the prover should be able to update them in sublinear time.

Proof of Space(PoS), introduced in [DFKP15] and further studied in [RD16, AAC+17, Fis19], is a protocol that allows miners (storage providers) to convince the network that they are dedicating physical storage over time in an efficient way. In a nutshell, a miner commits to a file (data) that uses a specified amount of disk space and then proves that it continues to store the data by answering to recurring audits that consist of random spot-checks.

A PoS construction based on vector commitments, as described in [Fis18], requires short opening proofs for subvectors to be stored in a blockchain, cross-commitments aggregation techniques and the possibility to implement space-time tradeoffs to reduce the proving time for the miner (ideally sublinear in the size of the vector).

In some applications, e.g. when performing HTTP queries, clients use the so-called *prefetching* and receive from a server not only the values of interest but other related values that could potentially be queried in the near future such as values in a neighboring range of the queried values. Vector commitments with efficient proofs for special subset openings allow to add verifiability to such queries in a way that does not affect the speed of the server since the proving procedure for a bigger subset is close or the same as for individual positions.

4.1.2 Related Work

In the seminal work by Catalano and Fiore, [CF13] two constructions were proposed under standard, constant-size, assumptions: CDH in bilinear groups and RSA respectively. Many following works built on these constructions to obtain better efficiency and more properties such as subvector openings, functional openings, aggregation, and updates. A number of works [CFG+20, BBF19] use the properties of hidden order groups to achieve constructions with attractive features such as constant size parameters or incremental aggregation but are concretely less efficient than pairing-based constructions.

Merkle tree-based constructions are widely used in practice nowadays. They only need a transparent setup, offer natural time-memory trade-offs due to their tree structure and are efficiently updatable. Nevertheless, they have linear-size proof, are not expressive in terms of openings, as they are not homomorphic and thus, difficult to aggregate.

The VC schemes based on bilinear groups made their way into offering an alternative to Merkle trees. If both, prover and verifier, have access to some linear-size srs (which for the many constructions happen to be updatable as in Definition 2),

they can achieve constant size proof and verifier work, while offering homomorphic properties. Their main drawback, apart from the size of the public parameters, is the lack of time-memory trade-offs as the prover is usually linear in the size of $\mathbf{v}$, and the inefficient proofs update, as they require usually linear time in $\mathbf{v}$ to update all proofs.

In [LRY16], Libert et al. construct vector commitments with openings to linear-forms of the vector based on the Diffie-Hellman exponent assumption over pairing groups. Later, Lai and Malavolta [LM19] introduce subvector openings and show applications to building succinct-arguments of knowledge (similar applications were shown by [BBF19]) in the bilinear group setting.

A weak variant of updatability requires the algorithms that update the commitment and the opening to take as input an opening for the position in which the vector update occurs called *hints*. Recent RSA-based constructions are hint-updatable [BBF19, CFG$^+$20]. Compared to hint updates, key-updates only need fixed update keys corresponding to the updated positions. Schemes based on bilinear groups require such fixed keys, and no extra information about the change made in the vector in order to update.

Campanelli et al. [CFG$^+$20] showed two constructions of incrementally aggregatable SVCs, that allow to aggregate and dis-aggregate proofs that have already been aggregated, have constant-size parameters and work over groups of unknown order. Unfortunately, the practical efficiency of these constructions is still not sufficient for their deployment in real-world systems.

Gorbunov et al. [GRWZ20] show how to extend the VC scheme of [LY10] to allow for same- and cross-commitment aggregation. Like our constructions, the security of theirs holds under the Algebraic Group Model (AGM) [FKL18] in bilinear groups and a random oracle. However, this approach allows only for one-hop aggregation, meaning that already aggregated proofs cannot be reused in further aggregations by external nodes, and does not consider updatability.

Tomescu et al. [TAB$^+$20] introduced a pairing-based construction of SVC with updatability of proofs and aggregation. Still, the former requires constant work for *each proof*, leading to a linear procedure in order to update all proofs, while the second is one-hop, meaning that proofs can be aggregated only once. They show how all individual proofs can be pre-computed in $O(m \log m)$ time and stored to later perform what in this work we call *native* aggregation, meaning that the aggregation of individual proofs for positions $i \in I$ equals a fresh proof of subvector opening for I (in particular, does not require the execution of a random oracle).

Importantly, in this work we extend both the construction in [GRWZ20] and [TAB$^+$20], showing that they can be adapted to satisfy further properties.

Apart from Merkle tree based Vector Commitments which are known to be main-

tainable, Srinivasan et. al. [SCP+22] show that the multilinear PST polynomial commitment [PST13] can be turned to a maintainable VC construction. Pre-computing all (single-position) opening proofs is done in quasilinear time (contrary to the trivial quadratic time) and updating all proofs after a (single position) vector update needs only logarithmic time. Contrary to Merkle tree based approaches, the scheme has *homomorphic* properties. Furthermore, due to its algebraic structure, it supports one-hop aggregation through generic means, namely, Inner Pairing Product Arguments [BMM+21], albeit with a concretely expensive proving computation. Tomescu et al. [TCZ+20] add the same attribute to KZG polynomial commitment schemes, allowing for a maintainable construction that, similar to ours, works under a tree structure. The main differences are that their tree stores proofs of positions rather than the vectors, and thus cannot be instantiated to open linear maps and proof size and verifier work are both logarithmic on the size of the vector, and that they work with polynomials of the same size at all levels of the tree, leading to $O(m \log m)$ prover work.

4.1.3 Contributions

Theoretical Advances. On the theoretical frontier, we unify previous definitions and augment them with additional properties. The basic notion we use is Linear Map Vector Commitments (LVC) and is borrowed by the work of Lai and Malavolta [LM19]. We then define additional properties on top of this definition and explore their relations. Specifically, we augment this notion with *updatability* and aggregation properties, including a novel notion -*unbounded aggregation*- capturing the ability to aggregate already aggregated proofs but relaxing incremental aggregation [CFG+20] in the sense that the verifier is allowed to do work linear in the number of aggregation hops (i.e. aggregation is "history" dependent), also, disaggregation is not possible. We show that having additional homomorphic properties is highly desirable, by arguing that any LVC that satisfies them: (1) can be augmented with unbounded same- and cross-commitment aggregation as well as updatability; (2) can support general linear map openings (i.e. for any $f : \mathbb{F}^m \to \mathbb{F}^n$) as long as it supports inner product openings (i.e. for $f' : \mathbb{F}^m \to \mathbb{F}$). This allows us to focus on efficient constructions for inner products with homomorphic properties.

VC Constructions. First, we present two pairing-based LVC constructions for inner products based on the properties of monomial and Lagrange polynomial basis and prove that they satisfy all the relevant homomorphic properties to obtain unbounded aggregation and support general linear maps. The latter is based in the inner product construction of Section 3.3. In terms of expressivity, these constructions generalize previous work [SCP+22, TAB+20] by supporting linear functions instead of position or subvector openings. Vector commitments for this class of functions are core components of important primitives such as arguments of knowledge for Inner Product (IP) relations or aggregation arguments [DRZ20].

VC Scheme	Aggregation	Updates	Setup	Assumption	Functional
PoS aggSVC [CFG+20]	Incremental Same-Com	hint	Trusted	RSA	SVC
Pointproofs [GRWZ20]	One-hop Cross-Com	-	Updatable	AGM	SVC
Stateless aggSVC [TAB+20]	One-hop Same-Com	key	Reusable	Computational	SVC
Our Lagrange LVC	Unbounded Cross-Com	key	Reusable	AGM	LVC
Our Monomial LVC	Unbounded Cross-Com	keyless	Reusable	AGM	LVC

Table 4.1: Comparison of our schemes with other recent VC. Updatable setup refers to an srs that consist on monomials but cannot re-use existing ones, as in [GRWZ20] where it needs to be missing an specific power. We later prove that some of the works in this table can be adapted to satisfy further properties.

As in some applications like Proof of Space the subset of opened positions is not very meaningful and its distribution is expected to be known in advance, we study how to improve verification efficiency for certain special subsets I openings in our inner-product constructions. For some structured sets I, we achieve a verifier that performs half of the work it does for arbitrary sets J of the same size in the Lagrange construction, and only a constant number of group operations in the one that uses the monomial basis. For the latter construction, we have efficient range openings and thus can be used in prefetching[2] whenever we need to query a resource whose digest consists of a vector commitment.

Lastly, we prove the inner product in Section 3.3 can be applied to such subsets and use both results to obtain a novel maintainable construction. Our construction allows a stronger, more flexible form of maintainability: it supports arbitrary memory/time trade-offs for openings, meaning that the prover can decide how much memory it wants to use to reduce the opening time. The setup is independent of the trade-off, in particular, it consist solely of monomials, and the relation memory/time can be decided by the prover on the fly.

We compare our LVC constructions with the most efficient ones in Table 4.1

4.2 Generic Constructions from Homomorphic Proofs

In this Section we present a framework to construct pairing-based Linear-map Vector Commitments that satisfy unbounded same- and cross-commitment aggregation, as well as updatability. In particular, we prove that any LVC scheme that has homomorphic proofs, openings and commitments can be endowed with algorithms such that they satisfy the mentioned properties. We present these algorithms for generic constructions and prove that, from any Inner Product argument with homomorphic properties, we can get Liner-map Vector Commitment schemes for arbitrary linear functions $f : \mathbb{F}^m \to \mathbb{F}^n$ with unbounded aggregation and updatability.

[2] `https://developer.mozilla.org/en-US/docs/Web/HTTP/Link_prefetching_FAQ`

4.2.1 New Notion: Unbounded Aggregation

The intuition for our unbounded aggregation definition is that, given n proofs, commitments or openings, we can aggregate them by performing a linear combination with random coefficients. Importantly, these coefficients have to be chosen after the claims are fixed and for that we rely on the RO model.

This approach has been used in previous works, that have defined other types of aggregation. In one-hop aggregation (or batching) [BBF19] aggregated proofs cannot be aggregated further. Incremental aggregation [CFG+20] does not have this limitation. The difference between the latter and our notion is that incremental aggregation does not require to keep track of the order in which the aggregation has been applied (for verification or further aggregation). On the other hand, we do require to track order, but we argue that this is not an overhead in many settings. In particular, even incremental aggregators and verifiers need to know the claims related to the proofs being aggregated, albeit in no order. Adding a structure to the claims roughly adds a number of bits linear in the length of the opening for additional separators.

In our work, aggregating *already aggregated* proofs consist on just sampling new coefficients and using them for fresh linear combinations. Importantly, the verifier needs to have access to the aggregation history: it has to recompute the coefficient corresponding to each proof $\boldsymbol{\pi}$, which is the product of all the coefficients used in the aggregations it was involved in. Note that this also means that the verifier has a small overhead: making a linear (in the number of aggregation "hops") number of hash computations to recompute the challenges. For the rest of this chapter, we will denote the proofs as $\boldsymbol{\pi}$ instead of π, the notation in Definitions 4,6. The reason is that we want to emphasize the fact that proofs of openings are vectors, and so we can operate linearly on them.

Example for same-commitment aggregation: Consider vector $\mathbf{v}$ committed in C, functions f_1, f_2 and f_3; let $\boldsymbol{\pi}_1, \boldsymbol{\pi}_2$ and $\boldsymbol{\pi}_3$ be proofs that $f_1(\mathbf{v}) = \mathbf{y}_1$, $f_2(\mathbf{v}) = \mathbf{y}_2$ and $f_3(\mathbf{v}) = \mathbf{y}_3$. An aggregated proof for $f_2(\mathbf{v}) = \mathbf{y}_2$, $f_3(\mathbf{v}) = \mathbf{y}_3$, would be $\boldsymbol{\pi}_1^* = \boldsymbol{\pi}_2 + \gamma_1 \boldsymbol{\pi}_3$, for $\gamma_1 = \mathsf{H}(\mathsf{C}, \{(f_2, \mathbf{y}_2), (f_3, \mathbf{y}_3)\})$. In a second step, we can aggregate a proof that $f_1(\mathbf{v}) = \mathbf{y}_1$, by performing $\boldsymbol{\pi}_2^* = \boldsymbol{\pi}_1 + \gamma_2 \boldsymbol{\pi}_1^*$, for $\gamma_2 = \mathsf{H}(\mathsf{C}, (f_1, \mathbf{y}_1), \gamma_1)$. At the verification step, the verifier would reconstruct the coefficients of each initial proof in $\boldsymbol{\pi}_2^*$. For instance, $\delta_1 = 1$, $\delta_2 = \gamma_2$, $\delta_3 = \gamma_1\gamma_2$. Then, the verifier can run the $\mathsf{LVC.Verify}$ algorithm to check whether $\boldsymbol{\pi}_2^* = \boldsymbol{\pi}_1 + \gamma_2 \boldsymbol{\pi}_1^* = \boldsymbol{\pi}_1 + \gamma_2 \boldsymbol{\pi}_2 + \gamma_1\gamma_2 \boldsymbol{\pi}_3$ is a valid proof that function $f = f_1 + \gamma_2 f_2 + \gamma_1\gamma_2 f_3$ evaluated at the vector committed in C opens to $y = y_1 + \gamma_2 y_2 + \gamma_1\gamma_2 y_3$. For this last step to work we need the homomorphic proof property and the verifier to have access to the aggregation "history".

To describe our history of claims we move to *trees* of statements $\{f_i, \mathbf{y}_i\}_{i=1}^n$. In these trees, leaves are pairs of function–output $(f, \mathbf{y})$. As in the usual case internal nodes are defined as an ordered list of subtrees. An empty history/tree is referred

to as null. We denote trees using the syntax $T_{f,\mathbf{y}}$ and the operation that "merges" two subtrees in order adding a new root as "$\therefore$". The following definition formalizes the above and will be useful in our construction. We remark that we include the commitment in each of the leaves of the trees $T_{f,\mathbf{y}}$. This does not increase the input size for cross-commitment aggregation where this information is necessary (for same-commitment aggregation the commitment is not necessary). This also allows to model more closely the "claims" for the cross-commitment case where each proof is for a statement $(\mathsf{C}, f, \mathbf{y})$.

Definition 20. *Given a tree T we associate to each of its internal nodes a hash label h defined so that $h(L \therefore R) := \mathsf{H}(\mathsf{C}, L, R)$. We then associate to each of the leaves in the tree a label*

$$\delta(\mathsf{leaf}) := \prod_{i=1,\ldots,t} h(x_i)^{r(x_i,\mathsf{leaf})}$$

where the x_i-s are the internal nodes along the path from leaf *to the root (root included and starting from the bottom), the predicate $r(x, \mathsf{leaf})$ is 1 if* leaf *is a right child of x and 0 otherwise.*

When we consider unbounded-aggregatable LVC, we assume KeyGen outputs additional parameters for aggregations in pp. The aggregation algorithm will follow this syntax[3]:

$$\mathsf{LVC.Agg}(\mathsf{pp}, T_{f,\mathbf{y}}, \boldsymbol{\pi}, T_{f',\mathbf{y}'}, \boldsymbol{\pi}') \rightarrow \boldsymbol{\pi}^*$$

We subsequently modify the syntax for the verification algorithm in an (unbounded) aggregatable LVC as follows:

$$\mathsf{LVC.Verify}(\mathsf{vrk}, \mathsf{C}, T_{f,\mathbf{y}} \therefore T'_{f,\mathbf{y}}, \boldsymbol{\pi}^*) \rightarrow b \in \{0,1\}$$

with $T_{f,\mathbf{y}}$ replacing $f, \mathbf{y}$.

We require the following correctness property and that function binding still holds.

Definition 21 (Unbounded Aggregation Correctness)**.** *For any $T_{f,\mathbf{y}}, T_{f',\mathbf{y}'}$ and any $\boldsymbol{\pi}$, $\boldsymbol{\pi}'$:*

$$\Pr\left[\begin{array}{c|c} \big(\mathsf{LVC.Verify}(\mathsf{vrk}, \mathsf{C}, T_{f,\mathbf{y}}, \boldsymbol{\pi}) = 1 \wedge & (\mathsf{prk}, \mathsf{vrk}, \mathsf{pp}) \leftarrow \mathsf{LVC.KeyGen}(1^\lambda, \mathcal{F}) \\ \mathsf{LVC.Verify}(\mathsf{vrk}, \mathsf{C}, T'_{f,\mathbf{y}}, \boldsymbol{\pi}') = 1\big) \Rightarrow & (\mathsf{C}, \mathsf{aux}) \leftarrow \mathsf{LVC.Commit}(\mathsf{prk}, \mathbf{v}) \\ \mathsf{LVC.Verify}(\mathsf{vrk}, \mathsf{C}, T_{f,\mathbf{y}} \therefore T'_{f,\mathbf{y}}, \boldsymbol{\pi}^*)=1 & \boldsymbol{\pi}^* \leftarrow \mathsf{LVC.Agg}(\mathsf{pp}, T_{f,\mathbf{y}}, \boldsymbol{\pi}, T_{f',\mathbf{y}'}, \boldsymbol{\pi}') \end{array}\right] = 1$$

Definition 22 (Unbounded Aggregation Function Binding)**.** *For any $T_{f,\mathbf{y}}, T_{f',\mathbf{y}'}$ the following probability is negligible in λ:*

$$\Pr\left[\begin{array}{c|c} \mathsf{LVC.Verify}(\mathsf{vrk}, \mathsf{C}, T_{f,\mathbf{y}} \therefore T'_{f,\mathbf{y}}, \boldsymbol{\pi}^*)=1 & (\mathsf{prk}, \mathsf{vrk}, \mathsf{pp}) \leftarrow \mathsf{LVC.KeyGen}(1^\lambda, \mathcal{F}) \\ \wedge\ \nexists \mathbf{a} \text{ s.t. } f(\mathbf{a}) = \mathbf{y} \wedge f'(\mathbf{a}) = \mathbf{y}' & (\mathsf{C}, \boldsymbol{\pi}^*, T_{f,\mathbf{y}}, T'_{f,\mathbf{y}}) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{prk}, \mathsf{vrk}) \end{array}\right]$$

[3]The algorithms can be generalized for more proofs. Proof size remains the same, also for cross-commitment aggregation.

Cross-Commitment Aggregation

Unbounded aggregation can be performed across different commitments as well. This property is called *Cross-commitment Aggregation* and makes sense when we have a set of commitments $\mathsf{C}'_1, \ldots, \mathsf{C}'_t$ that we want to open at one or more maps f, as it allows to compute a succinct proof of opening for linear-maps from different vectors committed separately. Below we show our syntax which directly expands on our same-commitment aggregation described above. Function binding and correctness are also straightforward to expand. We let $T_{f,\mathbf{y}}$ include our commitments in the leaves (see also next section).

Cross-commitment aggregation:

$$\mathsf{LVC.CrossAgg}(\mathsf{pp}, T_{f,\mathbf{y}}, \boldsymbol{\pi}, T_{f',\mathbf{y}'}, \boldsymbol{\pi}') \rightarrow \boldsymbol{\pi}^*$$

Cross-commitment verification:

$$\mathsf{LVC.CrossVfy}(\mathsf{vrk}, \left(\mathsf{C}'_j\right)_j, T_{f,\mathbf{y}}, \boldsymbol{\pi}^*) \rightarrow 0/1$$

4.2.2 Unbounded Aggregation for LVC

We now describe unbounded aggregation algorithms for *any* LVC scheme that satisfies the homomorphic properties of Section 2.3.6.

$\mathsf{LVC.KeyGen}(1^\lambda, \mathcal{F})$**:**
Run $(\mathsf{prk}, \mathsf{vrk}) \leftarrow \mathsf{LVC.KeyGen}(1^\lambda, \mathcal{F})$.
Generate the description of a hash function $\mathsf{H}(\cdot)$ and set it in pp.
Output $(\mathsf{prk}, \mathsf{vrk}, \mathsf{pp})$.

$\mathsf{LVC.Agg}(\mathsf{pp}, T_{f,\mathbf{y}}, \boldsymbol{\pi}, T_{f',\mathbf{y}'}, \boldsymbol{\pi}')$:
Compute $\gamma = \mathsf{H}(\mathsf{C}, T_{f,\mathbf{y}}, T_{f',\mathbf{y}'})$
Output $\boldsymbol{\pi}^* = \boldsymbol{\pi} + \gamma\boldsymbol{\pi}'$

$\mathsf{LVC.Verify}(\mathsf{vrk}, \mathsf{C}, T_{f,\mathbf{y}} \therefore T_{f',\mathbf{y}'}, \boldsymbol{\pi}^*)$:
Return $b \leftarrow \mathsf{LVC.Verify}\big(\mathsf{vrk}, \mathsf{C}, f^*, y^*, \boldsymbol{\pi}^*\big)$ where:

- let $\mathsf{leaf}_1, \ldots, \mathsf{leaf}_\ell$ be all the leaves in $T_{f,\mathbf{y}} \therefore T_{f',\mathbf{y}'}$.
- recall each leaf_i is of the form $(\mathsf{C}, f_i, \mathbf{y}_i)$
- For each i let $\delta_i := \delta(\mathsf{leaf}_i)$ be the value defined as in Definition 20,

$$f^* := \sum_i \delta_i f_i \qquad \mathbf{y}^* := \sum_i \delta_i \mathbf{y}_i.$$

Figure 4.1: Unbounded aggregation for LVC schemes with homomorphic proofs.

Theorem 11. *Any function binding LVC scheme with homomorphic proofs endowed with* $(\mathsf{LVC.Agg}, \mathsf{LVC.Verify})$ *as described in Figure 4.1 satisfies Unbounded Aggregation Correctness (as in Def. 21) and Function Binding (Def. 22) in the ROM.*

Proof. Correctness follows by inspection, using the fact that the LVC satisfies homomorphic proof, so we omit it.

For function binding, let $\mathcal{A}$ be an adversary against it, and $(\mathsf{C}, \boldsymbol{\pi}^*, T_{f,\mathbf{y}}, T_{f',\mathbf{y}'})$ be an output of them such that $\mathsf{LVC.Verify}(\mathsf{vrk}, \mathsf{C}, T_{f,\mathbf{y}} \therefore T_{f',\mathbf{y}'}, \boldsymbol{\pi}^*)$=1. By construction this implies $\mathsf{LVC.Verify}\big(\mathsf{vrk}, \mathsf{C}, \sum_i \delta_i f_i, \sum_i \delta_i \mathbf{y}_i, \boldsymbol{\pi}^*\big) = 1$. Because LVC is function binding, except with negligible probability, there exists a vector $\mathbf{a}$ such that $f(\mathbf{a}) = \mathbf{y}$, for $\mathbf{y} = \sum_i \delta_i \mathbf{y}_i$ and $f(\mathbf{X}) = \sum_i \delta_i f_i(\mathbf{X})$, i.e., there exists $\mathbf{a}$ such that:

$$\sum_{i=1}^{t} \delta_i f_i(\mathbf{a}) = \sum_{i=1}^{t} \delta_i \mathbf{y}_i.$$

Since H is a random oracle, the coefficients δ_i do not depend on $\mathbf{y}_i, f_i$, and by the Schwartz-Zippel lemma, except with probability $m/\mathbb{F}$, where m is the degree of f, $f_i(\mathbf{a}) = \mathbf{y}_i$ for all i, which concludes the proof. □

Cross-Commitment Aggregation.

$\mathsf{LVC.CrossAgg}(\mathsf{pp}, T_{f,\mathbf{y}}, \boldsymbol{\pi}, T_{f',\mathbf{y}'}, \boldsymbol{\pi}')$:
Compute $\gamma = \mathsf{H}(T_{f,\mathbf{y}}, T_{f',\mathbf{y}'})$
Output $\boldsymbol{\pi}^* = \boldsymbol{\pi} + \gamma\boldsymbol{\pi}'$

$\mathsf{LVC.CrossVfy}(\mathsf{vrk}, (\mathsf{C}, \mathsf{C}', T_{f,\mathbf{y}} \therefore T_{f',\mathbf{y}'}, \boldsymbol{\pi}^*)$:

- let $\mathsf{leaf}_1, \ldots, \mathsf{leaf}_\ell$ be all the leaves in $T_{f,\mathbf{y}} \therefore T_{f',\mathbf{y}'}$. We add to each leaf leaf_i and additional subindex j that refers to which commitment the proof in leaf_{ij} corresponds to. Note that we still consider ℓ leaves.
- each leaf_{ij} is of the form $(C_j, f_i, \mathbf{y}_i)$
- For each i let $\delta_{ij} := \delta(\mathsf{leaf}_{ij})$ be the value defined as in Definition 20.
- Compute
$$f_j^* := \sum_i \delta_{ij} f_i \qquad y_j^* := \sum_i \delta_{ij} \mathbf{y}_i$$
- Return 1 iff $b_j = 1$ for all $b_j \leftarrow \mathsf{LVC.Verify}\big(\mathsf{vrk}, \mathsf{C}_j, f_j^*, y_j^*, \boldsymbol{\pi}^*\big)$.

Figure 4.2: Cross-commitment aggregation for LVC schemes with homomorphic commitments and proofs.

For the case of cross-commitment aggregation, we proceed similarly but we also need to homomorphically operate on the commitments (recall that hashing on trees implicitly hashes the commitments too since we include them there), as described in Figure 4.2.

Efficiency. For our constructions, the verification equations for computing $b_i = \mathsf{IP.Verify}\big(\mathsf{vrk}, \mathsf{C}^*, f^*, y^*, \boldsymbol{\pi}^*\big)$ are two pairing equations where the elements in the right side can be aggregated, and thus the verifier performs only $\ell + 1$ pairings.

Security. The security of this augmented construction follows analogously to that for same-commitment aggregation, with the additional requirement for the LVC scheme to have homomorphic commitments and openings.

4.2.3 Updability for LVC

We consider updatability as an extra property of an LVC scheme. The KeyGen algorithm additionally computes the update key upk, while two extra algorithms are defined as follows:

$\mathsf{LVC.UpdCom}(\mathsf{upk}, \mathsf{C}, t, \delta) \to \mathsf{C}'$**:** takes as input C, a position $t \in [m]$, update key upk, and a constant $\delta \in \mathcal{M}$. It outputs C' as a commitment for $\mathbf{v}' = \mathbf{v} + \delta\mathbf{e}_t$[4].

$\mathsf{LVC.UpdOpen}(\mathsf{upk}, t, \delta, f, \mathbf{y}, \boldsymbol{\pi}) \to \boldsymbol{\pi}'$: Takes as input upk, t, δ, a function f, a valid opening pair $(\mathbf{y}, \boldsymbol{\pi})$ for f and outputs a proof π' for the new opening $\mathbf{y}' = f(\mathbf{v} + \delta\mathbf{e}_t)$.

Update correctness. Let $\big(\mathsf{prk}, \mathsf{vrk}, \mathsf{upk}\big) \leftarrow \mathsf{LVC.KeyGen}(1^\lambda, \mathcal{F})$, and let $(\mathsf{C}, j, f, \mathbf{y}, \boldsymbol{\pi})$ be a tuple such that $\mathsf{LVC.Verify}(\mathsf{vrk}, \mathsf{C}, f, \mathbf{y}, \boldsymbol{\pi}) = 1$. Then LVC satisfies update correctness if for any $\delta \in \mathcal{M}$,

$$\Pr\left[\begin{array}{c}\mathsf{LVC.Verify}(\mathsf{vrk}, \mathsf{C}', f, \mathbf{y}', \boldsymbol{\pi}')=1 \\ \wedge\ \mathbf{y}' = \mathbf{y} + \delta f(\mathbf{e}_t)\end{array}\middle|\begin{array}{c}\mathsf{C}' \leftarrow \mathsf{LVC.UpdCom}(\mathsf{upk}_j, \mathsf{C}, t, \delta) \\ \boldsymbol{\pi}' \leftarrow \mathsf{LVC.UpdOpen}(\mathsf{upk}_j, t, \delta, f, \mathbf{y}, \boldsymbol{\pi})\end{array}\right] = 1.$$

4.2.4 From Inner-Products to Arbitrary Linear-Maps

In this section we show we can obtain LVC schemes for any family of functions $\mathcal{F} \subset \{f : \mathbb{F}^m \to \mathbb{F}^n\}$ starting from simpler constructions that have homomorphic proofs and openings. Our starting point are LVC schemes for $\mathcal{F}_{\mathsf{IP}_m} = \{f : \mathbb{F}^m \to \mathbb{F}\}$, or inner-product VC schemes, that we will denote as $\mathsf{IP} = (\mathsf{IP.KeyGen}, \mathsf{IP.Commit}, \mathsf{IP.Open}, \mathsf{IP.Verify})$. All this algorithms work as the ones for LVC, except that instead of $f \in \mathcal{F}_{\mathsf{IP}_m}$, they use the vector $\mathbf{f} \in \mathbb{F}^m$ so that $f(\mathbf{v}) = \mathbf{f} \cdot \mathbf{v}$.

[4]This notion can be generalized to more than one position.

We can write any linear-map $f : \mathbb{F}^m \to \mathbb{F}^n$ as $f = (f_1, f_2, \dots f_n)$, where each f_i is an inner product function. If the IP scheme has homomorphic proofs, and we set π_i to be the proof that $f_i(\mathbf{v}) = \mathbf{f}_i \cdot \mathbf{v} = y_i$, an aggregation of $\{\pi_i\}_{i=1}^n$ is a proof of the statement $f(\mathbf{v}) = \mathbf{y}$. Later, in the following section, we show two possible constructions of IP vector commitments schemes that can be used to instantiate the framework in this section. A one-hop aggregation algorithm for IP[5] works as described in Figure 4.3, and we present in Figure 4.4 an alternative way of computing concise proofs of LVC for more general functions $f : \mathbb{F}^m \to \mathbb{F}^n$, using $\mathsf{IP.Agg}$.

$\mathsf{IP.Agg}(\mathsf{pp}, \{\mathbf{f}_i, y_i\}_{i=1}^n, \boldsymbol{\pi} = (\boldsymbol{\pi}_i)_{i=1}^n)$:
Parse $\mathsf{pp} = \mathsf{H}$, where H is a hash function, compute $\gamma = \mathsf{H}(\mathsf{C}, \{\mathbf{f}_i, y_i\}_{i=1}^n)$
Output $\boldsymbol{\pi}' = \sum_{i=1}^n \gamma^{i-1} \boldsymbol{\pi}_i$.

$\mathsf{IP.VfAgg}(\mathsf{vrk}, \mathsf{C}, \{\mathbf{f}_i, y_i\}_{i=1}^n, \boldsymbol{\pi}') \to b$:
Compute $\gamma = \mathsf{H}(\mathsf{C}, \{\mathbf{f}_i, y_i\}_{i=1}^n)$, $\quad \mathbf{f}' = \sum_{i=1}^n \gamma^{i-1} \mathbf{f}_i$, $\quad y' = \sum_{i=1}^n \gamma^{i-1} y_i$
Output $b \leftarrow \mathsf{IP.Verify}(\mathsf{vrk}, \mathsf{C}, \mathbf{f}', y', \boldsymbol{\pi}')$.

Figure 4.3: Aggregation for Inner Product arguments with homomorphic proofs

$\mathsf{LVC.KeyGen}(1^\lambda, \mathcal{F}_{\mathsf{IP},m})$**:**
Run $(\mathsf{prk}, \mathsf{vrk}) \leftarrow \mathsf{IP.KeyGen}(1^\lambda, \mathcal{F}_{\mathsf{IP},m})$
Generate aggregation parameters $\mathsf{pp} = \mathsf{H}$ (a hash function).
Output $(\mathsf{prk}, \mathsf{vrk}, \mathsf{pp})$.

$\mathsf{LVC.Commit}(\mathsf{prk}, \mathbf{v})$:
Run $(\mathsf{C}, \mathsf{aux}) \leftarrow \mathsf{IP.Commit}(\mathsf{prk}, \mathbf{v})$
Output $(\mathsf{C}, \mathsf{aux})$.

$\mathsf{LVC.Open}(\mathsf{prk}, \mathsf{pp}, \mathsf{aux}, f, \mathbf{y})$:
Parse $f = (f_1, f_2, \dots f_n)$ and $\mathbf{y} = (y_1, \dots y_n)$. Consider $\mathbf{f}_i$ as the vector representing inner-product function f_i.
Run $\boldsymbol{\pi}_i \leftarrow \mathsf{IP.Open}(\mathsf{prk}, \mathsf{aux}, \mathbf{f}_i, y_i)$ for $i \in [n]$
Output $\boldsymbol{\pi} \leftarrow \mathsf{IP.Agg}(\mathsf{pp}, \{\boldsymbol{\pi}_i\}_{i=1}^n)$.

$\mathsf{LVC.VfAgg}(\mathsf{vrk}, \mathsf{pp}, \mathsf{C}, f, \mathbf{y}, \boldsymbol{\pi})$:
Parse $f = (f_1, f_2, \dots, f_n)$ and $\mathbf{y} = (y_1, \dots y_n)$. Consider $\mathbf{f}_i$ as the vector representing inner-product function f_i.
Output $b \leftarrow \mathsf{IP.VfAgg}(\mathsf{vrk}, \mathsf{C}, \{\mathbf{f}_i, y_i\}_{i=1}^n, \boldsymbol{\pi})$.

Figure 4.4: LVC schemes from Inner Product arguments with homomorphic properties.

[5]Naturally, this can be seen as a particular case of unbounded aggregation.

Updates for IP.

We present a generic construction of the updatability algorithms for inner-product schemes in Figure 4.5. We state that even though algorithms can be generalized to LVC for arbitrary functions, storing and updating is a property desired (to the best of our knowledge) only for individual openings, which can be represented as inner product arguments an thus this is the only scenario where we focus on adding updatability.

It is easy to see that commitments can be updated when one value of the vector changes by simply applying the linear-homomorphic property of the underlying IP scheme. Given C such that $(\mathsf{C}, \mathsf{aux}) \leftarrow \mathsf{LVC.Commit}(\mathsf{prk}, \mathbf{v})$, when position t of the vector changes, i.e. $\mathbf{v}' = \mathbf{v} + \delta\mathbf{e}_t$ we can compute a commitment to the new vector $\mathbf{v}'$ as $\mathsf{C}' = (\mathsf{C} + \delta\hat{\mathsf{C}})$ where $(\hat{\mathsf{C}}, \hat{\mathsf{aux}}) \leftarrow \mathsf{LVC.Commit}(\mathsf{prk}, \mathbf{e}_t)$ is given as an update key.

$\mathsf{IP.KeyGen}(1^\lambda, \mathcal{F}_{\mathsf{IP},m})$:

Run $(\mathsf{prk}, \mathsf{vrk}) \leftarrow \mathsf{IP.KeyGen}(1^\lambda, \mathcal{F}_{\mathsf{IP},m})$

Additionally generate public update keys upk:

- Set $\boldsymbol{\pi}_{u_{ij}} \leftarrow \mathsf{IP.Open}(\mathsf{prk}, \mathsf{aux}_j, \mathbf{e}_i, u_{ij} = \mathbf{e}_i \cdot \mathbf{e}_j),\ \forall i, j \in [m]$
- Define $\mathsf{upk}_j = \{\boldsymbol{\pi}_{u_{ij}}\}_{i=1}^m$ for all $j \in [m]$

Output $(\mathsf{prk}, \mathsf{vrk}, \{\mathsf{upk}_j\}_{j=1}^m)$.

$\mathsf{IP.UpdCom}(\mathsf{prk}, \mathsf{C}, t, \delta)$:

Set $\hat{\mathsf{C}} \leftarrow \mathsf{IP.Commit}(\mathsf{prk}, \mathbf{e}_t)$.

Output $\mathsf{C}' = \mathsf{C} + \delta\hat{\mathsf{C}}$.

$\mathsf{IP.UpdOpen}(\mathsf{upk}_t, t, \delta, \mathsf{C}, \mathbf{f}, y, \boldsymbol{\pi})$:

Parse $\mathsf{upk}_t = \{\boldsymbol{\pi}_{u_{it}}\}_{i=1}^m$.

Compute $\hat{\boldsymbol{\pi}} = \sum_{i=1}^m f_i \boldsymbol{\pi}_{u_{it}}$.

Set $\boldsymbol{\pi}' = \boldsymbol{\pi} + \delta\hat{\boldsymbol{\pi}}$ as proof for $y' = y + \mathbf{f} \cdot \delta\mathbf{e}_t$.

Output $\boldsymbol{\pi}'$.

Figure 4.5: Updatability algorithms for IP arguments with homomorphic openings and commitments.

Moreover, it is possible to update existing proofs using the homomorphic openings property of the IP scheme: when position t of the vector changes as above, to update a prior proof we simply add to $\boldsymbol{\pi}$ a proof $\hat{\boldsymbol{\pi}}$ corresponding to the opening of $f(\delta\mathbf{e}_t)$. The resulting $\boldsymbol{\pi}' = \boldsymbol{\pi} + \hat{\boldsymbol{\pi}}$ corresponds to the opening of the sum $f(\mathbf{v}') = f(\mathbf{v}) + \delta f(\mathbf{e}_t)$ with respect to the updated commitment $\mathsf{C}' = \mathsf{C} + \delta\hat{\mathsf{C}}$.

We extend IP arguments to satisfy updatability by asking the $\mathsf{IP.KeyGen}$ al-

gorithm to additionally generate updatable keys and introduce IP.UpdCom and IP.UpdOpen that work as in Fig. 4.5.

Theorem 12. *If* IP *satisfies function binding and has homomorphic commitments and openings, the extension above satisfies update correctness.*

Proof. The proof follows directly by the definitions of the homomorphic properties and IP.UpdCom, IP.UpdOpen. □

4.3 Constructions for Inner-Pairing VC

In this section, we present two constructions of LVC for inner products, that is, for functions $f \subset \mathcal{F}_{\mathsf{IP},m} = \{f : \mathbb{F}^m \to \mathbb{F}\}$. Naturally, the first construction we consider is the one presented in Section 3.3 of this thesis. We re-introduce the sumcheck argument as a LVC commitment scheme for inner product functions, which implies some extra singularities, as well as different security definitions. Then, we present its analogous for encodings in the monomial basis. We prove they are indeed linear vector commitment arguments with homomorphic proofs and openings. Therefore, they can be used as a starting point to obtain further aggregation properties as shown in Section 4.2.1 and, in particular, lead to two different more generic linear-map vector commitment schemes.

4.3.1 Lagrange Basis

Using the Lagrange basis $\{\lambda_j(X)\}_{j=1}^m$ over a multiplicative group $\mathbb{H} = \{\mathsf{h}_1, \ldots, \mathsf{h}_m\}$ of size m in $\mathbb{F}_p$ we encode a vector $\mathbf{a} \in \mathbb{F}_p^m$ as a polynomial $a(X) = \sum_{j=1}^m a_i \lambda_j(X)$. The construction uses few properties of Lagrange basis over multiplicative groups that we would like to remind before formally presenting our scheme. When $\mathbb{H}$ is a multiplicative subgroup, $\lambda_j(0) = m^{-1}$ for all $j \in [m]$. Moreover, for the Lagrange basis and the vanishing polynomial $z_H(X) = \prod_{j=1}^m (X - \mathsf{h}_j)$ we have that

$$\lambda_j(X)\lambda_i(X) \equiv 0 \mod z_H(X), \qquad \lambda_j(X)^2 \equiv \lambda_j(X) \mod z_H(X).$$

The construction in Figure 4.6 exploits these properties in the proof of openings for inner-products:

$\mathsf{IP.KeyGen}(1^\lambda, \mathcal{F}_{\mathsf{IP}_m})$**:** Sample $\tau \leftarrow \mathbb{F}_p$ and output $\mathsf{prk} = \big(\{[\tau^j]_{1,2}, [\lambda_j(\tau)]_1\}_{j=1}^m\big)$ and $\mathsf{vrk} = \Big([1]_{1,2}, \{[\tau^j]_2, [\lambda_j(\tau)]_2\}_{j=1}^m\Big)$.

$\mathsf{IP.Commit}(\mathsf{prk}, \mathbf{a})$**:** Compute $\mathsf{C}_a = \sum_{j=1}^m a_j[\lambda_j(\tau)]_1$ and output $(\mathsf{C}_a, \mathbf{a})$.

$\mathsf{IP.Open}(\mathsf{prk}, \mathsf{aux}, \mathbf{b}, y)$:

Find $R(X), Q(X)$ such that $\deg(R) < m-1$ and

$$\left(\sum_{j=1}^m a_j\lambda_j(X)\right)\left(\sum_{j=1}^m b_j\lambda_j(X)\right) - m^{-1}y = XR(X) + z_H(X)Q(X)$$

Define $\hat{R}(X) = X^2R(X)$ and output $\boldsymbol{\pi} = ([R(\tau)]_1, [Q(\tau)]_1, [\hat{R}(\tau)]_1)$.

$\mathsf{IP.Vf}(\mathsf{vrk}, \mathsf{C}_a, \mathbf{b}, y, \pi)$: Calculate $\mathsf{C}_b = \sum_{j=1}^m b_j[\lambda_j(\tau)]_2$

Parse $\boldsymbol{\pi} = ([R]_1, [Q]_1, [\hat{R}]_1)$ and output 1 if and only if

$$e\big(\mathsf{C}_a, \mathsf{C}_b\big) - e\big(m^{-1}y[1]_1, [1]_2\big) = e\big([R]_1, [\tau]_2\big) + e\big([Q]_1, [z_H(\tau)]_2\big), \text{ and}$$

$$e\big([R]_1, [\tau^2]_2\big) = e\big([\hat{R}]_1, [1]_2\big).$$

Figure 4.6: Inner Product argument with Lagrange Basis.

Security

We omit the proof of completeness as it can be found in Section 3.3. Still, since there it is presented as an argument for inner-product relations as opposite to LVC scheme as considered in this chapter, we prove Strong Function Binding and homomorphic proofs and openings below.

Theorem 13. *The protocol in Figure 4.6 is a strong function binding LVC scheme under the algebraic group model if the qDHE and dLog assumptions hold.*

Proof. We will proceed through a series of games, and we set Game_0 to be the strong binding game of Definition 6. Let $\mathcal{A}$ be an adversary against of the scheme in Figure 4.6, whose advantage is $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{s.binding}}$. We define Game_1 and specify a reduction $\mathcal{B}_1$ such that

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{s.binding}} \leq \mathsf{Adv}_{\mathcal{B}_1}^{q\mathrm{DHE}} + \mathsf{Adv}_{\mathcal{B}_2}^{dlog}.$$

Let Game_1 be the game that goes exactly as Game_0 except that, upon receiving $[R]_1, [\hat{R}]_1$ from $\mathcal{A}$, it checks whether $\deg(R) \leq m-2$, where $R(X)$ is the algebraic representation of $[R]_1$ and aborts if it is not. If $\mathcal{A}$ wins Game_0 but not Game_1, then we construct $\mathcal{B}_1$ that extracts $R(X) = \sum_{s=0}^m r_sX^s$ as the algebraic representation

of $[R]_1$ where $\hat{r}_s \neq 0$ for $s = m-1$ or $s = m$. Then, $\mathcal{B}_1$ sets $\hat{R}'(X) = X^2 R(X) = \sum_{s=0}^{m} \hat{r}_s X^{2+s}$. Note that, from the second verification equation $[\hat{R}'(\tau)]_1 = [\hat{R}]_1$.

Now, $\mathcal{B}_1$ outputs $\left([\hat{R}]_1 - [R'(\tau)]_1\right)\frac{1}{r_s} = [\tau^{2+s}]_1$, wining qDHE as $2+s > m$, the highest available power of τ in $\mathbb{G}_1$. Thus,

$$\mathsf{Adv}^{\mathsf{s.binding}}_{\mathcal{A}} = \mathsf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{q\mathrm{DHE}}_{\mathcal{B}_1}(\lambda).$$

Now, we prove that the advantage of $\mathcal{A}$ in Game_1 is negligible. Indeed, let $C_a(X) = \sum_{j=1}^{m} a_j \lambda_j(X) + X^m \hat{a}$, $R(X)$ and $Q(X)$ be the algebraic representations of C_a, $[R]_1$ and $[Q]_1$, and recall $\deg(R) \leq m-2$ while $\deg(C_a), \deg(Q) \leq m$.

We set $P(X) = C_a(X)C_b(X) - m^{-1}y - XR(X) - Q(X)z_H(X)$, the first verification equation says that $P(\tau) = 0$, which means either that (i) $P(X)$ is the zero polynomial, or (ii) τ is a root of it. Assume for now that $P(X) \equiv 0$, then

$$\left(\sum_{j=1}^{m} a_j \lambda_j(X) + X^m \hat{a}\right)\left(\sum_{j=1}^{m} b_j \lambda_j(X)\right) = m^{-1}y + XR(X) + z_H(X)Q(X)$$

Because $deg(R) \leq m-2$, we know $z_H(X)$ does not divide $XR(X)$ and since for all $i \neq j$ $\lambda_i(X)\lambda_j(X) \equiv 0 \mod z_H(X)$ and $\lambda_i^2(X) \equiv \lambda_i(X) \mod z_H(X)$, we have that $\left(\sum_{j=1}^{m} a_j\lambda_j(X)\right)\left(\sum_{j=1}^{m} b_j\lambda_j(X)\right) = \sum_{j=1}^{m} a_j b_j \lambda_j(X) \mod z_H(X)$. Then,

$$\sum_{j=1}^{m} a_j b_j \lambda_j(X) + X^m \hat{a} \sum_{j=1}^{m} b_j \lambda_j(X) = m^{-1}y + XR(X)$$

and thus $m^{-1}y = \sum_{j=1}^{m} a_j b_j \lambda_j(0)$. As $\mathbb{H}$ is a multiplicative subgroup, $\lambda_j(0) = m^{-1}$ for all $j \in [m]$ and thus $\sum_{j=1}^{m} a_j b_j = y$. Namely, there exists $\mathbf{a} = (a_j)_{j=1}^{m}$ such that $\mathbf{a} \cdot \mathbf{b} = y$, and $\mathcal{A}$ looses Game_1.

Then, it must be the case that $P(X) \neq 0$ and $P(\tau) = 0$. We construct an adversary $\mathcal{B}_2$ against the *dlog* assumption. On input $[\tau]_1$, $\mathcal{B}_2$ calculates all the roots of $P(X)$ and checks, in polynomial time, which is the one that encoded in $\mathbb{G}_1$ equals $[\tau]_1$. Thus,

$$\mathsf{Adv}^{\mathsf{s.binding}}_{\mathcal{A}} \leq \mathsf{Adv}^{q\mathrm{DHE}}_{\mathcal{B}_1} + \mathsf{Adv}^{dlog}_{\mathcal{B}_2}.$$

□

Theorem 14. *The construction above has Homomorphic Proofs and Openings.*

Proof.
Homomorphic Proofs. Let $y_b = \mathbf{a} \cdot \mathbf{b}$, $y_c = \mathbf{a} \cdot \mathbf{c}$, $\boldsymbol{\pi}_b \leftarrow \mathsf{IP.Prove}(\mathsf{srs}, \mathbf{a}, \mathbf{b}, y_b)$ and $\boldsymbol{\pi}_c \leftarrow \mathsf{IP.Prove}(\mathsf{srs}, \mathbf{a}, \mathbf{c}, y_c)$, where $\boldsymbol{\pi}_b = ([R_b(\tau)]_1, [Q_b(\tau)]_1, [\hat{R}_b(\tau)]_1)$, $\boldsymbol{\pi}_c = ([R_c(\tau)]_1, [Q_c(\tau)]_1, [\hat{R}_c(\tau)]_1)$ are such that

$$\left(\sum_{j=1}^{m} a_j\lambda_j(X)\right)\left(\sum_{j=1}^{m} b_j\lambda_j(X)\right) - m^{-1}y_b = XR_b(X) + z_H(X)Q_b(X),$$

$$\left(\sum_{j=1}^{m} a_j\lambda_j(X)\right)\left(\sum_{j=1}^{m} c_j\lambda_j(X)\right) - m^{-1}y_c = XR_c(X) + z_H(X)Q_c(X),$$

$$\text{and } \hat{R}_b(X) = X^2R_b(X), \hat{R}_c(X) = X^2R_c(X).$$

In order to compute a proof that $\mathbf{a}\cdot(\alpha\mathbf{b}+\beta\mathbf{c}) = \alpha y_b + \beta y_c$, the prover proceeds as follows:

$$\begin{aligned}
&\left(\sum_{j=1}^{m} a_j\lambda_j(X)\right)\left(\alpha\sum_{j=1}^{m} b_j\lambda_j(X) + \beta\sum_{j=1}^{m} c_j\lambda_j(X)\right)\\
&= \alpha\left(\sum_{j=1}^{m} a_j\lambda_j(X)\right)\left(\sum_{j=1}^{m} b_j\lambda_j(X)\right) + \beta\left(\sum_{j=1}^{m} a_j\lambda_j(X)\right)\left(\sum_{j=1}^{m} c_j\lambda_j(X)\right)\\
&= \alpha\left(m^{-1}y_b + XR_b(X) + z_H(X)Q_b(X)\right) + \beta\left(m^{-1}y_c + XR_c(X) + z_H(X)Q_c(X)\right)\\
&= m^{-1}(\alpha y_b + \beta y_c) + X(\alpha R_b(X) + \beta R_c(X)) + z_H(X)(\alpha Q_b(X) + \beta Q_c(X)),
\end{aligned}$$

and therefore for $y = \alpha y_b + \beta y_c$ it outputs $\boldsymbol{\pi} = ([R(\tau)]_1, [Q(\tau)]_1, [\hat{R}(\tau)]_1)$ where $R(X) = \alpha R_b(X) + \beta R_c(X)$, $Q(X) = \alpha Q_b(X) + \beta Q_c(X)$ and $\hat{R}(X) = X^2R(X) = \alpha X^2R_b(X) + \beta X^2R_c(X) = \alpha\hat{R}_b(X) + \beta\hat{R}_c(X)$, i.e., $\boldsymbol{\pi} = \alpha\boldsymbol{\pi}_b + \beta\boldsymbol{\pi}_c$.

Homomorphic Openings. The proof for homomorphic openings work analogous as the previous case. Indeed, for $y_a = \mathbf{a}\cdot\mathbf{c}$, $y_b = \mathbf{b}\cdot\mathbf{c}$ and $\boldsymbol{\pi}_a \leftarrow \mathsf{IP.Prove}(\mathsf{srs}, \mathbf{a}, \mathbf{b}, y_a)$, $\boldsymbol{\pi}_c \leftarrow \mathsf{IP.Prove}(\mathsf{srs}, \mathbf{c}, \mathbf{b}, y_c)$, it is enough to see that:

$$\begin{aligned}
&\left(\alpha\sum_{j=1}^{m} a_j\lambda_j(X) + \beta\sum_{j=1}^{m} c_j\lambda_j(X)\right)\left(\sum_{j=1}^{m} b_j\lambda_j(X)\right)\\
&= \alpha\left(\sum_{j=1}^{m} a_j\lambda_j(X)\right)\left(\sum_{j=1}^{m} b_j\lambda_j(X)\right) + \beta\left(\sum_{j=1}^{m} c_j\lambda_j(X)\right)\left(\sum_{j=1}^{m} b_j\lambda_j(X)\right)\\
&= \alpha\left(m^{-1}y_a + XR_a(X) + z_H(X)Q_a(X)\right)\\
&+\beta\left(m^{-1}y_c + XR_c(X) + z_H(X)Q_c(X)\right)\\
&= m^{-1}(\alpha y_a + \beta y_c) + X(\alpha R_a(X) + \beta R_c(X)) + z_H(X)(\alpha Q_a(X) + \beta Q_c(X)),
\end{aligned}$$

and the rest of the proof is the same as the one for homomorphic proofs. □

Updatability with Hints

In this construction, a proof that $\mathbf{e}_j\cdot\mathbf{e}_j = 1$ is $[R_j(\tau)]_1$, for $R_j(X) = (\lambda_j(X) - 1)/X$. On the other hand, the proof that $\mathbf{e}_j\cdot\mathbf{e}_i = 0$ for $i \neq j$ is $[Q(\tau)]_1$, for $Q(X) =$

$((\lambda_j(X)\lambda_i(X))/z_H(X)$. Including the evaluation of all these polynomials in upk would require a srs of quadratic size. Still, as noted in [TAB+20],

$$\frac{\lambda_j(X)\lambda_i(X)}{z_H(X)} = \frac{z_H(X)}{(X-\mathsf{h}_j)(X-\mathsf{h}_i)},$$

and can be computed as

$$\frac{1}{\mathsf{h}_j-\mathsf{h}_i}\left(\frac{z_H(X)}{X-\mathsf{h}_j}+\frac{z_H(X)}{X-\mathsf{h}_i}\right).$$

Therefore, it is enough to include in upk the evaluations of $(\lambda_j(X)-1)/X$ for the proofs of same position and then the evaluations of $\{z_H(X)/(X-\mathsf{h}_j)\}_{j=1}^m$, so the verifier can reconstruct the one of $\lambda_j(X)\lambda_i(X)/z_H(X)$ from there, requiring access to $2m$ elements instead of m^2.

4.3.2 Monomial Basis

For this scheme, presented in Figure 4.7 we consider vectors $\mathbf{a} \in \mathbb{F}_p^m$ encoded as a polynomial in the monomial basis, that is as $a(X) = \sum_{j=1}^m a_j X^{j-1}$.

$\mathsf{KeyGen}(1^\lambda, \mathcal{F}_{\mathsf{IP}_m})$**:** Sample $\tau \leftarrow \mathbb{F}_p$ and output $\mathsf{prk} = \big(\{[\tau^j]_{1,2}\}_{i=0}^m\big)$, $\mathsf{vrk} = \big([\tau^{m-1}]_1, \{[\tau^j]_2\}_{i=0}^m\big)$.

$\mathsf{IP.Commit}(\mathsf{prk}, \mathbf{a})$**:** Compute $\mathsf{C}_a = \sum_{j=1}^m a_j[\tau^{j-1}]_1$ and output $(\mathsf{C}_a, \mathbf{a})$.

$\mathsf{IP.Open}(\mathsf{prk}, \mathsf{aux}, \mathbf{b}, y)$:

Find $R(X), Q(X)$ such that $\deg(R) < m-1$ and

$$\left(\sum_{j=1}^m a_j X^{j-1}\right)\left(\sum_{j=1}^m b_j X^{m-j}\right) - yX^{m-1} = R(X) + X^m Q(X).$$

Define $\hat{R}(X) = X^2 R(X)$

Output $\boldsymbol{\pi} = ([R(\tau)]_1, [Q(\tau)]_1, [\hat{R}(\tau)]_1)$.

$\mathsf{IP.Vf}(\mathsf{vrk}, \mathsf{C}_a, \mathbf{b}, y, \pi)$: Compute $\mathsf{C}_b = \sum_{j=1}^m b_j[\tau^{m-j}]_1$, parse $\boldsymbol{\pi} = ([R]_1, [Q]_1, [\hat{R}]_1)$ and output 1 if and only if

$$e\big(\mathsf{C}_a, \mathsf{C}_b\big) - e\big(y[\tau^{m-1}]_1, [1]_2\big) = e\big([R]_1, [1]_2\big) + e\big([Q]_1, [\tau^m]_2\big) \text{ and}$$

$$e([R]_1, [\tau^2]_2) = e([\hat{R}]_1, [1]_2).$$

Figure 4.7: Inner Product argument with encodings in the monomial basis.

Security

Theorem 15. *The construction above satisfies Completeness, Homomorphic Proofs and Homomorphic Openings.*

Proof. Completeness Let $\mathbf{a}, \mathbf{b} \in \mathbb{F}^m$ and $y = \mathbf{a} \cdot \mathbf{b}$

$$\begin{aligned}
&\left(\sum_{j=1}^{m} a_j X^{j-1}\right)\left(\sum_{j=1}^{m} b_j X^{m-j}\right) = \sum_{j=1}^{m}\sum_{i=1}^{m} a_j b_j X^{j-1+m-i} \\
&= \sum_{j=1}^{m} a_j b_j X^{-1+m} + \sum_{j=1}^{m}\sum_{i\neq j} a_j b_i X^{j-1+m-i} \\
&= (\mathbf{a}\cdot\mathbf{b})X^{m-1} + \sum_{j=1}^{m}\sum_{i<j} a_j b_i X^{j-1+m-i} + \sum_{j=1}^{m}\sum_{i>j} a_i b_j X^{j-1+m-i} \\
&= yX^{m-1} + \sum_{j=1}^{m}\sum_{i<j} a_j b_i X^{j-1+m-i} + X^m \sum_{j=1}^{m}\sum_{i>j} a_j b_i X^{j-1-i}.
\end{aligned}$$

An honest prover computes $R(X) = \sum_{j=1}^{m}\sum_{i<j} a_j b_i X^{j-1+m-i}$ and also $Q(X) = \sum_{j=1}^{m}\sum_{i>j} a_j b_i X^{j-1-i}$, $\hat{R}(X) = X^2 R(X)$, and sets $\boldsymbol{\pi} = ([R(\tau)]_1, [Q(\tau)]_1, [\hat{R}(\tau)]_1)$. For $C_a(X) = \sum_{j=1}^{m} a_j X^{j-1}$ and $C_b(X) = \sum_{j=1}^{m} b_j X^{m-j}$, we have $C_a(X)C_b(X) - yX^{m-1} = R(X) + X^m Q(X)$ and by definition $\hat{R}(X) = X^2 R(X)$. Then, for the same polynomials evaluated in the groups the pairing equations

$$e\big(\mathsf{C}_a, \mathsf{C}_b\big) - e\big(y[\tau^{m-1}]_1, [1]_2\big) = e\big([R]_1, [1]_2\big) + e\big([Q]_1, [\tau^m]_2\big) \text{ and}$$

$$e([R]_1, [\tau^2]_1) = e([\hat{R}]_1, [1]_2) \text{ also holds.}$$

Homomorphic proofs. Let $y_b = \mathbf{a}\cdot\mathbf{b}$, $y_c = \mathbf{a}\cdot\mathbf{c}$, $\boldsymbol{\pi}_b \leftarrow \mathsf{IP.Prove}(\mathsf{srs}, \mathbf{a}, \mathbf{b}, y_b)$, and $\boldsymbol{\pi}_c \leftarrow \mathsf{IP.Prove}(\mathsf{srs}, \mathbf{a}, \mathbf{c}, y_c)$, where $\boldsymbol{\pi}_b = ([R_b(\tau)]_1, [Q_b(\tau)]_1, [\hat{R}_b(\tau)]_1)$, $\boldsymbol{\pi}_c = ([R_c(\tau)]_1, [Q_c(\tau)]_1, [\hat{R}_c(\tau)]_1)$ are such that

$$\left(\sum_{j=1}^{m} a_j X^{j-1}\right)\left(\sum_{j=1}^{m} b_j X^{m-j}\right) - y_b X^{m-1} = R_b(X) + X^m Q_b(X),$$

$$\left(\sum_{j=1}^{m} a_j X^{j-1}\right)\left(\sum_{j=1}^{m} c_j X^{m-j}\right) - y_c X^{m-1} = R_c(X) + X^m Q_c(X),$$

$$\text{and } \hat{R}_b(X) = X^2 R_b(X), \hat{R}_c(X) = X^2 R_c(X).$$

Now, note that in order to compute a proof that $\mathbf{a}\cdot(\alpha\mathbf{b} + \beta\mathbf{c}) = \alpha y_b + \beta y_c$, the prover proceeds as follows:

$$\left(\sum_{j=1}^{m} a_j X^{j-1}\right)\left(\alpha\sum_{j=1}^{m} b_j X^{m-j} + \beta\sum_{j=1}^{m} c_j X^{m-j}\right)$$
$$= \alpha\left(\sum_{j=1}^{m} a_j X^{j-1}\right)\left(\sum_{j=1}^{m} b_j X^{m-j}\right) + \beta\left(\sum_{j=1}^{m} a_j X^{j-1}\right)\left(\sum_{j=1}^{m} c_j X^{m-j}\right)$$
$$= \alpha\left(X^{m-1}y_b + R_b(X) + X^m Q_b(X)\right) + \beta\left(X^{m-1}y_c + R_c(X) + X^m Q_c(X)\right)$$
$$= X^{m-1}(\alpha y_b + \beta y_c) + (\alpha R_b(X) + \beta R_c(X)) + X^m(\alpha Q_b(X) + \beta Q_c(X)),$$

and therefore for $y = \alpha y_b + \beta y_c$ it outputs $\boldsymbol{\pi} = ([R(\tau)]_1, [Q(\tau)]_1, [\hat{R}(\tau)]_1)$ where $R(X) = \alpha R_b(X) + \beta R_c(X)$, $Q(X) = \alpha Q_b(X) + \beta Q_c(X)$, and $\hat{R}(X) = X^2 R(X) = \alpha X^2 R_b(X) + \beta X^2 R_c(X) = \alpha\hat{R}_b(X) + \beta\hat{R}_c(X)$ i.e., $\boldsymbol{\pi} = \alpha\boldsymbol{\pi}_b + \beta\boldsymbol{\pi}_c$.

Homomorphic openings. The proof for homomorphic openings work analogous as the previous case. Indeed, for $y_a = \mathbf{a}\cdot\mathbf{c}$, $y_b = \mathbf{b}\cdot\mathbf{c}$ and $\boldsymbol{\pi}_a \leftarrow \mathsf{IP.Prove}(\mathsf{srs}, \mathbf{a}, \mathbf{b}, y_a)$, $\boldsymbol{\pi}_c \leftarrow \mathsf{IP.Prove}(\mathsf{srs}, \mathbf{c}, \mathbf{b}, y_c)$, it is enough to see

$$\left(\alpha\sum_{j=1}^{m} a_j X^{j-1} + \beta\sum_{j=1}^{m} b_j X^{j-1}\right)\left(\sum_{j=1}^{m} c_j X^{m-j}\right)$$
$$= \alpha\left(\sum_{j=1}^{m} a_j X^{j-1}\right)\left(\sum_{j=1}^{m} c_j X^{m-j}\right) + \beta\left(\sum_{j=1}^{m} b_j X^{j-1}\right)\left(\sum_{j=1}^{m} c_j X^{m-j}\right)$$
$$= \alpha\left(X^{m-1}y_a + R_a(X) + X^m Q_a(X)\right) + \beta\left(X^{m-1}y_b + R_b(X) + X^m Q_b(X)\right)$$
$$= X^{m-1}(\alpha y_a + \beta y_b) + (\alpha R_a(X) + \beta R_b(X)) + X^m(\alpha Q_a(X) + \beta Q_b(X)),$$

and the rest of the proof works as the one for homomorphic proofs.

□

Theorem 16. *The construction in Figure 4.7 is a strong function binding LVC scheme under the algebraic group model if the qDHE assumption holds.*

Proof. We will proceed through a series of games, and we set Game_0 to be the strong binding game of Definition 6. Let $\mathcal{A}$ be an adversary against of the scheme in Figure 4.6, whose advantage is $\mathsf{Adv}^{\mathsf{s.binding}}_{\mathcal{A}}$. Note that the second verification equation in our scheme is the same as in Figure 4.6, so we define Game_1 and the reduction $\mathcal{B}_1$ as in the proof of Theorem 13, and have

$$\mathsf{Adv}^{\mathsf{s.binding}}_{\mathcal{A}} \leq \mathsf{Adv}^{q\mathrm{DHE}}_{\mathcal{B}_1} + \mathsf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}}.$$

Now, we prove that the advantage of $\mathcal{A}$ in Game_1 is negligible. Similarly to the proof for the case of the Lagrange basis, we define $C_a(X) = \sum_{j=1}^{m+1} a_j X^{j-1}$ the algebraic representation of C_a and set $P(X) = C_a(X)C_b(X) - yX^{m-1} - R(X) -$

$Q(X)z_H(X)$ and the first verification equation says that, either τ is a root of $P(X)$, or $P(X) \equiv 0$. If the latter is the case, we have

$$\left(\sum_{j=1}^{m+1} a_j X^{j-1}\right)\left(\sum_{j=1}^{m} b_j X^{m-j}\right) - yX^{m-1} = R(X) + X^m Q(X).$$

The left side equals $\sum_{j=1}^{m}\sum_{j=1}^{m} a_j b_j X^{i-1+m-j} - yX^{m-1} + a_{m+1}X^m \sum_{j=1}^{m} b_j X^{m-j}$. Because $\deg(R) < m-1$ and $\deg(X^m Q(X)) > m-1$, we have that the right side of the equation has coefficient zero for X^{m-1} and so does the left side then. Thus, $\sum_{j=1}^{m} a_j b_j X^{m-1} - yX^{m-1} = 0$, which happens if and only if $\sum_{j=1}^{m} a_j b_j - y = 0$. Namely, there exists $\mathbf{a} = (a_j)_{j=1}^{m}$ such that $\mathbf{a} \cdot \mathbf{b} = y$, and $\mathcal{A}$ looses Game_1.

Then, it must be the case that $P(X) \neq 0$ and $P(\tau) = 0$. As in the proof of the previous theorem, we construct an adversary $\mathcal{B}_2$ against the *dlog* assumption. On input $[\tau]_1$, $\mathcal{B}_2$ calculates all the roots of $P(X)$ and checks, in polynomial time, which is the one that encoded in $\mathbb{G}_1$ equals $[\tau]_1$. Thus,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{s.binding}} \leq \mathsf{Adv}_{\mathcal{B}_1}^{q\mathrm{DHE}} + \mathsf{Adv}_{\mathcal{B}}^{dlog}.$$

□

Updates Without Hints.

In the case of this construction, we remark that we do not need any additional update keys added to the setup. Indeed, the update key is made by proofs of inner products between cannonic vectors $\mathbf{e}_j \cdot \mathbf{e}_j = 1$ or $\mathbf{e}_j \cdot \mathbf{e}_i = 0$. In our construction for encodings in the monomial basis, a proof that $\mathbf{e}_j \cdot \mathbf{e}_j = 1$ consists on commitments to $R(X) = Q(X) = 0$. On the other hand, to prove that $\mathbf{e}_j \cdot \mathbf{e}_i = 0$ for $i \neq j$ the proof is (the evaluation in the group of) either $R(X) = X^{m+j-i}$ if $i > j$, or $Q(X) = X^{j-i}$ if $j > i$. As such powers of τ are already included in prk, $\mathsf{upk} = \emptyset$.

4.4 Subvector Openings

In this section, we present algorithms for Subvector Openings(SVC), starting from the constructions in Section 4.3. As mentioned in Definition 5, we will consider SVC as an case of LVC schemes.

Note that the class of functions that open a set of positions $I = \{i_1, \ldots, i_n\}$ of a committed vector $\mathbf{v} \in \mathbb{F}^m$ is given by the linear-map f_I with

$$f_I : \mathbb{F}^m \to \mathbb{F}^n, \qquad f_I(\mathbf{v}) = (\mathbf{e}_{i_1} \cdot \mathbf{v}, \ldots \mathbf{e}_{i_n} \cdot \mathbf{v})$$

where for each $k \in [n]$, e_{i_k} is the i_kth vector of the canonical basis $\mathbb{F}^m$.

Thus, for a vector $\mathbf{v} \in \mathbb{F}^m$, we can naturally construct proofs of openings of subvectors $\mathbf{v}_I = (v_i)_{i \in I}$ by aggregating different inner product proofs for vectors $\mathbf{e}_{i_k}$ for $i_k \in I$ using the protocol in Figure 4.4. We refer to these aggregated proofs as *non-native* subvector openings, given that they require a random oracle and in particular, are no longer algebraic and homomorphic. As opposed to them, we call *native* subvector opening to those whose aggregated proofs are algebraic and homomorphic and do not use the random oracle.

In what follows, we improve on subvector openings in some special scenarios, achieving native aggregation for new schemes and reducing the verifier complexity in existing ones.

4.4.1 Native SV Openings for the Monomial Basis

For the construction of Section 4.3.2, we introduce native subvector openings for subsets with consecutive position $I = \{i, i+1, \ldots, i+k\}$. That is, for $\tilde{\mathbf{v}} = (v_i)_{i \in I}$ such that there exist $\mathbf{u}_1, \mathbf{u}_2$ with $\mathbf{v} = (\mathbf{u}_1, \tilde{\mathbf{v}}, \mathbf{u}_2)$. To prove an opening of $\tilde{\mathbf{v}}$, we only need commitments to $R(X) = \sum_{j=1}^{i-1} v_j X^{m-i+j-1}$ and $Q(X) = \sum_{j=i+k+1}^{m} v_{m-i+j+1} X^{j-1}$, which are shifted-encodings of $\mathbf{u}_1$, $\mathbf{u}_2$. The verifier checks that $\deg(R) < m-2$, computes $\tilde{C}(X) = \sum_{j=i}^{i+k} \tilde{v}_j X^{j-i}$ and $\tilde{\mathsf{C}} = [\tilde{C}(\tau)]_1$ and checks whether

$$e(\mathsf{C} - \tilde{\mathsf{C}}, [\tau^{m-i}]_1) = e([R]_1, [1]_2) + e([Q]_1, [\tau^{m+k}]_2).$$

Note that, given individual proofs of openings as in Section 4.3.2, that is, $[R_s(\tau)]_1$, $[Q_s(\tau)]_1$ such that $C(X)X^{m-j} - v_j X^{m-1} = R_j(X) + X^m Q_j(X)$ and $\deg(R_j) < m-1$, for the commitments defined above we have $[R]_1 = [R_i(\tau)]_1$ and $[Q]_1 = [Q_{i+k}(\tau)]_1$, that is, considering a prover that has pre-computed the proofs for individual openings, opening consecutive positions has no cost.

4.4.2 Non-native SV Openings for the Monomial Basis

For the LVC scheme of Section 4.3.2, the techniques of Section 4.2.1 allow us to redefine the Open and Verify algorithms to work for an arbitrary subset of positions $I \subset [m]$. More specifically, the prover will simply run $\mathsf{IP.Open}(\mathsf{prk}, \mathsf{aux}, \mathbf{e}_{i_k}, \mathbf{v})$ for $k = 1, \ldots, n$ to obtain $(v_{i_k}, \boldsymbol{\pi}_{i_k})$, where $\boldsymbol{\pi}_{i_k}$ is a proof of correct computation of v_{i_k}. Then, they use the random oracle to sample a randomness $\gamma \in \mathbb{F}$ and output $\boldsymbol{\pi}_I = \sum_{k=1}^{n} \gamma^{k-1} \boldsymbol{\pi}_{i_k}$.

The verifier will receive $\pi_I = ([R]_1, [Q]_1, [\hat{R}]_1)$, compute $y = \sum_{k=1}^{n} \gamma^{k-1} v_{i_k}$, and

check as before $e([R]_1, [\tau^2]_2) = e([\hat{R}]_1, [1]_2)$ and

$$e\left(\mathsf{C}, \sum_{k=1}^{n} \gamma^{k-1}[\tau^{m-i_k}]_2\right) - e\left(y[\tau^{m-1}]_1, [1]_2\right) = e\left([R]_1, [1]_2\right) + e\left([Q]_1, [\tau^m]_2\right).$$

Note that verifier's work is dominated by the computation of $\sum_{k=1}^{n} \gamma^{k-1}[\tau^{m-i_k}]_2$, so we analyze for which sets $I \subset [m]$ this computation can be cheaper than $|I|$ $\mathbb{G}_2$-exponentiations. Without loss of generality, we can re-assign $\gamma^{k-1} \to \gamma^{m-i_k}$, and thus our verifier now needs to compute $\sum_{k=1}^{n}[(\gamma\tau)^{m-i_k}]_2 = \sum_{i\in I}[(\gamma\tau)^{m-i}]_2$.

Now, note that if we consider a set of indexes $I_{k,s,n} \subset [m]$ that is an arithmetic progression, i.e. it is such that for a given ratio s, a starting power k and a number n of desired elements, $I_{k,s,n} = \{k, s+k, \ldots, (n-1)s+k\}$, then

$$\sum_{i\in I_{k,s,n}} (\gamma X)^{m-i} = (\gamma X)^k \frac{1-(\gamma X)^n}{1-(\gamma X)^s}.$$

This reduces the work of the verifier to compute $\sum_{i\in I_{k,s,n}}(\gamma X)^{m-i}$ to constant. Note that the verifier cannot compute $(1-(\gamma X)^s)^{-1}$, so we multiply all the terms of the equation by $1-(\gamma X)^s$. I.e, the verifier computes $y = \sum_{i\in I_{k,s,n}} \gamma^{m-i} y_i$ and checks whether

$$\begin{aligned} e\big(\mathsf{C}, \gamma^k[\tau^k]_2 - \gamma^{k+n}[\tau^{k+n}]_2\big) - e\big([\tau^{m-1}]_1 y - [\tau^{n+s-1}]_1 \gamma^s y, [1]_2\big) \\ = e\big([R]_1, 1-\gamma^s[\tau^s]_2)\big) + e\big([Q]_1, [\tau^n]_2 - \gamma^s[\tau^{n+s}]_2\big). \end{aligned}$$

4.4.3 Lagrange Basis

Native.

In the Lagrange Basis, one can use the native subset openings of [TAB+20]. There, the verifier needs to compute the vanishing polynomial $z_I(X) = \prod_{i\in I}(X - \mathsf{h}_i)$, as described in Section 2.5.3. To reduce verifer's work we focus on those subsets $I \subset [m]$ such that $z_I(X)$ can be calculated in less than $|I|$ computations. One answer to this question comes from cosets. That is, given $\mathbb{H} = \{\mathsf{h}_1, \ldots, \mathsf{h}_m\} = \{1, \omega, \omega^2, \ldots, \omega^{m-1}\}$ group of roots of unity where m is a power of 2, let $\mathbb{H}_{0,r}$ be the subgroup of order r of $\mathbb{H}$, where r goes from 2 to $m/2$. Then, for each $0 \le s < r$ we can construct the coset $\mathbb{H}_{s,r} = \omega^s\mathbb{H}_{0,r}$, whose vanishing polynomial is $z_{s,r}(X) = X^r - (\omega^s)^r$ (we prove this statement latter in Lemma 4). Verifier accepts if and only if

$$e\big(\mathsf{C} - \tilde{\mathsf{C}}, [1]_2\big) = e\big([Q]_1, [\tau^r]_2 - \omega^{sr}\big).$$

Non-native.

Given that the native subvector opening procedure above works for arbitrary subsets of indexes $I \subset [m]$, we don't consider aggregation of individual positions. The latter makes sense only when applying a linear function to the new subset. That is, when the verifier is given $\mathsf{C}_{f,I}$, claimed to be a commitment to $\mathbf{f} \cdot \mathbf{c}_I$, for some function represented as vector $\mathbf{f}$ and $\mathbf{c}_I = (c_i)_{i \in I}$.

4.5 Maintainable Vector Commitment Schemes

One of the key points of vector commitment schemes that allow aggregation of proofs is the ability to pre-compute and store individual openings and later use them to create subvector openings without incurring linear amount of computations each time. This is the case for the construction in [TAB+20], presented in Section 2.5.3, and also for the maintainable scheme of [TCZ+20].

In constructions such as the ones presented in Section 4.3, the proof of opening of one position is affected by *all* other elements in the vector. That is, the polynomials committed to create the proof have coefficients that involve all the values of the committed vector $\mathbf{v} \in \mathbb{F}^m$. As a consequence, prover work is linear in the size of $\mathbf{v}$ (as it has to evaluate polynomials of degree m) and after updating one position of $\mathbf{v}$, all all m proofs of opening are affected and need to be changed.

In this section, we present a protocol that offers a trade-off for different efficiency measures, using as building block the construction of Section 4.3.1. The prover can pre-compute and store some information and then save time in updating and computing individual proofs.

The intuition is the following: we divide the vector $\mathbf{v}$ in 2^ν small chunks $\{\mathbf{v}_i\} \in \mathbb{F}^{2^\kappa}$. We then arrange these chunks in a tree as follows: each leaf of the tree contains the commitment to one chunk and each node is a succinct representation of its children. The root of the tree is the(full) committed vector. An opening proof involves only the elements in the path of the root to the leaf containing the position to be opened. That is, if we want to open value a in position j of $\mathbf{v} \in \mathbb{F}^m$, we prove that (1) C_i is the leaf that contains the commitment to the $\mathbf{v}_i$ chunk containing j and (2) C_i opens to a in the position corresponding to j. The former part can be pre-computed and efficiently maintained, occupying storage linear in 2^ν, while the latter involves operations that are linear in 2^κ.

For vectors of size m, we offer the following trade-off: for any ν, κ, such that $m = 2^{\nu+\kappa+1}$, one can derive openings of size $\nu + 4$ group elements. The prover can pre-compute and store $2^\nu - 1$ proofs, and then compute proofs of functional openings by performing $O(\kappa 2^\kappa)$ group operations. We show also how to compute all proofs

with $O(\nu m)$ group operations (plus $O(m(\nu+\kappa))$ field operations). The proofs are maintainable, as an update in a position requires recomputing only $O(\nu)$ proofs.

The construction presented below is a consequence of the results in Section 4.4 and a construction presented in the paper that this chapter is based on [CNR+22].

As a first step, we prove that the inner product construction of Section 3.3 satisfies the same properties when using as interpolation set a coset of $\mathbb{H}$ as it does when $\mathbb{H}$ is a group of roots of unity.

4.5.1 Cosets of Roots of Unity

In this section, we prove some facts about the Lagrange and vanishing polynomials corresponding to cosets of subgroups of roots of unity, that will be used in our scheme.

We argue that the IP vector commitments construction in Section 4.3.1 can be implemented when we set $\mathbb{H}$ to be a set of roots of unity of size m where m is a power of two, and use as interpolation set a coset of size r(that is a smaller power of 2) instead of $\mathbb{H}$. We denote these cosets as $\mathbb{H}_{s,r} = \{\mathsf{h}_1^{s,r}, \ldots, \mathsf{h}_r^{s,r}\}$, where each $\mathsf{h}_i^{s,r} = \omega^{s+(i-1)\frac{m}{r}}$, $i = 1, \ldots, r$. We denote $\{\lambda_i^{s,r}(X)\}_{i=1}^r$ and $z_{s,r}(X)$ its Lagrange and vanishing polynomials.

Theorem 17. *Let $\mathbf{a}, \mathbf{b} \in \mathbb{F}^r$, and $\mathbb{H}_{s,r}$ a coset of size r of the group of roots of unity $\mathbb{H}_{0,r}$, with Lagrange interpolation polynomials $\{\lambda_i^{s,r}(X)\}_{i=1}^r$ and vanishing polynomial $z_{s,r}(X)$. Set $A(X) = \sum_{i=1}^r a_i \lambda_i^{s,r}(X)$ and $B(X) = \sum_{i=1}^r b_i \lambda_i^{s,r}(X)$. Then, $\mathbf{a} \cdot \mathbf{b} = y$ if and only if there exist polynomials $Q(X), R(X)$ with $\deg(R) < r-2$ such that*

$$A(X)B(X) - r^{-1}y = XR(X) + z_{r,s}(X)Q(X).$$

Proof. First, note that

$$A(X)B(X) = \left(\sum_{i=1}^r a_i \lambda_i^{s,r}(X)\right)\left(\sum_{i=1}^r b_i \lambda_i^{s,r}(X)\right) = \sum_{i=1}^r a_i b_i \lambda_i^{s,r}(X) \mod z_{r,s}(X)$$

Then, there exists $Q'(X)$ s.t. $A(X)B(X) = \sum_{i=1}^r a_i b_i \lambda_i^{s,r}(X) + z_{r,s}(X)Q'(X)$.

For the first implication, note that if $\mathbf{a} \cdot \mathbf{b} = y$, because $\lambda_i^{r,s}(0) = r^{-1}$ for all $i = 1, \ldots, r$ (See Lemma 4 below), $\sum_{i=1}^r a_i b_i \lambda_i^{s,r}(0) = r^{-1} \sum_{i=1}^r a_i b_i = r^{-1}y$, which implies that $\sum_{i=1}^r a_i b_i \lambda_i^{s,r}(X) - r^{-1}y$ vanishes at $X = 0$ and thus there exists $R(X)$ such that $\sum_{i=1}^r a_i b_i \lambda_i^{s,r}(X) = XR(X)$.

On the other hand, if we have that $A(X)B(X) = \sum_{i=1}^{r} a_i b_i \lambda_i^{s,r}(X) + z_{r,s}(X)Q'(X)$ and $A(X)B(X) - r^{-1}y = XR(X) + z_{r,s}(X)Q(X)$, because $\deg(XR(X)) < m$, $Q(X) = Q'(X)$ and $\sum_{i=1}^{r} a_i b_i \lambda_i^{s,r}(X) = XR(X)$. Set $X = 0$ in the equation and we have $r^{-1}\sum_{i=1}^{r} a_i b_i = r^{-1}m$, i.e, $\mathbf{a} \cdot \mathbf{b} = y$.

□

Lemma 4. *Consider $\mathbb{H}$ a group of roots of unity of size m, where m is a power of 2 and a coset $\mathbb{H}_{s,r}$ of size r. Then, $\lambda_i^{s,r}(0) = -r$, where $\lambda_i^{s,r}(X)$ is the ith Lagrange interpolation polynomial associated to $\mathbb{H}_{s,r}$.*

Proof. First, we note that $X^r - \omega^{sr}$ is the vanishing polynomial of $\mathbb{H}_{s,r}$. Indeed, it has degree r and for every $\mathsf{h}_i^{s,r}$ we have

$$(\mathsf{h}_i^{s,r})^r - \omega^{sr} = (\omega^{s+(i-1)\frac{m}{r}})^r - \omega^{sr} = \omega^{sr} - \omega^{sr} = 0.$$

Thus,

$$X^r - \omega^{sr} = \prod_{i=1}^{r}(X - \mathsf{h}_i^{s,r}) \text{ and } \frac{X^r - \omega^{sr}}{X - \mathsf{h}_i^{s,r}} = \prod_{j \neq i}^{r}(X - \mathsf{h}_j^{s,r}).$$

Now, we claim that if we denote $\lambda_i^{s,r}(X)$ as the ith Lagrange interpolation polynomial of $\mathbb{H}_{s,r}$, then

$$\lambda_i^{s,r}(X) = \frac{\mathsf{h}_i^{0,r}}{r\omega^{s(r-1)}} \frac{X^r - \omega^{sr}}{X - \mathsf{h}_i^{s,r}}.$$

To prove our claim, first note

$$\lambda_i^{s,r}(X) = \frac{\mathsf{h}_i^{0,r}}{r\omega^{s(r-1)}} \frac{X^r - \omega^{sr}}{X - \mathsf{h}_i^{s,r}} = \frac{\mathsf{h}_i^{0,r}}{r\omega^{s(r-1)}} \prod_{j \neq i}^{r}(X - \mathsf{h}_j^{s,r}).$$

It is clear from the above that $\lambda_i^{s,r}(\mathsf{h}_j^{s,r}) = 0$ for all $j \neq i$, now

$$\begin{aligned}\lambda_i^{s,r}(\mathsf{h}_i^{s,r}) &= \frac{\mathsf{h}_i^{0,r}}{r\omega^{s(r-1)}} \prod_{j \neq i}^{r}(\mathsf{h}_i^{s,r} - \mathsf{h}_j^{s,r}) = \frac{\mathsf{h}_i^{0,r}}{r\omega^{s(r-1)}} \omega^{s(r-1)} \prod_{j \neq i}^{r}(\mathsf{h}_i^{0,r} - \mathsf{h}_j^{0,r}) \\ &= \frac{\mathsf{h}_i^{0,r}}{r} \prod_{j \neq i}^{r}(\mathsf{h}_i^{0,r} - \mathsf{h}_j^{0,r})\end{aligned}$$

Since $\mathbb{H}_{0,r}$ is a group of roots of unity of size r, we know that $\frac{\mathsf{h}_i^{0,r}}{r}\prod_{j\neq i}^{r}(\mathsf{h}_i^{0,r}-\mathsf{h}_j^{0,r})$ is its ith Lagrange polynomial evaluated at $\mathsf{h}_i^{0,r}$, which is its ith interpolation point. Thus the equation above equals 1.

Then, $\lambda_i^{s,r}(X)$ is a polynomial of degree $r-1$ such that vanishes at all elements in $\mathbb{H}_{s,r}$ except for $\mathsf{h}_i^{s,r}$ where takes value 1 and so we conclude it is the ith Lagrange polynomial of $\mathbb{H}_{s,r}$.

Finally, recall that as $\mathbb{H}_{0,r}$ is a set of roots of unity of size r, all its Lagrange polynomials take value $-r$ when evaluated in 0. Then,

$$\lambda_i^{s,r}(0)=\frac{\mathsf{h}_i^{0,r}}{r\omega^{s(r-1)}}\prod_{j\neq i}^{r}(0-\mathsf{h}_j^{s,r})=\frac{\mathsf{h}_i^{0,r}}{r\omega^{s(r-1)}}\omega^{s(r-1)}\prod_{j\neq i}^{r}(0-\mathsf{h}_j^{0,r})=\frac{\mathsf{h}_i^{0,r}}{r}\prod_{j\neq i}^{r}(0-\mathsf{h}_j^{0,r})$$

equals the ith Lagrange polynomial of $\mathbb{H}_{0,r}$ evaluated in zero, that is, r^{-1}. □

The following lemma relates the vanishing polynomial and elements of two different cosets of size $r=\frac{m}{2^k}$ whose elements belong to the same coset of size $\frac{m}{2^{k-1}}$. Recall that this is the case for cosets $\mathbb{H}_{s,\frac{m}{2^k}}$, $\mathbb{H}_{s',\frac{m}{2^k}}$ if and only if $s\equiv s' \mod 2^{k-1}$. The lemma will be used in the next section to prove what constitutes, along with the result on arguments for inner products using cosets, the core of our maintainable construction.

Lemma 5. *Let $\mathbb{H}_{s,\frac{m}{2^k}}$, $\mathbb{H}_{s',\frac{m}{2^k}}$ be two cosets of $\mathbb{H}$ of size $\frac{m}{2^k}$ such that $s<s'$ and $s\equiv s' \mod 2^{k-1}$. Let $z_{s,\frac{m}{2^k}}(X)$ and $z_{s',\frac{m}{2^k}}(X)$ be its vanishing polynomials. Then, for every $\mathsf{h}\in\mathbb{H}_{s,\frac{m}{2^k}}$ and $\mathsf{h}'\in\mathbb{H}_{s',\frac{m}{2^k}}$,*

$$z_{s,\frac{m}{2^k}}(\mathsf{h}')=-2\omega^{s\frac{m}{2^k}},\qquad\qquad z_{s',\frac{m}{2^k}}(\mathsf{h})=2\omega^{s\frac{m}{2^k}}.$$

Proof. First, note that $s'=s+2^{k-1}$, $\mathsf{h}=\omega^{s+(i-1)2^k}$ for some $i=1,\dots,\frac{m}{2^k}$. Consequently, $\mathsf{h}'=\omega^{s+2^{k-1}+(i-1)2^k}$. Also, remark that $\omega^{\frac{m}{2}}=-1$, $z_{s,\frac{m}{2^k}}(X)=X^{\frac{m}{2^k}}-\omega^{s\frac{m}{2^k}}$ and $z_{s',\frac{m}{2^k}}(X)=X^{\frac{m}{2^k}}-\omega^{s'\frac{m}{2^k}}=X^{\frac{m}{2^k}}-\omega^{(s+2^{k-1})\frac{m}{2^k}}$. Then,

$$\begin{aligned}z_{s,\frac{m}{2^k}}(\mathsf{h}')&=(\omega^{s+2^{k-1}+(i-1)2^k})^{\frac{m}{2^k}}-\omega^{(s+(i-1)2^k)\frac{m}{2^k}}=(\omega^{s+2^{k-1}})^{\frac{m}{2^k}}-\omega^{s\frac{m}{2^k}}\\&=\omega^{s\frac{m}{2^k}}\left((\omega^{2^{k-1}})^{\frac{m}{2^k}}-1\right)=\omega^{s\frac{m}{2^k}}\left(\omega^{\frac{m}{2}}-1\right)=\omega^{s\frac{m}{2^k}}\left(-1-1\right)\\&=-2\omega^{s\frac{m}{2^k}}.\end{aligned}$$

Analogously,

$$\begin{aligned}z_{s',\frac{m}{2^k}}(\mathsf{h})&=\mathsf{h}^{\frac{m}{2^k}}-\omega^{(s+2^{k-1})\frac{m}{2^k}}=(\omega^{s+(i-1)2^k})^{\frac{m}{2^k}}-\omega^{(s+2^{k-1})\frac{m}{2^k}}\\&=(\omega^s)^{\frac{m}{2^k}}-\omega^{(s+2^{k-1})\frac{m}{2^k}}z_{s',\frac{m}{2^k}}(\mathsf{h})=\omega^{s\frac{m}{2^k}}\left(1-\omega^{\frac{m}{2}}\right)=\omega^{s\frac{m}{2^k}}\left(1-(-1)\right)\\&=2\omega^{s\frac{m}{2^k}}.\end{aligned}$$

□

4.5.2 The Scheme

Tree Structure. The idea is that the prover can pre-compute the tree before interacting with any verifier, according to the trade-off between storage and opening time they want to achieve. The root of the tree has a commitment $\mathsf{C} = \sum_{j=1}^{m} v_j[\lambda_j(\tau)]_1$ to the vector $\mathbf{v}$ the prover aims to claim openings to.

For the next level of the tree, the prover will divide vector $\mathbf{v}$ on two vectors of size $m/2$: $\mathbf{v}^{0,1}$ and $\mathbf{v}^{1,1}$. To commit to them, it will use Lagrange polynomials $\{\lambda_j^{b,1}(X)\}_{j=1}^{m/2}$ for $b = 1, 0$, that correspond to the *cosets* $\mathbb{H}_{0,\frac{m}{2}}, \mathbb{H}_{1,\frac{m}{2}}$. $\mathbb{H}_{0,\frac{m}{2}}$ is computed as the subgroup of roots of unity of size $m/2$, while $\mathbb{H}_{1,\frac{m}{2}} = \omega\mathbb{H}_{0,\frac{m}{2}}$. Notably, vectors $\mathbf{v}^{0,1}$ and $\mathbf{v}^{1,1}$ will not contain consecutive positions of $\mathbf{v}$, but those corresponding with the index of the elements in $\mathbb{H}_{0,1}$ and $\mathbb{H}_{1,1}$, respectively. That is, $\mathbf{v}^{0,1}$ will contain the even and $\mathbf{v}^{1,1}$ the odd positions of $\mathbf{v}$.

In the next level, $\mathbf{v}^{0,1}$ and $\mathbf{v}^{1,1}$ are divided as $\mathbf{v}^{0,2}$ and $\mathbf{v}^{1,2}$, and $\mathbf{v}^{2,2}, \mathbf{v}^{3,2}$, respectively, of size $m/4$ each. Once more, note that the positions in each $\mathbf{v}^{s,2}$ do not follow the order of $\mathbf{v}$, but they will only contain the elements h_{s+k4} for all $k = 0, \ldots, \frac{m-4}{4}$. To encode them, we use the Lagrange interpolation polynomial corresponding to cosets $\mathbb{H}_{0,\frac{m}{4}}$, $\mathbb{H}_{1,\frac{m}{4}}$, $\mathbb{H}_{2,\frac{m}{4}}$ and $\mathbb{H}_{3,\frac{m}{4}}$, where $\mathbb{H}_{0,\frac{m}{4}}$ is the subgroup of roots of unity of size $m/4$, $\mathbb{H}_{1,\frac{m}{4}} = \omega\mathbb{H}_{0,\frac{m}{4}}$, $\mathbb{H}_{2,\frac{m}{4}} = \omega^2\mathbb{H}_{0,\frac{m}{4}}$ and $\mathbb{H}_{3,\frac{m}{4}} = \omega^3\mathbb{H}_{0,\frac{m}{4}}$. At the end of the tree, in level ν, we have 2^ν vectors of size 2^κ, where ν and κ are chosen by the prover following the trade-off mentioned before.

Notation and some facts about roots of unity. As stated before, throughout this section, $m = 2^{\nu+\kappa+1}$ and $\mathbb{H} \subset \mathbb{F}$ is a set of roots of unity of size m.

For cosets and Lagrange polynomials, we will use a more intuitive notation for working with a tree structure. In Sections 4.4.3,4.5.1, we denoted as $\mathbb{H}_{s,r}$ the coset $\omega^s\mathbb{H}_{0,r}$ that consist of all the elements of the subgroup of roots of unity of size r multiplied by ω^s. In this section, r will vary between all powers of 2 in $\{2^{\nu+\kappa+1}, \ldots, 2^\nu\}$, taking value $\frac{m}{2^\ell}$ in level ℓ of the tree. For that reason, we will denote the coset of size $\frac{m}{2^\ell}$ corresponding to s as $\mathbb{H}_{s,\ell}$, instead of $\mathbb{H}_{s,r}$ for $r = \frac{m}{2^\ell}$. What is more, for each $i = 1, \ldots, 2^\kappa$, we denote as (s_ℓ, ℓ) the pair such that $\mathbf{v}^{s_\ell,\ell}$ is in the path from the root to $\mathbf{v}_i$; when i is not clear from the context, we will write (s_{ℓ_i}, ℓ). Then, for each $\mathbf{v}_i$, there are pairs $\{(s_\ell, \ell)\}_{\ell=1}^{\kappa}$ such that the vectors $\mathbf{v}^{s_\ell,\ell}$ are the ones in the path from $\mathbf{v}$ to $\mathbf{v}_i$, and are encoded as $[\boldsymbol{\lambda}^{s_\ell,\ell}]_1 \cdot \mathbf{v}$, where $[\boldsymbol{\lambda}^{s_\ell,\ell}]_1 = ([\lambda_1^{s_\ell,\ell}(\tau)]_1, \ldots, [\lambda_{2^{m-\ell}}^{s_\ell,\ell}(\tau)]_1)$ is the vector of the Lagrange interpolation polynomials corresponding to $\mathbb{H}_{s_\ell,\ell}$, evaluated at the trapdoor in $\mathbb{G}_1$.

High Level Description. The construction leverages the fact that vanishing and Lagrange polynomials corresponding to cosets of roots of unity are sparse, as has been shown in the proof of Lemma 4, so we minimize precomputation and verifier work. If nodes at level ℓ are split in two, at each level we are separating cosets of subgroups of roots of unity into cosets of half the size. Another key point is the fact that the sumcheck in Section 3.3 and so our functional vector commitment in Section 4.3.1 work with vectors encoded using Lagrange polynomials of cosets, as it has been proven in Theorem 17.

Finally, we relate the parent and the children nodes at each level, in a simple equation through this Lemma:

Lemma 6. *Consider two cosets $\mathbb{H}_{s_\ell,\ell}$ and $\mathbb{H}_{s'_\ell,\ell}$ such that $s_\ell \equiv s'_\ell \mod 2^{m-\ell-1}$. Without loss of generality, we assume $s_\ell < s'_\ell$, so $s' = s_\ell + 2^{m-\ell-1}$ for some $i = 1$. Let $C^{s_\ell,\ell}(X)$ be an encoding of vector $\mathbf{v}^{s_\ell,\ell}$, that is, the vector of size $\frac{m}{2^\ell}$ whose elements are those in $\mathbf{v}$ at the positions k such that $\mathsf{h}_k \in \mathbb{H}_{s_\ell,\ell}$.*

For all levels $\ell = 0, \ldots, \nu$ it is true that

$$C^{s_{\ell-1},\ell-1}(X) = z_{s'_\ell,\ell}(X)\frac{C^{s_\ell,\ell}(X) - C^{s'_\ell,\ell}(X)}{2\omega^{s\ell}} + C^{s'_\ell,\ell}(X)$$

$$C^{s_{\ell-1},\ell-1}(X) = z_{s_\ell,\ell}(X)\frac{C^{s_\ell,\ell}(X) - C^{s'_\ell,\ell}(X)}{-2\omega^{s\ell}} + C^{s_\ell,\ell}(X)$$

Proof. We start with the equality $C^{s_\ell,\ell}(X) = z_{s'_\ell,\ell}(X)\frac{C^{s_\ell,\ell}(X)-C^{s'_\ell,\ell}(X)}{2\omega^{sr}} + C^{s'_\ell,\ell}(X)$ and evaluate it in $\mathsf{h} \in \mathbb{H}_{s_\ell,\ell}$ and $\mathsf{h}' \in \mathbb{H}_{s'_\ell,\ell}$, using the result in Lemma 5.

$$C^{s_{\ell-1},\ell-1}(\mathsf{h}) = z_{s'_\ell,\ell}(\mathsf{h})\frac{C^{s_\ell,\ell}(\mathsf{h}) - C^{s'_\ell,\ell}(\mathsf{h})}{2\omega^{sr}} + C^{s'_\ell,\ell}(\mathsf{h}), \text{ that is, } \mathbf{v}_{\mathsf{h}}^{s_\ell,\ell} = 2\omega^{s\ell}\frac{\mathbf{v}_{\mathsf{h}}^{s_\ell,\ell}}{2\omega^{s\ell}} = \mathbf{v}_{\mathsf{h}}^{s_\ell,\ell}.$$

$$\text{Also, } C^{s_{\ell-1},\ell-1}(\mathsf{h}') = z_{s'_\ell,\ell}(\mathsf{h}')\frac{C^{s_\ell,\ell}(\mathsf{h}') - C^{s'_\ell,\ell}(\mathsf{h}')}{2\omega^{s\ell}} + C^{s'_\ell,\ell}(\mathsf{h}') \text{ and thus, } \mathbf{v}_{\mathsf{h}'}^{s_\ell,\ell} = \mathbf{v}_{\mathsf{h}'}^{s_\ell,\ell}$$

Then, the left and right side of the equation are polynomials of degree $\frac{m}{2^\ell} - 1$ that agree at $\frac{m}{2^\ell} - 1$ points, so we conclude they are equal.

For the other case, note that

$$C^{s_{\ell-1},\ell-1}(\mathsf{h}) = z_{s_\ell,\ell}(\mathsf{h})\frac{C^{s_\ell,\ell}(\mathsf{h}) - C^{s'_\ell,\ell}(\mathsf{h})}{-2\omega^{sr}} + C^{s_\ell,\ell}(\mathsf{h}) \text{ and } \mathbf{v}_{\mathsf{h}}^{s_\ell,\ell} = \mathbf{v}_{\mathsf{h}}^{s_\ell,\ell}. \text{ Also,}$$

$$C^{s_{\ell-1},\ell-1}(\mathsf{h}') = z_{s_\ell,\ell}(\mathsf{h}')\frac{C^{s_\ell,\ell}(\mathsf{h}') - C^{s'_\ell,\ell}(\mathsf{h}')}{-2\omega^{s\ell}} + C^{s_\ell,\ell}(\mathsf{h}'), \text{ so } \mathbf{v}_{\mathsf{h}'}^{s_\ell,\ell} = -2\omega^{s\ell}\frac{\mathbf{v}_{\mathsf{h}'}^{s_\ell,\ell}}{-2\omega^{s\ell}},$$

and the conclusion is the same. □

To open C to a certain leaf commitment C_i, the idea is to implicitly show from root to leaf that $\mathsf{C}_{s_{\ell-1},\ell-1} - \mathsf{C}_{s_\ell,\ell}$ agree in $\mathbb{H}^{s_\ell,\ell}$. This is proven by showing that their difference is divisible by $z_{s_\ell,\ell}(X)$.

We describe the protocol for any function f represented as a vector $\mathbf{f}$ that, when applied to $\mathbf{v}$ involves only the elements contained in one chunk $\mathbf{v}_i$. Importantly, this relation can be generalized for any linear function through the aggregation scheme presented in Section 4.2.2, but the most important use case is for opening individual positions, that is, when $\mathbf{f}$ is a canonical vector.

Now, in order to open C to an expression $\mathbf{f} \cdot \mathsf{C}_i = y$, the prover will (1) provide the quotient polynomials that prove the relation above, that is, that C_i is the chunk containing a commitment to the vector $\mathbf{v}_i$ affected by $\mathbf{f}$, and (2) prove that the inner product between $\mathbf{f}$ and $\mathbf{v}_i$ is indeed y.

Scheme Description. Formally, we present an LVC commitment scheme that works for the function family:

$$\mathsf{Ext}_\nu\text{-}\mathcal{F}_{p,2^\kappa} = \{f : \mathbb{F}^m \to \mathbb{F}, m = 2^{\kappa+\nu+1} \mid \exists \mathbf{f} \in \mathbb{F}^{2^\kappa}, i \in 2^\nu \text{ s.t.} \\ \forall \mathbf{v}_1, \ldots, \mathbf{v}_{2^\nu} \in \mathbb{F}^{2^\kappa} : \ f(\mathbf{v}_1, \ldots, \mathbf{v}_{2^\nu}) = \mathbf{v}_i \cdot \mathbf{f}\}$$

Algorithms $\mathsf{LVC.KeyGen}$ and $\mathsf{LVC.Commit}$ are the same as the Lagrange basis construction of Section 4.3 and are omitted. The commitment to $\mathbf{v}$ is $\mathsf{C} = [\boldsymbol{\lambda}^\top]_1 \mathbf{v}$ together with the auxiliary input information aux. Note that step 4. of the open algorithm is $\mathsf{IP.Open}$ from Section 4.3.1.

Maintainability. The cost of computing all proofs is $O(\nu m)$. For each piece $\mathbf{v}_i$ with $O(\kappa 2^\kappa)$ operations one can compute the coefficients in the monomial basis. Following expression (2), the parent node can be computed in cost dominated by $2^\kappa = \frac{m}{2^{\nu+1}}$ exponentiations from the expression of children nodes, and since there are 2^ν parent nodes de cost is dominated by $\frac{m}{2}$ exponentiations. Going one level up, the vector size doubles but the number of nodes is halved. We conclude that to compute all proofs one needs $O(\kappa 2^\kappa + \nu \frac{m}{2})$. The number of proofs to store (including leaf commitments) is $2^{\nu+1} - 1$.

Theorem 18. *When instantiated with a function binding argument for inner product relations* IP*, the scheme in Figure 4.8 is a function binding LVC argument under the AGM if the dlog assumption hold.*

Proof. Let $\mathcal{A}$ be an adversary against the function binding game as in Definition 6. We will see, through game reductions, that the advantage of $\mathcal{A}$ in strong function binding is negligible even for $k = 2$, that is, for two non-compatible functions f_1, f_2.

$\mathcal{A}$ plays Game_0, the strong function binding game, and outputs $(\mathsf{C}, \{f_b, y_b, \boldsymbol{\pi}_b\}_{b=1,2})$, where $\boldsymbol{\pi}_1 = (\{[H^{s_\ell,\ell}]_1\}_{\ell=1}^{\kappa}, \mathsf{C}_i[R]_1, [\hat{R}]_1, [H^\nu]_1)$, $\boldsymbol{\pi}_2 = (\{[H^{s_\ell,\ell'}]_1\}_{\ell=1}^{\kappa}, \mathsf{C}'_i, [R']_1, [\hat{R}']_1, [H^{\nu'}]_1)$, such that $\mathsf{LVC.Verify}(\mathsf{vk}, \mathsf{C}, f_1, y_1, \boldsymbol{\pi}_1) = 1$, $\mathsf{LVC.Verify}(\mathsf{vk}, \mathsf{C}, f_2, y_2, \boldsymbol{\pi}_2) = 1$, but there exists no $\mathbf{v} \in \mathbb{F}^m$ such that $f_1(\mathbf{v}) = y_1$.

Let Game_1 be exactly as Game_0 but upon receiving $\boldsymbol{\pi}_1, \boldsymbol{\pi}_2$, checks if C_i and $\mathsf{C}'_i$are equal and aborts otherwise. We prove that the latter happens with negligible probability. To start, recall $\mathcal{A}$ is algebraic and thus we can extract $C_i(X), C'_i(X)$, the algebraic representations of $\mathsf{C}_i, \mathsf{C}'_i$ and $H^\nu(X), H^{\nu'}(X)$, $\{H^{s_\ell,\ell}(X), H^{s_\ell,\ell'}(X)\}_{\ell=0}^{\nu-1}$ the ones for $[H^\nu]_1, [H^{\nu'}]_1$, $\{[H^{s_\ell,\ell}]_1, [H^{s_\ell,\ell'}]_1\}_{\ell=0}^{\nu-1}$, respectively. Now, consider the polynomial

$\mathsf{LVC.Open}(\mathsf{pk}, \mathbf{b}, \mathsf{aux}, f, \mathbf{y})$:

Let $f(\mathbf{v}_1, \ldots, \mathbf{v}_{2^\nu}) = \mathbf{v}_i \cdot \mathbf{f}$ for $\mathbf{f} \in \mathbb{F}^{2^\kappa}$. For all $\ell \in \{0, ..., \kappa\}$, set $(s_\ell, \ell) := (s_{\ell_i}, \ell)$, where s_ℓ is s.t. $\mathsf{C}_{s_\ell,\ell}$ is the element in the path from the root to $\mathbf{v}_i$ at level ℓ.
For every pair (s_ℓ, ℓ), find the pair (s'_ℓ, ℓ) such that $\mathbb{H}_{s_\ell,\ell} \cup \mathbb{H}_{s'_\ell,\ell} = \mathbb{H}_{s_{\ell-1},\ell-1}$.
If $s_\ell < s'_\ell$, set $r_\ell = 1$, otherwise, set $r_\ell = 0$. Set $K_\ell = (-1)^{r_\ell}(2\omega^{s_\ell\ell})^{-1}$
Compute all $\mathsf{C}_{s_\ell,\ell} = \sum_{j=1}^{\ell} v_j^{s_\ell,\ell}[\lambda_j^{s_\ell,\ell}(\tau)]_1$ and $[H^{s_\ell,\ell}]_1 = K_\ell(\mathsf{C}_{s_\ell,\ell} - \mathsf{C}_{s'_\ell,\ell})$.

Find $R(X), H^\nu(X)$ such that

$$\left(\sum_{j=1}^{2^\kappa} v_{i,j}\lambda_j^{i,\kappa}(X)\right)\left(\sum_{j=1}^{2^\kappa} f_j\lambda_j^{i,\kappa}(X)\right) - \frac{y}{m} = XR(X) + H^\nu(X)z_{i,\kappa}(X).$$

The polynomial $R(X)$ should be of degree at most $2^\kappa - 2^6$.
Define $\hat{R}(X) = X^{m-1-2^\kappa}R(X)$.
Output $\boldsymbol{\pi} = (\{[H^{s_\ell,\ell}]_1\}_{\ell=0}^{\nu-1}, \mathsf{C}_i = \mathsf{C}^{s_\nu,\nu}, [R(\tau)]_1, [\hat{R}(\tau)]_1, [H^\nu(\tau)]_1)$.

$\mathsf{LVC.Verify}(\mathsf{vk}, \mathsf{C}, f, \mathbf{y}, \boldsymbol{\pi})$:
Compute $\mathsf{C}_f = \sum_{j=1}^{2^\kappa} f_j[\lambda_j^{i,\kappa}(\tau)]_2$.
Check that

$$e(\mathsf{C} - \mathsf{C}_i, 1) = e([H^\nu]_1, [z_{i,\kappa}(\tau)]_2) + \sum_{\ell=0}^{\nu-1} e([H^{s_\ell,\ell}]_1, [z_{s_\ell,\ell}(\tau)]_2) \tag{4.1}$$

$$e\big(\mathsf{C}_i, \mathsf{C}_f\big) - e\big(m^{-1}y[1]_1, [1]_2\big) = e\big([R]_1, [1]_2\big) + e\big([H^\nu]_1, [z_{i,\kappa}(\tau)]_2\big) \tag{4.2}$$

$$e\big([R]_1, [\tau^{m-1-2^\kappa}]_2\big) = e\big([\hat{R}]_1, [1]_2\big) \tag{4.3}$$

Figure 4.8: Mantainable LVC schemes with memory/time trade-offs.

[6] We assume as in Section 4.3 that at most $m-1$ powers of τ are in the SRS in group $\mathbb{G}_1$.

$$p(X) = C_i(X) - C'_i(X) - (H^{\nu}(X) - H^{\nu'}(X))z_{i,\kappa}(X) + \sum_{\ell=0}^{\nu-1}(H^{s_\ell,\ell}(X) - H^{s_\ell,\ell'}(X))z_{s_\ell,\ell}(X).$$

$\mathcal{A}$ wins Game_1 if (i) $p(\tau) = 0$, which implies there is an adversary $\mathcal{B}_{\mathsf{dlog}}$ that using $\mathcal{A}$ as subroutine breaks dlog by setting $[\tau]_1$ as input and calculating the roots of $p(X)$ in polynomial time, or (ii) $p(X) \equiv 0$ which, in particular, implies that $p(\mathsf{h}_j^{i,k}) = 0$ for all $j = 1, \ldots, 2^\kappa$, and so $0 = c_{i,j} - c'_{i,j}$, since $z_{i,k}(X)|z_{s_\ell,\ell}(X)$ for all the pairs (s_ℓ, ℓ). Then, the advantage of $\mathcal{A}$ in the strong function binding game is bounded by

$$\mathsf{Adv}^{\mathsf{s.binding}}_{\mathcal{A}} \leq \mathsf{Adv}^{dlog}_{\mathcal{B}_1} + \mathsf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}}.$$

Let Game_2 be exactly as Game_1 except that it checks whether there exists a vector $\mathbf{v}_i$ such that $f_1(\mathbf{v}_i) = y_1$ and $f_2(\mathbf{v}_i) = y_2$ and aborts otherwise. If the latter is the case, we can construct an adversary $\mathcal{B}_2$ against function binding of the IP scheme that takes $\mathcal{A}$ outputs, set $\tilde{f}$ as the linear function represented by vector $\tilde{\mathbf{f}}_1 = (f_{1j})_{j\in i,k}$, and $\tilde{f}_2$ as the one represented by $\tilde{\mathbf{f}}_2 = (f_{2j})_{j\in i,k}$, $\tilde{\boldsymbol{\pi}}_1 = ([R]_1, [H^{\nu}]_1, [\hat{R}]_1)$, $\tilde{\boldsymbol{\pi}}_2 = ([R']_1, [H^{\nu'}]_1, [\hat{R}']_1)$, outputs $(\tilde{f}_1, y_1, \tilde{\boldsymbol{\pi}}_1)$, $(\tilde{f}_2, y_2, \tilde{\boldsymbol{\pi}}_2)$ and wins with the same probability of $\mathcal{A}$. Then,

$$\mathsf{Adv}^{\mathsf{s.binding}}_{\mathcal{A}} \leq \mathsf{Adv}^{dlog}_{\mathcal{B}_1} + \mathsf{Adv}^{\mathsf{s.b.IP}}_{\mathcal{B}_2}.$$

□

Chapter 5

Position-Hiding Linkability

This chapter is based on the paper 'Caulk: Lookup Arguments in Sublinear Time' [ZBK+22], which is a joint work with Vitalik Buterin, Dmitry Khovratovich, Mary Maller, Anca Nitulescu, and Mark Simkin.

5.1 Introduction

Vector commitment schemes offer an spectrum of advantages on storing, ordering and managing data, as it has been emphasized in the previous Chapter. In particular, we can commit to a potentially very big set of data and later prove that a specific element or set of elements has been committed into it. The rise of privacy-preserving applications makes it vital to make these type of protocols zero-knowledge i.e. hiding the element(s) that is asserted to be in the commitment while still establishing a certain relationship, or *link*, to the original set.

Making this link in zero-knowledge, that is, proving that some element committed in zero-knowledge belongs to a public set, previously committed in a succinct way, can be crucial in many real-world applications. The simplest example is the proof of authorization where a party proves the knowledge of a secret key beyond one of the public keys in the set: it first isolates the public key from the set in zero-knowledge in order not to reveal its identity. A more elaborate example is a *proof of coin ownership* in private cryptocurrencies: with coins stored as hashes of a secret k and value v in a list or a tree, any user can prove that they are allowed to spend v by showing its k in zero knowledge. A third example is a lookup argument in verifiable computation: prove that intermediate values $a_1, a_2, \ldots, a_k$ are all contained in a certain table (e.g., a table of all 16-bit numbers for the purpose of overflow checks in financial or mathematical computations). Applications also include membership proofs, ring signatures, anonymous credentials and other schemes.

So far all these examples have been handled by working but not so efficient mechanisms, which limit scalability and adoption. The first version of the Zcash cryptocurrency [ZCary] used a SHA-2-based Merkle tree to store the coins and the Groth16 [Gro16] SNARK to prove the coin ownership. Relatively heavy machinery of Groth16 and the unfit of SHA-2 to prime-field circuits made the resulting prover time of 40 seconds barely usable. Even the most recent developments of algebraic hashes [AGR+16, GKR+21] reduce *prover time* by one order of magnitude only. Another application of concern, lookup tables, so far has required the generic construction of Plookup [GW20] that makes the prover linear in the size of the table itself, no matter how many values are meant to be isolated.

5.1.1 Contributions

In this Chapter we first formalize the property mentioned above as *position-hiding linkability* for vector commitments. Concisely, two vector commitment schemes VC_1 and VC_2 are position-hiding linkable if a prover can convince a verifier that one element or set of elements committed as cm using VC_2 are also elements in some vector $\mathbf{v}$ publicly committed as C using VC_1. As it is always the case for proving systems, we require this scheme to satisfy completeness and soundness, defined as linkability, and we also require position-hiding which implies zero-knowledge for the positions and values being opened.

We present a novel position-hiding linkability construction, named Caulk, which performs with unprecedented efficiency. We set VC_1 to be the commitment scheme described in Section 2.5.2 and 4.3.1, which encodes a vector $\mathbf{v}$ of size N as $C(X) = \sum_{s=1}^{N} v_s \rho_s(X)$, where $\{\rho_s(X)\}_{s=1}^{N}$ are the Lagrange interpolation polynomials corresponding to a subgroup of roots of unity $\mathbb{K} = \{\mathsf{k}_1, \ldots, \mathsf{k}_N\}$ of size N, and later commits to it as a $\mathbb{G}_1$-element $\mathsf{C} = \sum_{s=1}^{N} v_s [\rho_s(\tau)]_1$ where τ is secret.

For the case where cm is a commitment to a single element, we set VC_2 to be a Perdersen Commitment scheme, as described in Section 2.5.4, and for the case $m > 1$, $\mathsf{VC}_2 = \mathsf{VC}_1$. Both our constructions are based in KZG proofs of openings for vector commitments, whose details are given in Section 2.5.2. We first note that an individual proof of opening of value v at position s is an element $[Q_s]$ such that

$$e(\mathsf{C} - [v]_1, [1]_2) = e([Q_s]_1, [\tau - \mathsf{k}_s]_2),$$

and a proof of opening of a subset of positions $I \subset [N]$ is an element $[Q_I]_1$ such that if $\{\eta_s(X)\}_{s \in I}$ are the Lagrange interpolation polynomials of $\mathbb{K}_I = \{\mathsf{k}_s\}_{s \in I}$, $C_I(X) = \sum_{s \in I} v_s \eta_s(X)$ and $z_I(X) = \prod_{s \in I}(X - \mathsf{k}_s)$,

$$e(\mathsf{C} - [C_I(\tau)]_1, [1]_2) = e([Q_I]_1, [z_I(\tau)]_2).$$

Our prover time is unaffected by the computation of the non-hiding KZG proofs

$[Q_s]_1$ and $[Q_I]_1$, as the former can be pre-computed along with all individual positions using $N \log N$ group operations and the latter obtained from the pre-computed proofs for all $s \in I$, in time dependent on I as opposite to N, as shown in [TAB+20, FK] and discussed in Section 2.5.3. As a result, note that our prover does require linear storage.

The challenge is then to hide the polynomials committed in $[v]_1$ or $[C_I(\tau)]_1$ and $[\tau - \mathsf{k}_s]_1$ or $[z_I(\tau)]_1$. For the polynomials containing the opening values we use standard zero-knowledge techniques, while the vanishing polynomials require a more careful approach. The prover will add blinders to the polynomials before committing, and then needs to convince the verifier that the polynomials committed in the group elements are indeed vanishing polynomials of some subset of $\mathbb{K}$. This last step is the most challenging part and main contribution of this Chapter.

Case $m = 1$**.** We use standard arguments of knowledge for Pedersen commitments so the prover demonstrates knowledge of v and r such that $\mathsf{cm} = [v + \mathsf{g}r]_1$ for unknown g given to them as $[\mathsf{g}]_1$ in the setup. The challenge is then to prove well formation of $[a(\tau - \mathsf{k}_s)]_2$, a blind commitment to $X - \mathsf{k}_s$, which implies proving that it is a polynomial of degree 1 and that k_s is an Nth root of unity i.e. that $\mathsf{k}_s^N = 1$.

In order to avoid working with unnecessarily big polynomials, we introduce a new subgroup of roots of unity $\mathbb{U}_n = \{\mathsf{u}_1, \ldots, \mathsf{u}_n\}$ with $n = \log(N) + 6$. We create a polynomial $f(X)$ of degree n that using its first 5 coefficients, *extracts* not k_s but its inverse (recall that if k_s^{-1} is a Nth root of unity, k_s is one as well), the next $\log(N)$ coefficients to obtain the powers of 2 of k_s^{-1}, and the last one to prove that $\mathsf{k}_s^N = (\mathsf{k}_s^{-1})^N = 1$.

Indeed, the prover shows that if $z(X)$ has degree 1, meaning that there exist a, b such that $z(X) = aX + b = a(X + \frac{b}{a})$, then $f(\mathsf{u}_5) = \frac{a}{b}$. The other coefficients of $f(X)$ are constructed so $f(\mathsf{u}_{5+i}) = \left(\frac{a}{b}\right)^{2^i}$, and the last one used to show that $\left(\frac{a}{b}\right)^N = \left(\frac{b}{a}\right)^N = 1$. By iteratively demonstrating that $f(\mathsf{u}_{5+i+1}) = f(\mathsf{u}_{5+i}) f(\mathsf{u}_{5+i})$ we can compute the powers of $\frac{a}{b}$ up to $2^{\log N} = N$ while performing only $O(\log(N))$ computations.

Case $m > 1$**.** In this case, we construct a scheme for proving position hiding linkability between two KZG vector commitment schemes. cm is a commitment to $\mathbf{a} = (a_1, \ldots, a_j)$, where a_j is an element in $\mathbf{v}$ for all $j = 1, \ldots, m$. We commit to $\mathbf{a}$ as a polynomial $\phi(X) = \sum_{j=1}^{m} a_j \lambda_j(X)$ where $\{\lambda_j(X)\}_{j=1}^m$ are Lagrange interpolation polynomials over a subgroup of roots of unity $\mathbb{H} = \{\mathsf{h}_1, \ldots, \mathsf{h}_m\}$. Let $I = \{s_1, \ldots, s_m\} \subset [N]$, where each $s \in I$ is included only once; that is, the set of all index s such that v_s is an element of $\mathbf{a}$, without repetitions. Prover commits to $C_I(X) = \sum_{s \in I} v_s \eta_s(X)$, where $\{\eta_s(X)\}_{s \in I}$ are the Lagrange interpolation polynomials of $\mathbb{K}_I = \{\mathsf{k}_s\}_{s \in I}$, as explained in Section 2.5.3.

Using a KZG proof of openings with blinded commitments to $C_I(X)$ and $z_I(X)$, the prover sends $[Q_I(\tau)]_1$ where $Q_I(X)$ is a blinded version of $Q'_I(X)$ such that

$$C(X) - C_I(X) = z_I(X)Q'_I(X).$$

Then, it remains to prove that (i) $z_I(X)$ has the right form, (ii) $[C_I(\tau)]_1$ is a commitment to the same values as $\mathsf{cm} = \sum_j^m a_j\lambda_j(X)$, without repetitions and in a different basis: $\{\eta_s(X)\}$ vs $\{\lambda_j(X)\}$. For the first statement we introduce an auxiliary polynomial $f(X) = \sum_{j=1}^m \mathsf{k}_{s_j}\lambda_j(X)$ that includes all the k_s with $s \in I$ but with the corresponding repetitions, meaning that if $a_{j_1} = a_{j_2} = c_{\hat{s}}$, $\mathsf{k}_{s_1} = \mathsf{k}_{s_2} = \mathsf{k}_{\hat{s}}$. We prove that $f(X)$'s coefficients are Nth roots of unity by providing a proof that $f_j(X) = f_{j-1}(X)f_{j-1}(X)$ for $j = 1, \ldots, m$, when evaluated at elements in $\mathbb{H}$, and showing that $f_0(X) = f(X)$ and $f_n(X) = 1$. Then it remains to prove that $z_I(X)$ vanishes at all the coefficients of $f(X)$ i.e. $z_I(f(X))$ vanishes in all elements of $\mathbb{H}$. This is done by providing $Q_2(X)$ such that $z_I(f(X)) = z_H(X)Q_2(X)$. Note that the argument holds also when $f(X)$ has repeated coefficients.

The second statement is proven by asserting the polynomial equation

$$C_I(f(X)) - \phi(X) = z_H(X)Q_3(X)$$

holds for some $Q_3(X)$, thus linking an input $\phi(X)$ in the known basis $\{\lambda_j(X)\}_{j=1}^m$ to $C_I(X)$ in the unknown basis $\{\eta_s(X)\}_{s\in I}$. In our protocol, the procedure above is performed by also including blinders to hide the positions and the values taken to construct $z_I(X), \mathbf{v}_I = (v_s)_{s\in I}$ and $\mathbf{a}$.

In brief, we construct a proof of membership where prover performs $O(\log N)$ operations for N-sized commitments. Our construction naturally extends to proof of subset memberships, thus leading the way to more efficient lookup arguments. We have removed the bottleneck of big tables by achieving the yet impossible $O(m \log N + m^2)$ cost for m-subvector lookups. The verifier is succinct as it requires only $O(\log(\log N))$ scalar operations as well as constant number of pairings to verify a constant-size proof. We envision the widespread deployment of our construction both in generic lookup-equipped proof systems [GW20, PFM+22] and specific applications with membership proofs. Mary Maller and Dmitry Khovratovich have implementated Caulk[1] in Rust, and we use that implementation for concrete comparison with other solutions as well.

5.1.2 Related Work

Merkle-SNARK. Zcash protocol [ZCary] proposed a SNARK over a circuit describing a Merkle tree opening for the anonymous proof of coin ownership, which remains

[1] https://github.com/caulk-crypto/caulk

Scheme	Setup	|srs|	Proof size	Prover work	Verifier work
Merkle trees + zkSNARKs	Reusable	$m\log(N)$	$13\mathbb{G}_1, 8\mathbb{F}$	$\tilde{O}(m\log(N))$	2P
RSA accumulators	Trusted	$O(1)$	$2\ \mathcal{G}$	$O(\log(m))$	m exp
Caulk single opening (Sec. 5.3)	Reusable	$O(N)$	$6\mathbb{G}_1, 2\mathbb{G}_2, 4\mathbb{F}$	$\tilde{O}(\log(N))$	4P
Caulk lookup (Sec. 5.4)	Reusable	$O(N)$	$14\mathbb{G}_1, 1\mathbb{G}_2, 4\mathbb{F}$	$\tilde{O}(m^2 + m\log(N))$	4P

Table 5.1: Cost comparison of our scheme with alternative proofs for membership and lookups. N is the size of the table and m the size of the set to be opened. We consider that Merkle trees + zk-SNARKs are implemented using Marlin [CHM+20] and note that these numbers are different with other SNARKs. Note that the asymptotic prover work for the Merkle trees + zkSNARKs hides the large constants involved in arithmetising hash functions. The RSA accumulator asymptotics hides large constants: for example $\mathcal{G}$ denotes a hidden order group that has larger size than $\mathbb{G}_1$, $\mathbb{G}_2$.

a very popular approach for various set membership proof protocols [Tor21, ZkS21]. The prover costs are logarithmic in the number of tree leafs, but the concrete efficiency varies depending on the hash function that comprises the tree [AGR+16, GKR+21]. Regular hash functions such as SHA-2 are known to be very slow, whereas algebraic alternatives such as Poseidon are rather novel and some applications are reluctant to use them.

Pairing Based. Camenisch et al.[CCs08] describe a vector commitment that only requires constant prover and verifier costs for individual openings. However the commitments themselves are computed by a trusted third party and have linear size because the prover requires access to $[(\tau - c_i)^{-1}]$ for all elements c_i in the vector and τ secret.

Discrete-Log Based. In the discrete-logarithm setting a series of works have looked into achieving logarithmic sized zero-knowledge membership proof [BG12, GK15, BCC+15, BG18]. These have the advantage that there is no trusted setup or pairings, but the prover and verifier costs are asymptotically dominated by a linear number of field operations. For modest sized vectors this can be practical because the number of more computationally intensive group operations is logarithmic.

RSA Accumulators. Camenisch and Lysyanskaya [CL02] design a proof of knowledge protocol for linking a commitment over a prime ordered group to an RSA accumulator. There are no a-priori bounds on the size of the vector and nicely, RSA based schemes have constant size public parameters. This approach is used by Zerocoin [MGGR13] which is a privacy preserving payments system (the predecessor to Zerocash [BCG+14]). Benarroch et al. [BCF+21] improve on this result by allowing the use of prime ordered groups of standard size, e.g., 256 bits, whereas [CL02] needs a much larger group. As opposite to Merkle tree constructions, [BCF+21] has prover time constant on the size of the table, and gets up to almost four times faster for elements of arbitrary size and between 4.5 and 23.5 for elements that are large prime numbers; as drawback, proof size goes from 4 to 5 KB. Later, Campanelli et

al. [CFH+21] present also an scheme for position-hiding linkability of RSA accumulators for large prime numbers and Pedersen commitments. Their proving times does not depend on the size of the accumulator and outperforms Merkle tree approaches by orders of magnitude; however they require either a trusted RSA modulus or class groups.

Pairing-Based Vector Commitments. Benarroch et al. introduced in [BCF+21] what we define as position-hiding linkability for a commitment C corresponding to the PST vector commitment scheme [BMM+21] and a commitment cm to one element using Pedersen's scheme. Similar to ours, their construction consists on opening a public polynomial encoding a vector at some hiding position s(instead of at element k_s) and prove that the output is the element committed in cm, along with well formation of the input (by showing that $s < N$). Still, their construction has a proof of size logarithmic in N and asks the verifier to perform $O(\log N)$ group operations and $\log(N)$ pairings.

5.2 Position-Hiding Linkable VC schemes

We introduce the concept of position-hiding linkable vector commitment schemes. Informally, two vector commitment schemes VC_1 and VC_2 are position-hiding linkable if a prover is able to convince a verifier that for given commitments C corresponding to VC_1 and cm corresponding to VC_2, it is true that all the elements in the vector committed in cm are also elements of the vector committed in C.

Basicallly, position-hiding linkability allows the prover to extract or isolate in zero-knowledge elements from some public set or table, and later prove further attributes on them. This new primitive should satisfy three security notions: completeness, as usual; *linkability*, that captures the fact that if the proof verifies then there is no element committed in cm that is not also committed in C; and *position-hiding*, which holds only if no information about the set of elements in C that have been used to construct cm is leaked.

Definition 23 (Position-Hiding Linkability for Vector Commitments)**.** *Two vector commitment schemes* VC_1 *and* VC_2 *are position-hiding linkable if there exist algorithms* $\left(\mathsf{KeyGen}, \mathsf{Prove}_{\mathsf{link}}, \mathsf{Verify}_{\mathsf{link}}, \mathsf{Simulate}_{\mathsf{link}}\right)$ *that behave as follows,*

- $(\mathsf{srs}, \tau) \leftarrow \mathsf{KeyGen}(1^\lambda, d_1, d_2)$: *takes as input the security parameter, bounds on the length of vectors in* VC_1 *and* VC_2*, and outputs common parameters* srs *that include* $\mathsf{srs}_1 = \mathsf{srs}_{\mathsf{VC}_1}$ *and* $\mathsf{srs}_2 = \mathsf{srs}_{\mathsf{VC}_2}$ *as well as a trapdoor* τ*, including the corresponding trapdoors* τ_1 *and* τ_2*.*

- $\pi \leftarrow \mathsf{Prove}_{\mathsf{link}}(\mathsf{srs}, r, r', \mathbf{v}, \mathbf{a})$: *on input the* srs*, commitment randomness* r *to vector* $\mathbf{v} \in \mathbb{F}^N$ *for some* $N \leq d_1$ *and commitment randomness* r' *to* $\mathbf{a} \in \mathbb{F}^m$

for $m \leq d_2$, outputs a proof π that there exists some $I \subset [N]$ such that for all $j = 1, \ldots, m$, $a_j = v_i$ for some $i \in I$.

- $b \leftarrow \mathsf{Verify}_{\mathsf{link}}(\mathsf{srs}, \mathsf{C}, \mathsf{cm}, \pi)$: *On input the srs, commitments* C *and* cm*, and proof π, accepts or rejects.*
- $\pi_{\mathsf{sim}} \leftarrow \mathsf{Simulate}_{\mathsf{link}}(\tau, \mathsf{C}, \mathsf{cm})$: *On input the trapdoor τ and commitments* C *and* cm*, outputs a simulated proof π_{sim},*

and satisfy the following properties:

Completeness: *For all N, m with $N \leq d_1$ and $m \leq d_2$, all $\mathbf{v} \in \mathbb{F}^N$, and all $\mathbf{a} \in \mathbb{F}^m$ it holds that:*

$$\Pr\left[\mathsf{Verify}_{\mathsf{link}}(\mathsf{srs}, \mathsf{C}, \mathsf{cm}, \pi) = 1 \;\middle|\; \begin{array}{c} (\mathsf{srs}, \tau) \leftarrow \mathsf{KeyGen}(1^\lambda, d_1, d_2); \\ \mathsf{C} \leftarrow \mathsf{VC}_1.\mathsf{Commit}(\mathsf{srs}_1, \mathbf{v}, r); \\ \mathsf{cm} \leftarrow \mathsf{VC}_2.\mathsf{Commit}(\mathsf{srs}_2, \mathbf{a}, r'); \\ \pi \leftarrow \mathsf{Prove}_{\mathsf{link}}(\mathsf{srs}, r, r', \mathbf{v}, \mathbf{a}) \end{array}\right] = 1.$$

Linkability *For all N, m with $N \leq d_1$ and $m \leq d_2$, and all PPT adversaries $\mathcal{A}$, there exists an extractor $\mathcal{E}$ such that:*

$$\Pr\left[\begin{array}{c} \mathsf{Verify}_{\mathsf{link}}(\mathsf{srs}, \mathsf{C}, \mathsf{cm}, \pi) = 1 \;\wedge \\ |\mathbf{v}| = N \;\wedge \\ \big(\exists\, j \in [m] \;s.t.\; a_j \neq c_i \;\forall i \in [N] \;\vee \\ \mathsf{VC}_2.\mathsf{Commit}(\mathsf{srs}_2, \mathbf{a}, r') \neq \mathsf{cm}\big) \end{array} \;\middle|\; \begin{array}{c} (\mathsf{srs}, \tau) \leftarrow \mathsf{KeyGen}(1^\lambda, d_1, d_2); \\ \mathbf{v} \leftarrow \mathcal{A}(\mathsf{srs}); \\ \mathsf{C} \leftarrow \mathsf{VC}_1.\mathsf{Commit}(\mathsf{srs}_1, \mathbf{v}); \\ (\pi, \mathsf{cm}) \leftarrow \mathcal{A}(\mathsf{srs}, \mathsf{C}); \\ (\mathbf{a}, r') \leftarrow \mathcal{E}(\mathsf{cm}, \pi) \end{array}\right] = \mathsf{negl}(\lambda)\,.$$

Position-Hiding *For all N, m with $N \leq d_1$ and $m \leq d_2$, all $\mathbf{v}$ and $\mathbf{a}$, all PPT adversaries $\mathcal{A}$, there exists a PPT algorithm $\mathsf{Simulate}_{\mathsf{link}}$ such that:*

$$\left[\mathcal{A}(\mathsf{srs}, \mathsf{C}, \mathsf{cm}, \pi) = 1 \;\middle|\; \begin{array}{c} (\mathsf{srs}, \tau) \leftarrow \mathsf{KeyGen}(1^\lambda, d_1, d_2) \\ \mathsf{C} \leftarrow \mathsf{VC}_1.\mathsf{Commit}(\mathsf{srs}_1, \mathbf{v}, r) \\ \mathsf{cm} \leftarrow \mathsf{VC}_2.\mathsf{Commit}(\mathsf{srs}_2, \mathbf{a}, r') \\ \pi \leftarrow \mathsf{Prove}_{\mathsf{link}}(\mathsf{srs}, r, r', \mathbf{v}, \mathbf{a}) \end{array}\right] \approx_c$$
$$\left[\mathcal{A}(\mathsf{srs}, \mathsf{C}, \mathsf{cm}, \pi_{\mathsf{sim}}) = 1 \;\middle|\; \begin{array}{c} (\mathsf{srs}, \tau) \leftarrow \mathsf{KeyGen}(1^\lambda, N, m) \\ \mathsf{C} \leftarrow \mathsf{VC}_1.\mathsf{Commit}(\mathsf{srs}_1, \mathbf{v}, r) \\ \mathsf{cm} \leftarrow \mathsf{VC}_2.\mathsf{Commit}(\mathsf{srs}_2, \mathbf{a}, r') \\ \pi_{\mathsf{sim}} \leftarrow \mathsf{Simulate}_{\mathsf{link}}(\tau, \mathsf{C}, \mathsf{cm}) \end{array}\right]$$

Below, we introduce position-hiding linkability for KZG commitments of arbitrary size and Pedersen commitments for single elements (Section 5.3), as well as for two KZG commitments (Section 5.4).

5.3 Linking Vectors with Elements

In this section we present a method to link a commitment C to a vector $\mathbf{v} \in \mathbb{F}^N$ computed as $\mathsf{C} = [C(\tau)]_1$ with $C(X) = \sum_{s=1}^{N} v_s \rho_s(X)$, to a Pedersen commitment cm for a value v. By this we mean a method for a prover to convince a verifier that there exists an s such that C opens to v at k_s and $\mathsf{cm} = [v + \mathsf{g}r_1]_1$, where r_1 is a blinding value.

We will consider two groups of roots of unity:

- $\mathbb{K} = \{\mathsf{k}_1, \ldots, \mathsf{k}_N\}$ of size N where $\mathsf{k}_s = \omega^{s-1}$ for $\omega^N = 1$, Lagrange interpolation polynomials $\{\rho_s(X)\}_{s=1}^N$ where $\rho_s(\mathsf{k}_s) = 1$ and $\rho_s(\mathsf{k}_j) = 0$ if $j \neq s$, and vanishing polynomial $z_K(X)$.
- $\mathbb{U}_n = \{\mathsf{u}_1, \ldots, \mathsf{u}_n\}$ of size $n = \log(N) + 6$ with $\mathsf{u}_s = \nu^{s-1}$, $\nu^n = 1$, Lagrange interpolation polynomials $\{\mu_i(X)\}_{i=1}^n$ and vanishing polynomial $z_U(X)$.

Our construction can be divided into three components. The main one is a modified protocol for computing blinded versions of KZG openings for statements $C(\mathsf{k}_s) = v$, that does not reveal the coordinate s or the evaluation v, which we describe below. The high-level idea here is to re-randomize a regular KZG opening with an additional blinding factor. It uses as subroutines the other two components. In one side, a proof of knowledge for the element v committed in cm, that is a proof for relation R_{ped} as defined in Section 2.5.4. Finally, it also needs an scheme to prove that the re-randomized vanishing polynomial used for the KZG opening is well-formed, i.e., a NIZK argument (as in Def. 1) for the relation

$$R_{\mathsf{unity}} = \left\{ (\mathsf{srs}, [z]_2;\ (a, s)) : \quad [z]_2 = [a(\tau - \mathsf{k}_s)]_2 \ \wedge\ (\mathsf{k}_s)^N = 1 \right\}.$$

5.3.1 Our Blinded Evaluation Construction

Our prover takes $(r' = \perp, \mathbf{v})$ and (r_1, v) as input, where the first tuple represents the vector inside the (deterministic) KZG commitment using the Lagrange basis, and the second tuple represents the randomness and value for the Pedersen commitment. Let $C(X) = \sum_{s=1}^{N} v_s \rho_s(X)$ be the polynomial encoding vector $\mathbf{v}$. In a regular KZG opening for position s, the prover would compute $Q(X) = \frac{C(X) - v}{X - \mathsf{k}_s}$ and reveal $[Q]_1 = [Q(\tau)]_1$. Instead, our prover computes a special kind of obfuscated commitment to k_s by selecting a random a and committing to $z(X) = aX - b = a(X - \mathsf{k}_s)$ where $\mathsf{k}_s = \frac{b}{a}$, i.e. the commitment is $[z]_2 = [z(\tau)]_2$. The blinding factor is necessary, because the set $\{\mathsf{k}_s\}_{s=1}^N$ is polynomial sized, so revealing $[\tau - \mathsf{k}_s]_1$ would allow the verifier to do a brute force search to find the index. The prover then computes $[T]_1 = [T(\tau)]_1$ and $[S]_2 = [S(\tau)]_2$, where

$$T(X) = \frac{Q(X)}{a} + \mathsf{g}r_2 \quad \text{and} \quad S(X) = -r_1 - r_2 z(X),$$

and r_2 is a uniformly random value chosen by the prover. $T(X)$ is the KZG quotient polynomial $Q(X)$ divided by a (the blinding factor above) to compensate for $z(X)$ having that blinding factor. The additional term $[\mathsf{g}r_2]_1$ mixed in to fully blind the evaluation $[\frac{Q(\tau)}{a}]_1$ and preserve zero-knowledge. $[S]_2$ is a term that compensates for the g terms in both $[T]_1$ and cm. In the pairing equation that checks these points, $[S]_2$ will be paired with g to ensure that it can only cancel out terms containing g and cannot make incorrect quotient polynomials appear correct.

The prover also provides two proofs of knowledge π_{ped} and π_{unity} as described in 2.5.4 and 5.3.2 respectively. The proof π_{ped} is for v, r_1 such that $\mathsf{cm} = [v + \mathsf{g}r_1]_1$. The proof π_{unity} is for a, b such that $[z]_2 = [a\tau - b]_2$ and $a^N = b^N$. The verifier checks the pairing equation

$$e(\mathsf{C} - \mathsf{cm}, [1]_2) = e([T]_1, [z]_2) + e([\mathsf{g}]_1, [S]_2).$$

Common inputs: C, cm

Prover: Sample blinders $a, r_2 \xleftarrow{\$} \mathbb{F}$

Re-compute $C(X) = \sum_{s=1}^{N} v_s \rho_s(X)$, encoding of $\mathbf{v}$ and $\mathsf{cm} = v[1]_1 + r_1[\mathsf{g}]_1$

Define

$$z(X) = a(X - \mathsf{k}_s), \quad T(X) = \frac{C(X) - v}{z(X)} + r_2\mathsf{g}, \quad S(X) = -r_1 - r_2 z(X)$$

$\pi_{\mathsf{ped}} \leftarrow \mathsf{Prove}(R_{\mathsf{ped}}, \mathsf{cm}, (v, r_1))$

$\pi_{\mathsf{unity}} \leftarrow \mathsf{Prove}(R_{\mathsf{unity}}, (\mathsf{srs}, [z]_2), (a, a\mathsf{k}_s))$

Set $[z]_2 = [z(\tau)]_2, [T]_1 = [T(\tau)]_1, [S]_2 = [S(\tau)]_2$.

Output $\pi = ([z]_2, [T]_1, [S]_2, \pi_{\mathsf{ped}}, \pi_{\mathsf{unity}})$

Verifier: Accept if and only if the following conditions hold

$$e(\mathsf{C} - \mathsf{cm}, [1]_2) = e([T]_1, [z]_2) + e([\mathsf{g}]_1, [S]_2)$$
$$1 \leftarrow \mathsf{Verify}_{\mathsf{ped}}(\mathsf{srs}, \mathsf{cm}, \pi_{\mathsf{ped}})$$
$$1 \leftarrow \mathsf{Verify}_{\mathsf{unity}}(\mathsf{srs}, [z]_2, \pi_{\mathsf{unity}})$$

Figure 5.1: Zero-knowledge proof of membership. Shows that (v, r_1) is an opening of cm and that C opens to v at k_s.

This equation checks that, for the polynomials $C(X), T(X), z(X), S(X)$ encoded in $\mathsf{C}, [T]_1, [z]_2$, and $[S]_2$ respectively, it holds that

$$C(X) - v - \mathsf{g}r_1 = T(X)z(X) + \mathsf{g}S(X).$$

Because $T(X) = \frac{Q(X)}{a} + r_2\mathsf{g}$, $z(X) = a(X - \mathsf{k}_s)$, and $S(X) = -r_1 - r_2 z(X)$, this is

$$C(X) - v - \mathsf{g}r_1 = \left(\frac{Q(X)}{a} + r_2\mathsf{g}\right) z(X) - \mathsf{g}r_1 - \mathsf{g}r_2 z(X) \iff$$
$$C(X) - v = \left(\frac{Q(X)}{a}\right) z(X).$$

The full description of our protocol is given in Figure 5.1.

Theorem 19. *Let R_{ped} and R_{unity} be relations for which zero-knowledge arguments of knowledge systems are given. The construction in Figure 5.1 implies position-hiding linkability for the commitment schemes corresponding to* C *and* cm *in the algebraic group model under the q-SDH and* dlog *assumptions.*

Proof. We will proceed through a series of games to show that the protocol defined in Fig. 5.1 satisfies the linkability property. Let $\mathcal{A}$ be an arbitrary algebraic PPT adversary in the linkability game and let $\mathsf{Adv}^{\mathsf{linkability}}_{\mathcal{A}}(\lambda)$ be their advantage. Let Game_0 be defined as in Definition 23, which is where we want to bound the adversarys success probability. We define $\mathsf{Game}_1, \mathsf{Game}_2$ and denote $\mathsf{Adv}^{\mathsf{Game}_i}_{\mathcal{A}}$ as the advantage of the adversary $\mathcal{A}$ in game i. We also specify reductions $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4$ such that

$$\begin{aligned}\mathsf{Adv}^{\mathsf{linkability}}_{\mathcal{A}} = \mathsf{Adv}^{\mathsf{Game}_0}_{\mathcal{A}} &\leq \mathsf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{unity}}_{\mathcal{B}_1}(\lambda) \\ &\leq \mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{ped}}_{\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{unity}}_{\mathcal{B}_1}(\lambda) \\ &\leq \mathsf{Adv}^{\mathsf{unity}}_{\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{ped}}_{\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{dlog}}_{\mathcal{B}_3}(\lambda) + \mathsf{Adv}^{\mathsf{qSDH}}_{\mathcal{B}_4}(\lambda)\end{aligned}$$

In Game_0 the adversary will return cm along with a proof $([z]_2 = [z(\tau)]_2, [T]_1 = [T(\tau)]_1, [S]_2 = [S(\tau)]_2, \pi_{\mathsf{ped}}, \pi_{\mathsf{unity}})$. We define Game_1 identically to Game_0, but after the adversary returns cm along with the proof, Game_1 additionally checks whether there exists a, b such that $z(X) = a(X - b)$ with $a^N = b^N$ and abort if this is not the case. Note that Game_1 can extract $z(X)$, the algebraic representation of $[z]_2$, because the adversary $\mathcal{A}$ is algebraic .

We observe that the adversarys advantage in Game_0 and Game_1 is identical, unless it manages to break the knowledge soundness of R_{unity}. Given such an $\mathcal{A}$, we can thus directly get a reduction $\mathcal{B}_1$ against the knowledge soundness of R_{unity} and let the advantage of this adversary be $\mathsf{Adv}_{\mathcal{B}_1}$. The reduction $\mathcal{B}_1$ simply runs $\mathcal{A}$ and returns π_{unity} that is returned by $\mathcal{A}$. It thus holds that

$$\mathsf{Adv}^{\mathsf{linkability}}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{Game}_0}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{unity}}_{\mathcal{B}_1}(\lambda).$$

Now define Game_2, which is identical to Game_1, but after the (algebraic) adversary $\mathcal{A}$ outputs cm the game Game_2 extracts v and r such that $\mathsf{cm} = [v + \mathsf{g}r]_1$. If this extraction fails, meaning that cm is not correctly formed, then Game_2 aborts. We note that the $\mathcal{A}$s advantage in Game_1 is identical to its advantage in Game_2, unless it manages to break the knowledge soundness of R_{ped}. Given $\mathcal{A}$, we can construct a reduction $\mathcal{B}_2$ against the knowledge soundness of R_{ped} analogously to the reduction above and let the advantage of this adversary be $\mathsf{Adv}_{\mathcal{B}_2}$. We observe that

$$\mathsf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{ped}}_{\mathcal{B}_2}(\lambda).$$

Recall that any adversary who successful wins Game_2 must output a proof that satisfies the following equation from the verification procedure

$$\begin{aligned} C(\tau) - v - \mathsf{g}r_1 = T(\tau)z(\tau) + \mathsf{g}S(\tau) \Leftrightarrow \\ C(\tau) - v = T(\tau)a(\tau - \mathsf{k}_s) + \mathsf{g}(r_1 + S(\tau)), \end{aligned}$$

while at the same time it must hold that

$$C(X) - v \neq (X - \mathsf{k}_s)aT(X)$$

for any polynomial $aT(X)$, since v is not in the committed vector $\mathbf{v}$. Intuitively, the adversary cannot satisfy this equation, since g is unknown to the prover and thus $(r_1 + S(X))$ is chosen independently of g. More formally, we consider two cases here. If

$$C(\tau) - v \neq T(\tau)a(\tau - \mathsf{k}_s)$$

then we can construct a reduction $\mathcal{B}_3$ breaking the discrete logarithm problem. Else if

$$C(\tau) - v = T(\tau)a(\tau - \mathsf{k}_s)$$

then we can construct a reduction $\mathcal{B}_4$ breaking the qSDH problem.

The reduction $\mathcal{B}_3$ takes as input a challenge $[y]_1$. It runs the adversary $\mathcal{A}$ against Game_2 over an srs in which $[\mathsf{g}]_1 = [y]_1$ and $\mathcal{B}_3$'s choice of τ (where τ is the trapdoor information of the KZG commitment). Whenever the adversary returns an output $([z]_2 = [z(\tau)]_2, [T]_1 = [T(\tau)]_1, [S]_2 = [S(\tau)]_2, \pi_{\mathsf{ped}}, \pi_{\mathsf{unity}})$ which wins the Game_2 game, then $\mathcal{B}_3$ returns

$$\mathsf{g} = \frac{C(\tau) - v - T(\tau)z(\tau)}{r_1 + S(\tau)},$$

where $T(X), r_1$ and $S(X)$ are extracted from the outputs of $\mathcal{A}$. The reduction's success probability is exactly the success probability of the adversary conditioned on $(r_1 + S(\tau)) \neq 0$.

The reduction $\mathcal{B}_4$ takes as input the challenge $[y_1]_1, \ldots, [y_q]_1$. It runs the following reduction $\mathcal{B}_{\mathsf{KZG}}$ as a subroutine. The $\mathcal{B}_{\mathsf{KZG}}$ runs the adversary $\mathcal{A}$ against Game_2 over an srs in which $[\tau]_1 = [y_1]_1$ and $\mathcal{B}_{\mathsf{KZG}}$'s choice of g. Whenever the adversary returns

an output $([z]_2 = [z(\tau)]_2, [T]_1 = [T(\tau)]_1, [S]_2 = [S(\tau)]_2, \pi_{\mathsf{ped}}, \pi_{\mathsf{unity}})$ which wins the Game_2 game, then $\mathcal{B}_{\mathsf{KZG}}$ returns the KZG openings

$$(v, [a^{-1}T]_1) \text{ and } (C(\mathsf{k}_s), [\frac{C(\tau) - C(\mathsf{k}_s)}{\tau - \mathsf{k}_s}]_1)$$

for $v \neq C(\tau)$. Then $\mathcal{B}_4$ can extract a qSDH solution from these openings following the proof in Theorem 1 of [KZG10].

We can thus conclude that

$$\mathsf{Adv}^{\mathsf{linkability}}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{unity}}_{\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{ped}}_{\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{dlog}}_{\mathcal{B}_3}(\lambda) + \mathsf{Adv}^{\mathsf{qSDH}}_{\mathcal{B}_3}(\lambda).$$

Lastly, we prove the position hiding property of our construction. We define a simulator $\mathsf{Simulate}$ that has access to the trapdoor x of srs that is indistinguishable from an honest prover. First, $\mathsf{Simulate}$ calls the simulators of R_{ped} and R_{unity} on input the trapdoor x, and gets simulated proofs π_{ped} and π_{unity}. Then, it samples $a, r_2 \leftarrow \mathbb{F}$ and sets $[z]_2 = [a]_2$, $[S]_2 = [r_2]_2$, $[T]_1 = (\mathsf{C} - \mathsf{cm} - [\mathsf{g}r_2]_1)/a$, and outputs $([z]_2, [T]_1, [S]_2, \pi_{\mathsf{ped}}, \pi_{\mathsf{unity}})$. Note that honestly generated $[z]_2, [S]_2$ are randomized by a and r_2, respectively, and thus indistinguishable from $[z]_2, [S]_2$. Finally, $[T]_1$ is the only element satisfying the verifying equation for given $[z]_2, [S]_2$ and thus indistinguishable from honest $[T]_1$ as well, which concludes the proof. □

5.3.2 Correct computation of $z(X)$

The purpose of this section is to provide a zero-knowledge proof of knowledge for relation R_{unity}, i.e. that the prover knows a, b such that $[z]_2 = [a\tau - b]_2$ and $a^N = b^N$. This proof is used as a subprotocol in Fig. 5.1.

In order to prove that $\frac{b}{a}$ is inside the evaluation domain (i.e. is an Nth root of unity) in zero-knowledge we add another polynomial $f(X)$ of degree $n = \log(N)+6$. The polynomial $f(X)$ essentially recovers $\frac{a}{b}$ from $[z]_2$ and then includes its powers 2^i until $i = \log(N)$. It will be enough then to prove that (i) $f(X)$ is correctly formed with respect to $[z]_2$, (ii) it does indeed contain all 2-powers of $\frac{a}{b}$, and (iii) the coefficient corresponding to $(\frac{a}{b})^{2^{\log(N)}} = (\frac{a}{b})^N$ equals 1.

The core of our construction is the following lemma:

Lemma 7. *Let $z(X)$ be a polynomial of degree 1, $n = \log(N) + 6$ and $\mathbb{U} = \{\mathsf{u}_1, \ldots, \mathsf{u}_n\}$ a set of roots of unity where $\mathsf{u}_i = \nu^{i-1}$ for $\nu^n = 1$. If there exists a polynomial $f(X) \in \mathbb{F}[X]$ such that*

1. *$f(X) = z(X)$ for $\mathsf{u}_1, \mathsf{u}_2$.*
2. *$f(\mathsf{u}_3)(\mathsf{u}_1 - \mathsf{u}_2) = f(\mathsf{u}_1) - f(\mathsf{u}_2)$*

3. $f(\mathsf{u}_4) = \mathsf{u}_2 f(\mathsf{u}_3) - f(\mathsf{u}_2)$

4. $f(\mathsf{u}_5) f(\mathsf{u}_4) = f(\mathsf{u}_3)$

5. $f(\mathsf{u}_{5+\log(N)}) = f(\mathsf{u}_{6+\log(N)}\nu^{-1}) = 1$

6. $f(\mathsf{u}_{5+i+1}) = f(\mathsf{u}_{5+i})^2$, *for all* $i = 0, \ldots, \log(N) - 1$

Then, $z(X) = aX - b$, *where* $\frac{b}{a}$ *is an* N*-th root of unity.*

Proof. Because $z(X)$ has degree 1, there exist $a, b \in \mathbb{F}$ such that $z(X) = aX - b$.

From the first condition, we have $f(\mathsf{u}_1) = a(\mathsf{u}_1) = a(\nu^0) = a - b$, and $f(\mathsf{u}_2) = a(\mathsf{u}_2) = a\mathsf{u}_2 - b$. From items 2 and 3,

$$f(\mathsf{u}_3) = \frac{f(\mathsf{u}_1) - f(\mathsf{u}_2)}{\mathsf{u}_1 - \mathsf{u}_2} = \frac{a - a\mathsf{u}_2}{1 - \mathsf{u}_2} = a,$$
$$f(\mathsf{u}_4) = \mathsf{u}_2 f(\mathsf{u}_3) - f(\mathsf{u}_2) = \mathsf{u}_2 a - a\mathsf{u}_2 + b = b.$$

By substituting $f(\mathsf{u}_3) = a$ and $f(\mathsf{u}_4) = b$ into condition 4 we see that $f(\mathsf{u}_5) = \frac{a}{b}$. Therefore, from item 5 we have that for every $i = 0, \ldots, \log(N) - 1$,

$$f(\mathsf{u}_{5+i+1}) = f(\mathsf{u}_{5+i})^2 = \left(\frac{a}{b}\right)^{2^{i+1}}.$$

In particular,

$$f(\mathsf{u}_{5+(\log(N)-1)+1}) = f(\mathsf{u}_{5+\log(N)}) = \left(\frac{a}{b}\right)^{2^{\log(N)}} = \left(\frac{a}{b}\right)^N,$$

that equals 1 by the 5th condition, proving that $\frac{a}{b}$ is a Nth root of unity as required.

□

In our protocol the prover will construct the polynomial $f(X)$ as

$$\begin{aligned} f(X) = (a-b)\mu_1(X) \quad & +(a\mathsf{u}_2 - b)\mu_2(X) + a\mu_3(X) + b\mu_4(X) \\ & + \textstyle\sum_{i=0}^{\log(N)} \left(\frac{a}{b}\right)^{2^i} \mu_{5+i}(X). \end{aligned} \tag{5.1}$$

and commit to it in zero-knowledge. Then, it will show it is correct by comparing $f(\mathsf{u}_i)$ with the corresponding values from the constraints in Lemma 7. Namely, for some α chosen by the verifier, it sets $\alpha_1 = \nu^{-1}\alpha$, $\alpha_2 = \mathsf{u}^{-2}\alpha$ and sends $v_1 = f(\alpha_1)$ and $v_2 = f(\alpha_2)$ along with the corresponding proofs of opening. Recall that $\mathsf{u}_i\alpha_1 = \mathsf{u}_{i-1}\alpha$ and $\mathsf{u}_i\alpha_2 = \mathsf{u}_{i-2}\alpha$.

Given v_1, v_2 it then shows that the following polynomial, which proves the constraints in Lemma 7, evaluates to 0 in α:

$$\begin{aligned} p_\alpha(X) = &- Q(X)z_U(\alpha) + \big(f(X) - z(X)\big)(\mu_1(\alpha) + \mu_2(\alpha)) \\ &+ \big((1-\mathsf{u}_2)f(X) - f(\alpha_2) + f(\alpha_1)\big)\mu_3(\alpha) \\ &+ \big(f(X) + f(\alpha_2) - \mathsf{u}_2 f(\alpha_1)\big)\mu_4(\alpha) + \big(f(X)f(\alpha_1) - f(\alpha_2)\big)\mu_5(\alpha) \\ &+ \big(f(X) - f(\alpha_1)f(\alpha_1)\big)(\alpha - \mathsf{u}_n)\prod_{j=1}^{5}(\alpha - \mathsf{u}_j) + \big(f(\alpha_1) - 1\big)\mu_n(\alpha). \end{aligned}$$

Note that the polynomials that are already evaluated in α in $p_\alpha(X)$ are thus that either the verifier can compute its own in $\log(\log(N))$ time, or are opened by the prover.

Using v_1, v_2, the commitments to $Q(X), f(X)$ and after computing $\mu_i(\alpha)$ for $i = 1, 2, 3, 4, 5, n$ and $(\alpha - \mathsf{u}_n)\prod_{j=1}^{5}(\alpha - \mathsf{u}_j)$, the verifier computes a commitment $[P]_1$ to $p_\alpha(X)$ and checks that (i) v_1, v_2 are correct openings of $f(X)$ at $\alpha_1 = \nu^{-1}\alpha$ and $\alpha_2 = \nu^{-2}\alpha$, (ii) 0 is a correct opening of $p_\alpha(X)$ at α, and (iii) $[z]_2$ has degree 1.

For this last check, we ask the prover to include a term $X^{d-1}z(X)$ in $Q(X)$ and then the verifier computes $[P]_1$ without the terms including $z(X)$, i.e, without $-X^d z(X)z_U(\alpha) - z(X)(\mu_1(\alpha) + \mu_2(\alpha))$. It will instead add them in the group via the pairing later, to assure that it cannot be the case that $\deg(z) > 1$, unless $\deg(p_\alpha) > d$, which is not possible under the AGM.

We describe the protocol in Fig. 5.2. PC denotes the KZG polynomial commitment scheme as described in Section 2.5.1.

Theorem 20. *The protocol in Fig. 5.2 is a knowledge-sound argument (as defined in Def.1) for relation* $\mathsf{R}_{\mathsf{unity}}$ *if KZG is a sound polynomial commitment scheme, under the the Algebraic Group and Random Oracle models. When used as a building block in the argument of Figure 5.1, it also satisfies zero-knowledge*[2]*.*

Proof. We proceed through a series of games to show that the protocol defined in 5.2 satisfies knowledge soundness. We set Game_0 to be the soundness game as in Def. 1 and consider an algebraic adversary $\mathcal{A}$ against it which has advantage $\mathsf{Adv}^{\mathsf{k\text{-}sound}}_{\mathcal{A}}$. We define Game_1, Game_2 and specify reductions $\mathcal{B}_1$ and $\mathcal{B}_2$ such that

$$\begin{aligned} \mathsf{Adv}^{\mathsf{k\text{-}sound}}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{Game}_0}_{\mathcal{A}}(\lambda) &\leq \mathsf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{qSDH}}_{\mathcal{B}_1}(\lambda) \\ &\leq \mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{qDHE}}_{\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{qSDH}}_{\mathcal{B}_1}(\lambda) \\ &\leq \mathsf{Adv}^{\mathsf{qSDH}}_{\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{qDHE}}_{\mathcal{B}_2}(\lambda) + \mathsf{negl}(\lambda)\,. \end{aligned}$$

[2]When used as an independent argument, $[z]_2$ must be an output of the prover in the first round, or in any round of the main scheme when plugged into other protocols.

Common input: $[z]_2$

Prove: Sample $r_0, r_1, r_2, r_3 \xleftarrow{\$} \mathbb{F}$ and let $r(X) \leftarrow r_1 + r_2 X + r_3 X^2$

$f(X) = (a-b)\mu_1(X) + (a\mathsf{u}_2 - b)\mu_2(X) + a\mu_3(X)$

$\quad +b\mu_4(X) + \sum_{i=0}^{\log(N)} \left(\frac{a}{b}\right)^{2^i} \mu_{5+i}(X) + r_0\mu_n(X) + r(X)z_U(X),$

$p(X) = \big(f(X) - (aX-b)\big)\big(\mu_1(X) + \mu_2(X)\big) + \big((1-\mathsf{u}_2)f(X)$

$\quad -f(\nu^{-2}X) + f(\nu^{-1}X)\big)\mu_3(X) + \big(f(X) + f(\nu^{-2}X) - \mathsf{u}_2 f(\nu^{-1}X)\big)\mu_4(X)$

$\quad +\big(f(X)f(\nu^{-1}X) - f(\nu^{-2}X)\big)\mu_5(X) + \big(f(\nu^{-1}X) - 1\big)\mu_n(X)$

$\quad +\big(f(X) - f(\nu^{-1}X)f(\nu^{-1}X)\big)(X - \mathsf{u}_n)\prod_{j=1}^{5}(X - \mathsf{u}_j),$

Set $\hat{h}(X) = \frac{p(X)}{z_U(X)}$, $Q(X) = \hat{h}(X) + X^{d-1}z(X)$,

Output $([F]_1 = [f(\tau)]_1, [Q]_1 = [Q(\tau)]_1)$.

Verify : Send challenge $\alpha \in \mathbb{F}$

Prove : $\alpha_1 = \nu^{-1}\alpha$, $\alpha_2 = \nu^{-2}\alpha$;

$p_\alpha(X) = -z_U(\alpha)Q(X) + \big(f(X) - z(X)\big)\big(\mu_1(\alpha) + \mu_2(\alpha)\big)$

$\quad +\big((1-\mathsf{u}_2)f(X) - f(\alpha_2) + f(\alpha_1)\big)\mu_3(\alpha)$

$\quad +\big(f(X) + f(\alpha_2) - \mathsf{u}_2 f(\alpha_1)\big)\mu_4(\alpha) + \big(f(X)f(\alpha_1) - f(\alpha_2)\big)\mu_5(\alpha)$

$\quad +\big(f(X) - f(\alpha_1)f(\alpha_1)\big)(\alpha - \mathsf{u}_n)\prod_{j=1}^{5}(\alpha - \mathsf{u}_j) + \big(f(\alpha_1) - 1\big)\mu_n(\alpha),$

Compute $((v_1, v_2), \pi_1) \leftarrow \mathsf{PC.Open}(\mathsf{srs}_{\mathsf{PC}}, f(X), \deg = \perp, (\alpha_1, \alpha_2))$,

$\quad (0, \pi_2) \leftarrow \mathsf{PC.Open}(\mathsf{srs}_{\mathsf{PC}}, p_\alpha(X), \deg = \perp, \alpha)$,

and output $\big(v_1, v_2, \pi_1, \pi_2\big)$.

Verify : Set $\alpha_1 = \nu^{-1}\alpha$; $\alpha_2 = \nu^{-2}\alpha$,

$[P]_1 = -z_U(\alpha)[Q]_1 + \big(\mu_1(\alpha) + \mu_2(\alpha)\big)[F]_1 + \mu_3(\alpha)\big((1-\mathsf{u}_2)[F]_1 + v_1 - v_2\big)$

$\quad +\mu_4(\alpha)\big([F]_1 + v_2 - \mathsf{u}_2 v_1\big) + \mu_5(\alpha)\big(v_1[F]_1 - v_2\big) + \mu_n(\alpha)\big(v_1 - 1\big)$

$\quad +(\alpha - \mathsf{u}_n)\prod_{j=1}^{5}(\alpha - \mathsf{u}_j)\big([F]_1 - v_1^2\big),$

Parse $\pi_2 = [\hat{Q}]_1$ and accept if and only if

$$1 \leftarrow \mathsf{PC.Verify}\big(\mathsf{srs}_{\mathsf{PC}}, [F]_1, \deg = \perp, (\alpha_1, \alpha_2), (v_1, v_2), \pi_1\big),$$

$$e\big([P]_1, [1]_2\big) + e\big(-(\mu_1(\alpha) + \mu_2(\alpha)) - z_U(\alpha)[\tau^{d-1}]_1, [z]_2\big) = e\big([\hat{Q}]_1, [\tau - \alpha]_2\big)$$

Figure 5.2: NIZK argument of knowledge for $\mathsf{R}_{\mathsf{unity}}$ and $\deg(z) \leq 1$.

In Game_0 the adversary will return $[z]_2$ along with a proof $([F]_1 = [f(\tau)]_1, [Q]_2 = [Q(\tau)], v_1, v_2, \pi_1, \pi_2)$. We also consider $\hat{p}(X)$, the algebraic representation of $[P]_1$ as constructed by the verifier. Note that π_2 is KZG opening proof for $p(X) = \hat{p}(X) - z(X)(\mu_1(\alpha) + \mu_2(\alpha) - z_U(\alpha)X^{d-1})$ opening to 0 at α. We define Game_1 identically to knowledge soundness, but after the adversary returns $[z]_2$ along with the proof, Game_2 additionally checks whether $f(\alpha_1) = v_1$, $f(\alpha_2) = v_2$, $p(\alpha) = 0$ and aborts otherwise. Note that Game_1 can extract $f(X)$, $Q(X)$ because the adversary $\mathcal{A}$ is algebraic, and $p(X)$ is constructed from them.

We show the probability that $f(\alpha_1) = v_1$, $f(\alpha_2) = v_2$, $p(\alpha) = 0$ is bounded by qSDH. We construct a reduction $\mathcal{B}_1$ that takes as input a challenge $[y]_1, \ldots, [y_q]_1$. It runs the following reduction $\mathcal{B}_{\mathsf{KZG}}$ as a subroutine. The $\mathcal{B}_{\mathsf{KZG}}$ runs the adversary $\mathcal{A}$ against Game_0 over an srs in which $[\tau]_1 = [y]_1$. Whever the adversary returns an output $([F]_1 = [f(\tau)]_1, [Q]_2 = [Q(\tau)], v_1, v_2, \pi_1, \pi_2, \pi_3)$ that wins the Game_0 but not the Game_1 game, then $\mathcal{B}_{\mathsf{KZG}}$ returns the KZG openings

$$\Big((v_1, [F]_1) \text{ and } (f(\alpha_1), [\frac{f(\tau) - f(\alpha_1)}{\tau - \alpha_1}])\Big),$$
$$\Big((v_2, [F]_1) \text{ and } (f(\alpha_2), [\frac{f(\tau) - f(\alpha_2)}{\tau - \alpha_2}])\Big) \text{ or } \Big((0, [P]_1) \text{ and } (p(\alpha), [\frac{p(\tau) - p(\alpha)}{\tau - \alpha}])\Big)$$

for either $v_1 \neq f(\alpha_1)$, $v_2 \neq f(\alpha_2)$ or $p(\alpha) \neq 0$. Then $\mathcal{B}_1$ can extract a qSDH solution from these openings following the proof of Theorem 3 in [KZG10]. Thus

$$\mathsf{Adv}^{\mathsf{k\text{-}sound}}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{Game}_0}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{qSDH}}_{\mathcal{B}_1}(\lambda).$$

We define Game_2 as Game_1 except that Game_2 additionally checks if $\deg(z) \leq 1$ for $z(X)$ being the algebraic representation of $[z]_2$, and aborts otherwise. We show that $\mathcal{A}$'s advantage in both games is the same unless it breaks qDHE. Indeed, assume $\deg(z) = 2$, we construct an adversary $\mathcal{B}_{\mathsf{qDHE}}$ against qDHE. The $\mathcal{B}_2$ takes as input the challenge $[y_1]_1, \ldots, [y_q]_1$ and runs $\mathcal{A}$ against Game_1 over an srs in which $[\tau]_1 = [y]_1$. When $\mathcal{A}$ returns an output $([F]_1 = [f(\tau)]_1, [Q]_2 = [Q(\tau)], v_1, v_2, \pi_1, \pi_2)$ that wins the Game_1 but not the Game_2 game, then $\mathcal{B}_2$ extracts $\hat{p}(X) = \sum_{s=0}^{d+1} \hat{p}_s X^s$ as the algebraic representation of $[P]_1$ computed by the verifier. Note that, since $\big(-\mu_1(\alpha) - \mu_2(\alpha) - z_U(\alpha)X^{d-1}\big)z(X)$ does not vanish at $X = \alpha$, we have that $\hat{p}_{d+1} \neq 0$. Then, $\mathcal{B}_2$ sets $\hat{P}(X) = P(X) - \hat{p}_{d+1}X^{d+1}$ and outputs $\big([P]_1 - [\hat{P}(\tau)]_1\big)\frac{1}{\hat{p}_{d+1}} = [\tau^{d+1}]_1$, wining qDHE. Thus

$$\mathsf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{qDHE}}_{\mathcal{B}_2}(\lambda).$$

Finally, let us show that

$$\mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)\,.$$

Consider $f(X)$, $Q(X)$ the algebraic representations of $[F]_1$, $[Q]_1$. The algebraic

representation of the element $[P]_1$ that the verifier constructs is

$$\begin{aligned} p(X) = &-z_U(\alpha)Q(X) + \big(\mu_1(\alpha) + \mu_2(\alpha)\big)f(X) \\ &+ \mu_3(\alpha)\big((1-\mathsf{u}_2)f(X) + v_1 - v_2\big) + \mu_4(\alpha)\big(f(X) + v_2 - \mathsf{u}_2 v_1\big) \\ &+ \mu_5(\alpha)\big(v_1 f(X) - v_2\big) + \mu_n(\alpha)\big(v_1 - 1\big) + (\alpha - \mathsf{u}_n)\prod_{j=1}^{5}(\alpha - \mathsf{u}_j)\big(f(X) - v_1^2\big) \end{aligned}$$

Since Game_2 checks that $v_1 = f(\nu^{-1}\alpha), v_2 = f(\nu^{-2}\alpha)$, we can replace these values and see that

$$\begin{aligned} p(X) = &-z_U(\alpha)Q(X) + \big(\mu_1(\alpha) + \mu_2(\alpha)\big)f(X) \\ &+ \mu_3(\alpha)\big((1-\mathsf{u}_2)f(X) + f(\nu^{-1}\alpha) - f(\nu^{-2}\alpha)\big) \\ &+ \mu_4(\alpha)\big(f(X) + f(\nu^{-2}\alpha) - \mathsf{u}_2 f(\nu^{-1}\alpha)\big) + \mu_5(\alpha)\big(f(\nu^{-1}\alpha)f(X) - f(\nu^{-2}\alpha)\big) \\ &+ (\alpha - \mathsf{u}_n)\prod_{j=1}^{5}(\alpha - \mathsf{u}_j)\big(f(X) - f(\nu^{-1}\alpha)^2\big) + \mu_n(\alpha)\big(f(\nu^{-1}\alpha) - 1\big) \end{aligned}$$

Now, because $p(\alpha) = 0$ and α has been chosen by the verifier after the prover has sent $[Q]_1, [F]_1$, except in the negligible case that α is a root of $p(X)$, we have that $p(X) \equiv 0$, i.e,

$$\begin{aligned} z_U(X)Q(X) = &-\big(\mu_1(X) + \mu_2(X)\big)f(X) \\ &+ \mu_3(X)\big((1-\mathsf{u}_2)f(X) + f(\nu^{-1}X) - f(\nu^{-2}X)\big) \\ &+ \mu_4(X)\big(f(X) + f(\nu^{-2}X) - \mathsf{u}_2 f(\nu^{-1}X)\big) \\ &+ \mu_5(X)\big(f(\nu^{-1}X)f(X) - f(\nu^{-2}X)\big) \\ &+ (X - \mathsf{u}_n)\prod_{j=1}^{5}(X - \mathsf{u}_j)\big(f(X) - f(\nu^{-1}X)^2\big) + \mu_n(X)\big(f(\nu^{-1}X) - 1\big) \end{aligned}$$

$z_U(X)$ divides the right side of the equation and thus, the latter vanishes for all $\{\mathsf{u}_i\}_{i=1}^n$. This implies that

- $f(\mathsf{u}_1) = a(\mathsf{u}_1)$, $f(\mathsf{u}_2) = a(\mathsf{u}_2)$
- $f(\mathsf{u}_3) = \frac{v_2 - v_1}{\mathsf{u}_1 - \mathsf{u}_2} = \frac{f(\mathsf{u}_3\nu^{-2}) - f(\mathsf{u}_3\nu^{-1})}{\mathsf{u}_1 - \mathsf{u}_2} = \frac{f(\mathsf{u}_1) - f(\mathsf{u}_2)}{\mathsf{u}_1 - \mathsf{u}_2}$
- $f(\mathsf{u}_4) = \mathsf{u}_2 f(\mathsf{u}_4\nu^{-1}) - f(\mathsf{u}_4\nu^{-2}) = \mathsf{u}_2 f(\mathsf{u}_3) - f(\mathsf{u}_2)$
- $f(\mathsf{u}_5)f(\mathsf{u}_5\nu^{-1}) = f(\mathsf{u}_5\nu^{-2})$, i.e, $f(\mathsf{u}_5)f(\mathsf{u}_4) = f(\mathsf{u}_3)$
- $1 = f(\mathsf{u}_n\nu^{-1}) = f(\mathsf{u}_{5+\log(N)})$
- $\big(f(\mathsf{u}_{5+i+1}) - f(\mathsf{u}_{5+i+1}\nu^{-1})f(\mathsf{u}_{5+i+1}\nu^{-1})\big)(\mathsf{u}_i - \mathsf{u}_n)\prod_{j=1}^{5}(\mathsf{u}_i - \mathsf{u}_j) = 0$ for all $i = 1, \ldots, n$. Note that $(\mathsf{u}_i - \mathsf{u}_n)\prod_{i=1}^{5}(\mathsf{u}_i - \mathsf{u}_j) \neq 0$ if $i \notin \{1, 2, 3, 4, 5, n\}$, which implies that $0 = f(\mathsf{u}_{5+i+1}) - f(\mathsf{u}_{5+i+1}\nu^{-1})f(\mathsf{u}_{5+i+1}\nu^{-1}) = f(\mathsf{u}_{5+i+1}) - f(\mathsf{u}_{5+i})^2$ for all $i = 0, \ldots, \log(N)$.

By Lemma 7 we have that $z(X) = aX - b$ where $\frac{a}{b}$ is an N-th root of unity.

For zero-knowledge, we define a simulator $\mathsf{Simulate}$ that has access to the trapdoor of srs and is indistinguishable from an honest prover. The simulator first chooses s_1, s_2, v_1, v_2 uniformly at random and sets $[F]_1 = [s_1]_1$ and $[Q]_1 = [s_2]_1$. It computes $\alpha_1 = \nu^{-1}\alpha$, $\alpha_2 = \nu^{-2}\alpha$. It then computes $[w_1]_1 = \big([F]_1 - v_1\eta_1(\tau) - v_2\eta_2(\tau)\big)\frac{1}{(\tau-\alpha_1)(\tau-\alpha_2)}$, for $\eta_1(\tau) = \frac{\tau-\alpha_2}{\alpha_1-\alpha_2}$, $\eta_2(\tau) = \frac{\tau-\alpha_1}{\alpha_2-\alpha_1}$.

It sets $[P]_1$ the same as the verifier i.e.

$$\begin{aligned}[P]_1 = &-[Q]_1 z_U(\alpha) + [F]_1\big(\mu_1(\alpha) + \mu_2(\alpha)\big) \\ &+ \big([F]_1(1 - \mathsf{u}_2) - v_2 + v_1\big)\mu_3(\alpha) + \big([F]_1 + v_2 - \mathsf{u}_2 v_1\big)\mu_4(\alpha) \\ &+ \big([F]_1 v_1 - v_2\big)\mu_5(\alpha) + \big(v_1 - 1\big)\mu_n(\alpha) + \big([F]_1 - v_1^2\big)(\alpha - \mathsf{u}_n)\prod_{i=1}^{5}(\alpha - \mathsf{u}_j)\end{aligned}$$

and then computes $[w_2]_1 = ([P]_1 - (\mu_1(\alpha) + \mu_2(\alpha) + z_U(\alpha)\tau^{d-1})z)\frac{1}{\tau-\alpha}$, where $z = a$ is the output of the simulator in the proof of Theorem 19.

It returns $([F]_1, [Q]_1, v_1, v_2, \pi_1 = [w_1]_1, \pi_2 = [w_2]_2)$.

We must argue that the simulators output is distributed identically to the honest provers. Then the provers components are randomised by

$$\begin{aligned} F: &\quad r_0\mu_n(\tau) & \qquad H: &\quad r(\tau) \\ v_1: &\quad r(\nu^{-1}\alpha)z_U(\alpha) & \qquad v_2: &\quad r(\nu^{-2}\alpha)z_U(\alpha)\end{aligned}$$

and the elements $[w_1]_1, [w_2]_1$ are the unique elements satisfying the verifies equations given $[F]_1, [Q]_1, v_1, v_2$. The probability that $r_0\mu_n(\tau)$, $r(\tau)$, $r(\nu^{-1}\alpha)z_U(\alpha)$, $r(\nu^{-2}\alpha)z_U(\alpha)$ are dependent at random α is negligible because r_0 is a random element and $r(X)$ a random degree 2 polynomial and the probability that $\nu^{-1}\alpha = \tau$ or $\nu^{-2}\alpha = \tau$ is $\frac{2}{|\mathbb{F}|}$. Where the simulators terms $[F]_1, [Q]_1, v_1, v_2$ are chosen uniformly at random and $[w_1]_1, [w_2]_1$ are the unique terms that satisfy the verifies equations, we have that these distributions are identical except with negligible probability. □

5.4 Lookup tables for hiding values

In this section we present position-hiding linkability for KZG vector commitment schemes (Section 2.5.2). The aim is to prove that a commitment cm contains a *subset* of some larger vector committed in C. We refer to a subset instead of a subvector since our scheme proves that all the elements committed in cm are also committed in C, but with no specific order and possible repetitions. This is essentially a lookup table if we consider that C contains the honestly generated table.

Preliminaries We will consider three evaluation domains

1. $\mathbb{K} = \{\mathsf{k}_1, \ldots, \mathsf{k}_N\}$ is a group of roots of unity of size N with Lagrange interpolation polynomials $\{\rho_s(X)\}_{s=1}^N$ where $\rho_s(\mathsf{k}_s) = 1$ and $\rho_s(\mathsf{k}_j) = 0$ if $j \neq s$, and vanishing polynomial $z_K(X)$.

2. For subset $\mathbb{K}_I = \{\mathsf{k}_s\}_{s\in I}$ of $\mathbb{K}$ defined by $I \subset [N]$, we set $\{\eta_s(X)\}_{s\in I}$ as its interpolation Lagrange polynomials with degree $|I| - 1$, and $z_I(X)$ as its vanishing polynomial. Note that typically $\mathbb{K}_I$ is not a subgroup.

3. For some constant m that bounds the size of the vector committed in cm, we consider another group of roots of unity $\mathbb{H} = \{\mathsf{h}_1, \ldots, \mathsf{h}_m\}$ and its Lagrange and vanishing polynomials, $\{\lambda_j(X)\}_{j=1}^m$ and $z_H(X)$.

Our scheme uses a subprotocol a NIZK argument of knowledge for relation $\mathsf{R}_{\mathsf{unity}}$,

$$R_{\mathsf{unity}} = \left\{ \begin{array}{c} (\mathsf{srs}, [z_I]_2, N;\ (I, r)) :\ I \subset [N]\ \wedge\ [z_I]_1 = r \prod_{s\in I} [\tau - \mathsf{k}_s]_1, \\ \text{with } (\mathsf{k}_s)^N = 1,\ \forall\ s \in I \end{array} \right\}$$

In our protocol, the prover takes as input a commitment $C(X) = \sum_{s=1}^N v_s \rho_s(X)$ to the lookup table $\mathbf{v}$, a structured reference string srs, and a commitment

$$\mathsf{cm} = [\phi(\tau)]_1 = \left[\sum_{j=1}^m a_j \lambda_j(\tau) + a_{m+1} z_H(\tau) \right]_1$$

to some vector $\mathbf{a}$ and the opening witness $\mathbf{a} = (a_1, \ldots, a_{m+1})$. Here a_m is a random field element that blinds cm. The prover must show that it knows an opening $\phi(X) = \sum_{j=1}^m a_j \lambda_j(X) + a_{m+1} z_H(X)$ to cm such that $a_j \in \{v_s\}_{s=1}^N$ for all $1 \leq j \leq m$. The full argument is given in 5.3 and can be divided into three steps.

First, the prover considers the subset $I \subset [N]$ such that for all $j = 1, \ldots, m$, $a_j = v_s$ for some $s \in I$, and constructs the subvector $\mathbf{v}_I = (v_s)_{s\in I}$ of $\mathbf{v}$. It commits to it in the Lagrange basis corresponding to $\{\mathsf{k}_s\}_{s\in I}$; namely, $C_I(X) = \sum_{s\in I} v_s \eta_s(X)$. Basically, the prover isolates the elements of $\mathbf{v}$ that will compare with $\mathbf{a}$ so they can work with polynomials of smaller degree.

To convince the verifier that all the elements in $C_I(X)$ are elements of $C(X)$, it provides commitments to $z_I(X), Q_1(X)$ such that

$$C(X) - C_I(X) = z_I(X) Q_1(X). \tag{5.2}$$

Here is the place where the precomputation is used: $C(X)$ has degree N and so does $Q_1(X)$. In order to compute a commitment to $Q_1(X)$ in time independent from N,

we use the aggregation method described in 2.5.3, which consist on performing a linear combination of the pre-computed $[Q_s]_1$ such that $s \in I$.

The challenge now is hiding $C_I(X)$ and $z_I(X)$ from the verifier without breaking soundness. For $C_I(X)$ we use standard zero knowledge techniques and add a term of the form $r(X)z_I(X)$, where $r(X)$ is a polynomial with random coefficients. The prover also blinds $z_I(X)$, but upon sending $[z_I]_1$ needs to provide a zero-knowledge proof of well formation of it; indeed, that it is a commitment to the vanishing polynomial of some subset $\mathbb{K}_I$ of $\mathbb{K}$.

We divide the proof of well formation of $z_I(X)$ in two steps. First, the prover creates the polynomial $f(X) = \sum_{j=1}^{m} \mathsf{k}_{s_j} \lambda_j(X)$ of degree $m-1$ whose coefficients are the roots of unity $\{\mathsf{k}_s\}_{s\in I}$ in some order and including repetitions. They then prove, in zero knowledge, its well formation. For that, they demonstrate that for all $\mathsf{h}_j \in \mathbb{H}$ it is the case that $(f(\mathsf{h}_j))^N = 1$, via a call to a subprotocol Π_{unity} that we describe in 5.4.1. This guarantees that $f(X)$ is a commitment to elements in $\mathbb{K}$.

Later, on input a commitment to $f(X)$ as above and given that $f(X)$ passes the verification of Π_{unity}, we prove well formation of $z_I(X)$ by showing that it is a polynomial that vanishes at all the coefficients of $f(X)$ in the basis $\{\lambda_j(X)\}_{j=1}^m$. For that, the prover sends $Q_2(X)$ such that

$$z_I(f(X)) = z_H(X)Q_2(X), \text{ for some polynomial } Q_2(X). \tag{5.3}$$

Finally, note that $C_I(X)$ uses an unknown-to-the-verifier Lagrange basis, which is $\{\eta_s(X)\}_{s\in I}$. So the last step of our argument consists on linking the commitment to $C_I(X)$ with $[\phi(\tau)]_1$, which is an input to the argument and a commitment to $\mathbf{a}$ in a known basis. The prover does so by providing $Q_3(X)$ such that

$$C_I(f(X)) - \phi(X) = z_H(X)Q_3(X). \tag{5.4}$$

In order to achieve zero-knowledge, upon receiving an aggregation challenge χ from the verifier, prover actually provides one commitment $[Q_2]_1 + \chi[Q_3]_1$ to prove equations 5.3 and 5.4 together.

Note that for equation 5.2 to be satisfied, $C_I(X)$ cannot take more than once each of the coefficients of $C(X)$. On the other hand, when linking $C_I(X)$ and $\phi(X)$ through equation 5.4, we can only prove that all the coefficients of $\phi(X)$ in the basis $\{\lambda_j(X)\}_{j=1}^m$ are also coefficients of $C_I(X)$ in the basis $\{\eta_s(X)\}_{s\in I}$, but we design the scheme so the prover cannot say in which order or how many times each of them appears. At the end, what we get, is a lookup table argument that assures that some element $[\phi(\tau)]_1$ is a commitment in the Lagrange basis $\{\lambda_j(X)\}_{j=1}^m$ to some vector $\mathbf{a} = (a_1, \ldots, a_m)$ where for all $j = 1, \ldots, m$ there exists some $s_j \in I$ such that $a_j = v_{s_j}$, i.e., a lookup table for potentially repeated indexes.

We describe the protocol in Fig. 5.2. PC refers to the KZG vector commitment scheme of Section 2.5.2.

Common input: $C = [C(\tau)]_1$, for $C(X) = \sum_{s=1}^{N} v_s \rho_s(X)$ and $\mathsf{cm} = [\phi(x)]_1$.

Prover: Take as input srs and $\phi(X)$ and proof $[Q(\tau)]_2$ attesting that $\{v_s\}_{s\in I}$ are openings of C. I.e., a commitment to $Q(X) = \frac{C(X) - \sum_{s\in I} v_s \eta_s(X)}{\prod_{s\in I}(X - \mathsf{k}_s)}$.

Choose blinders $r_1, r_2, r_3, r_4, r_5, r_6, r_7 \leftarrow \mathbb{F}$ uniformly at random.

For $\mathbb{K}_I = \{\mathsf{k}_s\}_{s\in I}$, compute interpolation polynomials $\{\eta_s(X)\}_{s\in I}$ and $z_I(X) = r_1 \prod_{s\in I}(X - \mathsf{k}_s)$ and $C_I(X) = \sum_{s\in I} v_s \eta_s(X) + (r_2 + r_3 X + r_4 X^2) z_I(X)$.

Find $[Q_1(\tau)]_2 = [r_1^{-1} Q(\tau) - (r_2 + r_3\tau + r_4\tau^2)]_2$ such that

$$C(X) - C_I(X) = z_I(X) Q_1(X).$$

Define k_{s_j} as the jth element in $\{\mathsf{k}_s\}_{s\in I}$ and compute

$$f(X) = \sum_{j=1}^{m} \mathsf{k}_{s_j} \lambda_j(X) + (r_5 + r_6 X + r_7 X^2) z_H(X).$$

Compute a proof π_{unity} that $[F]_1$ has been correctly computed as in 5.4

Send $[C_I]_1 = [C_I(\tau)]_1$, $[z_I]_1 = [z_I(\tau)]_1$, $[F]_1 = [f(\tau)]_1$, $[Q_1]_2 = [Q_1(\tau)]_2$, π_{unity}.

Verifier: Send challenge $\chi \in \mathbb{F}$

Prover: Find $Q_2(X)$ such that $z_I(f(X)) + \chi(C_I(f(X)) - \phi(X)) = z_H(X) Q_2(X)$ and output $[Q_2]_1 = [Q_2(\tau)]_1$.

Verifier : Send challenge $\alpha \in \mathbb{F}$

Prover : Compute

$$p_1(X) \leftarrow z_I(X) + \chi C_I(X)$$

$$p_2(X) \leftarrow z_I(f(\alpha)) + \chi(C_I(f(\alpha)) - \phi(X)) - z_H(\alpha) Q_2(X)$$

$$(v_1, \pi_1) \leftarrow \mathsf{PC.Open}(\mathsf{srs}_{\mathsf{PC}}, f(X), \deg = \bot, \alpha)$$

$$(v_2, \pi_2) \leftarrow \mathsf{PC.Open}(\mathsf{srs}_{\mathsf{PC}}, p_1(X), \deg = \bot, v_1)$$

$$(0, \pi_3) \leftarrow \mathsf{PC.Open}(\mathsf{srs}_{\mathsf{PC}}, p_2(X), \deg = \bot, \alpha)$$

Output $\big(v_1, v_2, \pi_1, \pi_2, \pi_3\big)$.

Verifier : Compute $[P_1]_1 \leftarrow [z_I]_1 + \chi[C_I]_1$ and $[P_2]_1 \leftarrow v_2 - \chi\mathsf{cm} - z_H(\alpha)[Q_2]_1$.

Accept if and only if (i) $V_{\pi_{\mathsf{unity}}}$ accepts, (ii)

$$1 \leftarrow \mathsf{PC.Verify}\big(\mathsf{srs}_{\mathsf{PC}}, [F]_1, \deg = \bot, \alpha, v_1, \pi_1\big)$$

$$1 \leftarrow \mathsf{PC.Verify}\big(\mathsf{srs}_{\mathsf{PC}}, [P_1]_1, \deg = \bot, v_1, v_2, \pi_2\big)$$

$$1 \leftarrow \mathsf{PC.Verify}\big(\mathsf{srs}_{\mathsf{PC}}, [P_2]_1, \deg = \bot, \alpha, 0, \pi_3\big),$$

and (iii),

$$e\big([C]_1 - [C_I]_1, [1]_2\big) = e\big([z_I]_1, [Q_1]_2\big) \tag{5.5}$$

Figure 5.3: Lookup table that uses a proof for $\mathsf{R}_{\mathsf{unity}}$ as blackbox.

Theorem 21. *Suppose that the argument of Fig. 5.3 is instantiated with a knowledge-sound scheme for relation $\mathcal{R}_{\mathsf{unity}}$. Then in the AGM with non-programmable ROs, either the argument of Fig. 5.3 implies linkability for the vector commitment schemes of C and cm, or there exists an adversary that breaks the qSDH assumption.*

Proof. We will proceed through a series of games to show that the protocol defined in Fig. 5.3 satisfies linkability as defined in Def. 23. Let $\mathcal{A}$ be an arbitrary PPT adversary in the linkability game with advantage $\mathsf{Adv}^{\mathsf{linkability}}_{\mathcal{A}}(\lambda)$. We define Game_1, Game_2 and specify reductions $\mathcal{B}_1$ and $\mathcal{B}_2$ such that

$$\mathsf{Adv}^{\mathsf{linkability}}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{qSDH}}_{\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{k\text{-}sound}}_{\mathcal{B}_2}(\lambda) + \mathsf{negl}(\lambda)\,.$$

Let us transition from the linkability game to a game Game_1. Game_1 behaves as linkability except that when $\mathcal{A}$ returns v_1, v_2, Game_1 checks whether $f(\alpha) = v_1$, $p_1(v_1) = v_2$, and $p_2(\alpha) = 0$, for $f(X), p_1(X), p_2(X)$, the algebraic representations of $[F]_1, [P_1]_1 = [z_I]_1 + \chi[C_I]_1$, and $[P_2]_1 = v_2 - \chi\mathsf{cm} - z_H(\alpha)[Q_2]_1$. If not then Game_1 aborts. We design $\mathcal{B}_1$ such that

$$\mathsf{Adv}^{\mathsf{linkability}}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{qSDH}}_{\mathcal{B}_1}(\lambda)$$

Indeed, assume that $\mathcal{A}$ succeeds against linkability but not Game_1. Then this corresponds to the case where $\mathcal{A}$ returns verifying v_1, v_2, π_1, π_2, π_3 but the equality does not hold for some $p(X) \in \{f(X), p_1(X), p_2(X)\}$. Thus $\mathcal{B}_1$ takes as input a challenge $[y_1]_1, \ldots, [y_q]_1$ and runs the following reduction $\mathcal{B}_{KZG}$ as a subroutine. The $\mathcal{B}_{KZG}$ runs the adversary $\mathcal{A}$ against Game_0 over an srs in which $[\tau]_1 = [y_1]_1$. Whenever the adversary wins the Game_0 but not the Game_1 game, then $\mathcal{B}_{KZG}$ returns the KZG opening

$$(v, \pi) \text{ and } (p(\alpha), [(p(X) - p(\alpha))/(\tau - \alpha)]_1)$$

for $(v, p(X))$ corresponding to either $(v_1, f(X))$, $(v_2, p_1(X))$, $(v_3, p_2(X))$ and π the corresponding proof. Then $\mathcal{B}_1$ can extract a solution from these openings following the proof in Theorem 1 in [KZG10].

Now let us transition to a new game. Game_2 behaves identically except that when $\mathcal{A}$ returns $[F]_1$, then Game_2 checks whether its algebraic representation $f(X)$ is such that $f(\mathsf{h}_j)^N = 1$ for all j. If not then Game_2 aborts. We design $\mathcal{B}_2$ such that

$$\mathsf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{k\text{-}sound}}_{\mathcal{B}_2}(\lambda)$$

Assume that $\mathcal{A}$ succeeds against Game_1 but not Game_2. Then $\mathcal{B}_2$ chooses $[F]_1 = [f(\tau)]_1$ in its own game and uses it as input to run $\mathcal{A}$. When $\mathcal{A}$ returns π_{unity}, $\mathcal{B}_2$ forwards it and wins knowledge-soundness of Π_{unity} whenever $\mathcal{A}$ succeeds.

Next we transition to a game Game_3 that behaves as Game_2 except that when $\mathcal{A}$ returns its proof, Game_3 checks whether $C(X) - C_I(X) = z_I(X)Q_1(X)$, for

$C(X), C_I(X), z_I(X), Q_1(X)$ the algebraic representations of $[C]_1, [C_I]_1, [Q_1]_2, [z_I]_1$. If not then Game_3 aborts. We design $\mathcal{B}_3$ such that

$$\mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{Game}_3}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{qSDH}}_{\mathcal{B}_3}(\lambda)$$

The $\mathcal{B}_3$ takes as input a challenge $[y_1]_1, \ldots, [y_q]_1$ and runs the adversary $\mathcal{A}$ against Game_2 over an srs in which $[\tau]_1 = [y_1]_1$. Whenever the adversary wins the Game_2 but not the Game_3 game, then $\mathcal{B}_3$ learns

$$d(X) = C(X) - C_I(X) - z_I(X)Q_1(X)$$

such that $d(\tau) = 0$ and $d(X) \neq 0$. Thus $\mathcal{B}_3$ returns $(1, [1/(\tau - 1)]_1)$ as a valid qSDH solution.

Finally we show that the probability that Game_3 returns 1 but that for some $j \in [m]$, and for $\mathbf{v}$ such that $C(X) = \sum_{s=1}^{N} v_s \rho_s(X)$,

$$\phi(\mathsf{h}_j) \notin \mathbf{v}$$

is negligible.

Recall that $p_2(\alpha) = v_2 - \chi\mathsf{cm} - z_H(\alpha)Q_2(\alpha) = z_I(v_1) + \chi C_I(v_1) - \chi\mathsf{cm} - z_H(\alpha)Q_2(\alpha) = z_I(f(\alpha)) + \chi C_I(f(\alpha)) - \chi\mathsf{cm} - z_H(\alpha)Q_2(\alpha) = 0$. First, because α has been sent by the verifier after the prover commits to $\phi(X), z_I(X), f(X), Q_2(X)$ and $C_I(X)$, we have that

$$z_I(f(X)) + \chi C_I(f(X)) - \chi\phi(X) - z_H(X)Q_2(X) = 0$$

for all X except with negligible probability. Further, because χ has been sent by the verifier after the prover commits, we have that there exists $Q_{2,1}(X)$ and $Q_{2,2}(X)$ such that

$$\begin{aligned} 0 &= z_I(f(X)) - z_H(X)Q_{2,1}(X) \\ 0 &= C_I(f(X)) - \phi(X) - z_H(X)Q_{2,2}(X) \end{aligned}$$

except with negligible probability.

Thus,

$$z_I(f(\mathsf{h}_j)) = z_I(\mathsf{k}_{s_j}) = 0 \text{ for all } j = 1, \ldots, m.$$

and $z_I(X) = \prod_{j=1}^{m}(X - \mathsf{k}_{s_j})\hat{z}(X) = \prod_{s \in I}(X - \mathsf{k}_s)\hat{z}(X)$, for some polynomial $\hat{z}(X)$. From the second equation we also we have that

$$C_I(f(\mathsf{h}_j)) = \phi(\mathsf{h}_j) \ \forall \ j \in [m], \text{i.e., } C_I(\mathsf{k}_{s_j}) = \phi(\mathsf{h}_j).$$

Using

$$C(f(X)) - C_I(f(X)) = z_I(f(X))Q_1(f(X))$$

we hence gets that

$$0 = C(f(\mathsf{h}_j)) - C_I(f(\mathsf{h}_j)) = C(\mathsf{k}_k) - \phi(\mathsf{h}_j)$$

which concludes the proof.

□

Avoiding Repetitions. To convince the verifier that each element in $\mathbf{v}$ has been chosen at most once, we run our protocol for lookup tables but with two additional steps: (i) the prover declares $\deg_z$ and proves $\deg(z_I) = \deg_z$, and (ii) they send a commitment to $\hat{f}(X) = f(X)/\prod_{j=\deg_z+1}^{m}(X - \mathsf{h}_j)$, whose well formation can be checked by the verifier performing the pairing equation $e([\hat{f}(\tau)]_1, \sum_{j=\deg_z+1}^{m}[\tau - \mathsf{h}_j]_2) = e([F]_1, [1]_2)$. These two steps combined prove that $f(X)$ has no repeated indexes and thus, our argument will guarantee that cm is a commitment to a polynomial $\phi(X)$ where each element corresponds to a unique coefficient in $C(X)$.

Range Proofs. We argue our protocol can be adapted to work as a range proof where prover's work is $m^2 + m\log(\log(M))$ when showing some element v whose binary representation has m non-zero elements is in $[M]$. We explain next the high level idea of the scheme, omitting the necessary details to achieve zero-knowledge. Let $N - 1 = \log(M)$ and C be a commitment to $\mathbf{v} = (1, 2, 2^2, \ldots, 2^{N-1} = M)$, i.e, $\mathsf{C} = [C(\tau)]_1$ for $C(X) = \sum_{s=1}^{N} 2^{s-1}\rho_s(X)$. The prover will select from $\mathbf{v}$ the powers of 2 corresponding to the binary representation of v, that is, will take all the $s_j \in J \subset [N-1]$ such that $\sum_{j\in J} 2^{s_j} = v$. By using the two extra steps described above, the prover convinces the verifier that there are no repeated coefficients of $\mathbf{v}$ in $\phi(X)$. Finally, note that, because the $\{\lambda_j(X)\}_{j=1}^{m}$ are Lagrange polynomials corresponding to a set of roots of unity, $\phi(0) = m^{-1}\sum_{j=1}^{m} 2^{s_j} = m^{-1}v$; that is, the prover opens $\phi(X)$ in 0 to the verifier and convinces them that $v \in [M]$. When implemented with Caulk as it is, the scheme above lacks of practicality since m would be linear in $\log(N)$ in most of the cases. We aim to remove the $\log(N)$ factor in prover's work for **Caulk** and be able to implement these range proofs.

Subtables There is another nice feature that can be derived by the protocol in Fig. 5.3 and is the creation of sub-lookup tables. Namely, for some $I \subset [N]$, prover generates $t(X) = \prod_{s\in I}(X - v_s)$. To prove well formation of it, after having some $C_I(X)$ that has been proven correct, it shows that there exists some $Q_3(X)$ such that

$$t(C_I(X)) = z_H(X)Q_3(X).$$

Then, for any polynomial $a(X)$ of degree up to $m-1$, if there exists $Q_4(X)$ such that

$$t(a(X)) = z_H(X)Q_4(X),$$

then the coefficients of $a(X)$ in the basis $\{\lambda_j(X)\}_{j=1}^{m}$ are coefficients of $C_I(X)$ in basis $\{\eta_s(X)\}_{s\in I}$, with no specific order and potential repetitions. Note that, once the subtable is created, prover's work is quadratic in, potentially small, m and does not need to store pre-computed proofs.

5.4.1 Multi-Unity Proof or Proving well formation of $f(X)$

The aim of this section is to prove in zero-knowledge that $f(X) = \sum_{j=1}^{n} \mathsf{k}_{i_j} \lambda_j(X) = \sum_{j=1}^{n} f_j \lambda_j(X)$, where k_{i_j} is the j-th element in $\mathbb{K}_I$, is well formed. Namely, that $f(X)$ is such that all its coefficients are elements in $\mathbb{K}$ and thus, they are all Nth roots of unity, or what is the same, that $f_j^N = 1$ for all $j = 1, \ldots, m$.

For this argument, we will consider another group of roots of unity of size $n = \log(N)$ $\mathbb{U}_n = \{\mathsf{u}_1, \ldots, \mathsf{u}_n\}$, where $\mathsf{u}_i = \nu^{i-1}$ for $\nu^n = 1$. We denote the Lagrange interpolation polynomials $\{\mu_i(X)\}_{i=1}^n$ and vanishing polynomial $z_U(X)$.

Soundness. Let $\mathbf{f}_0 = (f_1, \ldots, f_m) \in \mathbb{F}^n$ be the vector whose elements are the coefficients of $f(X)$, and define $\mathbf{f}_1 = \mathbf{f}_0 \circ \mathbf{f}_0 = (f_1^2, \ldots, f_m^2)$. Now, define $\mathbf{f}_2 = \mathbf{f}_1 \circ \mathbf{f}_1$ and see that $\mathbf{f}_2 = (f_1^{2^2}, \ldots, f_m^{2^2})$. The intuition of the protocol below is that if the prover constructs vectors $\{\mathbf{f}_i\}_{i=0}^n$ and proves that (i) $\mathbf{f}_0$ consists on the coefficients of $f(X)$, (ii) $\mathbf{f}_i = \mathbf{f}_{i-1} \circ \mathbf{f}_{i-1}$ meaning that $\mathbf{f}_i = (f_1^{2^i}, \ldots, f_m^{2^i})$ for all $i = 1, \ldots, n-1$ and (iii) $\mathbf{f}_{n-1} \circ \mathbf{f}_{n-1} = \mathbf{1}$, we have that all the coefficients f_j are Nth roots of unity.

We will work with encodings as polynomials rather than vectors, so the prover sets $f_0(X) = f(X) = \sum_{j=1}^{m} \mathsf{k}_{s_j} \lambda_j(X) = \sum_{j=1}^{m} f_j \lambda_j(X)$, $f_n(X) = 1$, and shows to the verifier that each of the following equations hold:

$$\begin{aligned} f(X)f(X) - f_1(X) &= z_H(X)Q_1(X), \\ f_1(X)f_1(X) - f_2(X) &= z_H(X)Q_2(X), \\ &\vdots \\ f_{n-1}(X)f_{n-1}(X) - 1 &= z_H(X)Q_n(X), \end{aligned}$$

Aggregation. Naturally, the equations above can be checked all together using standard aggregation techniques, i.e., that for some uniformly sampled field elements $\gamma_1, \ldots, \gamma_n$,

$$\left(\gamma_1 f^2(X) + \sum_{i=2}^{n} \gamma_i f_{i-1}^2(X)\right) - \left(\sum_{i=1}^{n-1} \gamma_i f_i(X) + \gamma_n 1\right) = z_H(X) \sum_{i=1}^{n} \gamma_i Q_i(X). \quad (5.6)$$

Importantly, the verifier must be the one computing the terms $\gamma_1 f^2(X)$ and $\gamma_n 1$ in order to check that the coefficients in all the other polynomials $f_i(X)$ are indeed powers of thus in $f(X)$, and that their Nth power is 1.

This aggregation, though, will not be performed by the verifier. Instead, we will have the prover committing to the polynomials by using extra powers of X and

then partially open them at some challenge sent by the verifier. For instance, the prover commits to $U(X) = \sum_{i=1}^{n} f_i(X)\mu_i(X^m)$ and upon receiving challenge β from the verifier, partially opens $U(X)$ at $X^m = \beta$, obtaining $U_\beta(X) = \sum_{i=1}^{n} f_i(X)\mu_i(\beta)$. That is, an aggregation as in Eq. 5.6 where $\gamma_i = \mu_i(\beta)$.

Well formation. The equation to be proven then is

$$\begin{aligned}\left(f^2(X)\mu_1(X^m) + \sum_{i=2}^{n} f_{i-1}^2(X)\mu_i(X^m)\right) \\ - \left(\sum_{i=1}^{n-1} f_i(X)\mu_i(X^m) + \mathsf{id}(X)\mu_n(X^m)\right) = z_H(X)Q(X),\end{aligned} \tag{5.7}$$

for some polynomial $Q(X)$. For simplicity, we will make a change of notation and set $Y = X^m$.

It is not difficult to see that two polynomials of the form $\sum_{i=1}^{n} p_s^2(X)\mu_i(Y)$ and $\left(\sum_{i=1}^{n} p_s(X)\mu_i(Y)\right)^2$ are the same modulus $z_U(Y)$, or for our case, that there exists a polynomial $h_1(X, Y)$ such that

$$\begin{aligned}\left(f^2(X)\mu_1(Y) + \sum_{i=2}^{n} f_{i-1}^2(X)\mu_i(Y)\right) \\ = \left(f(X)\mu_1(Y) + \sum_{i=2}^{n} f_{i-1}(X)\mu_i(Y)\right)^2 - z_U(Y)h_1(X, Y),\end{aligned}$$

and thus the prover can convince the verifier of the well formation of $f^2(X)\mu_1(Y) + \sum_{i=2}^{n} f_{i-1}^2(X)\mu_i(Y)$ given $\sum_{i=2}^{n} f_{i-1}(X)\mu_i(Y)$ and $h_1(X, Y)$. It can also convince them of correctness of partial evaluations by performing a KZG opening, and of the correctness of Eq. 5.6 by providing $h_2(X, Y) = \sum_{i=1}^{n} Q_i(X)\mu_i(Y)$.

The challenge is then to relate $U_1(X, Y) = \sum_{i=1}^{n-1} f_i(X)\mu_i(Y)$ with $U_2(X, Y) = \sum_{i=2}^{n} f_{i-1}(X)\mu_i(Y)$.

Recall that because $\mathbb{U} = \{\mathsf{u}_1, \ldots, \mathsf{u}_n\}$ is a set of roots of unity, where $\mathsf{u}_i = \nu^{i-1}$, so we have

$$\mu_i(Y\nu^{-1}) = \mu_{i+1}(Y) \quad \text{and} \quad \mu_n(Y\nu^{-1}) = \mu_1(Y).$$

Then, $U_1(X, Y)$ is a *shift* of the coefficients for $\mu_i(Y)$ in $U_2(X, Y)$, since $U_1(X, Y\nu^{-1}) + f(X)\mu_1(Y) = U_2(X, Y) + 1\mu_1(Y)$.

Namely, the verifier can compute $U_2(X, Y)$ from $U_1(X, Y)$ but, as it needs to include the therm $f(X)\mu_1(X)$, it will rather have access to a polynomial $\bar{U}(X, Y) = \sum_{i=2}^{n} f_{i-1}(X)\mu_i(Y)$, and check that $(\bar{U}(X, Y\nu^{-1}) + f(X)\mu_1(Y))^2$, modulus $z_U(Y)$ and $z_H(X)$, equals $\bar{U}(X, Y) + \mathsf{id}(X)\mu_1(Y)$.

The protocol is shown in Figure 5.4. As before, PC denotes the KZG polynomial commitment scheme, as explained in Section 2.5.1.

Common input: $[F]_1$ where $[F]_1 = [f_0(\tau)]_1$

Prover: Take as input srs and $f(X)$

Samples blinders $t_1, \ldots t_n \leftarrow \mathbb{F}$.

For $i = 1, \ldots, n$, define $f_i(X) = \sum_{j=1}^{n} \left(\mathsf{k}_{i_j}\right)^{2^i} \lambda_j(X) + t_i z_H(X)$,

Define $U(X, Y) = \sum_{i=1}^{n} f_{i-1}(X)\mu_i(Y)$.

Define $\bar{U}(X, Y) = U(X, Y) - f(X)\mu_i(Y)$

Define $h_2(X) = \sum_{i=1}^{n} \mu_i(Y) Q_s(X)$ for $Q_s(X) = (f_{i-1}^2(X) - f_i(X))/z_H(X)$

Output $\left([\bar{U}]_1 = [\bar{U}(\tau^n, \tau)]_1, [h_2]_1 = [h_2(\tau^n, \tau)]_1\right)$

Verifier: Send challenge $\alpha \in \mathbb{F}$

Prover: Define $h_1(Y) \leftarrow \left(U^2(\alpha, Y) - \sum_{i=1}^{n} f_{i-1}^2(\alpha)\mu_i(Y)\right)/z_U(Y)$

Output $[h_1]_1 = [h_1(\tau)]_1$

Verifier: Send challenge $\beta \in \mathbb{F}$

Prover:

$$p(Y) \leftarrow (U^2(\alpha, \beta) - h_1(Y) z_U(\beta)) - \bar{U}(\alpha, \beta\nu) + \mathsf{id}(\alpha)\mu_n(\beta)) - z_H(\alpha) h_2(\alpha, Y)$$

$$(v_1, \pi_1) \leftarrow \mathsf{PC.Open}\left(\mathsf{srs}, f(X), \deg = \bot, X = \alpha\right)$$

$$([\bar{U}(\alpha, \tau)]_1, \pi_2) \leftarrow \mathsf{PC.Open}\left(\mathsf{srs}, \bar{U}(X, Y), \deg = \bot, X = \alpha\right)$$

$$([h_2(\alpha, \tau)]_1, \pi_3) \leftarrow \mathsf{PC.Open}\left(\mathsf{srs}, h_2(X, Y), \deg = \bot, X = \alpha\right)$$

$$((0, v_2, v_3), \pi_4) \leftarrow \mathsf{PC.Open}\left(\mathsf{srs}, \bar{U}(\alpha, Y), \deg = n - 1, Y = (1, \beta, \beta\nu)\right)$$

$$(0, \pi_5) \leftarrow \mathsf{PC.Open}\left(\mathsf{srs}, p(Y), \deg = n - 1, Y = \beta\right)$$

Set $[\bar{U}_\alpha]_1 = [\bar{U}(\alpha, \tau)]_1, [h_{2,\alpha}]_1 = [h_2(\alpha, \tau)]_1$.

Output $\left([\bar{U}_\alpha]_1, [h_{2,\alpha}]_1, v_1, v_2, v_3, \pi_1, \pi_2, \pi_3, \pi_4, \pi_5\right)$

Verifier: Set $U \leftarrow v_1\mu_i(\beta) + v_2$, $[P]_1 \leftarrow U^2 - [h_1]_1 z_U(\beta) - (v_3 + \mathsf{id}(\alpha)\mu_n(\beta)) - z_H(\alpha)[h_{2,\alpha}]_1$. Accept if and only if

$$1 = \mathsf{PC.Verify}\left(\mathsf{srs}_{\mathsf{PC}}, [F]_1, \deg = \bot, X = \alpha, v_1, \pi_1\right)$$

$$1 = \mathsf{PC.Verify}\left(\mathsf{srs}_{\mathsf{PC}}, [\bar{U}]_1, \deg = \bot, X = \alpha, [\bar{U}_\alpha]_1, \pi_2\right)$$

$$1 = \mathsf{PC.Verify}\left(\mathsf{srs}_{\mathsf{PC}}, [h_2]_1, \deg = \bot, X = \alpha, [h_{2,\alpha}]_2, \pi_3\right)$$

$$1 = \mathsf{PC.Verify}\left(\mathsf{srs}_{\mathsf{PC}}, [\bar{U}_\alpha]_1, \deg = n - 1, Y = (1, \beta, \beta\nu), (0, v_2, v_3), \pi_4\right)$$

$$1 = \mathsf{PC.Verify}\left(\mathsf{srs}_{\mathsf{PC}}, [P]_1, \deg = n - 1, Y = \beta, 0, \pi_5\right)$$

Figure 5.4: Argument for proving that some polynomial $f(X)$ has Nth roots of unity as coefficients in the basis $\{\lambda_j(X)\}_{j=1}^{n}$.

Theorem 22. *The protocol in Figure 5.4 is a knowledge-sound argument for relation* $\mathsf{R}_{\mathsf{unity}}$ *under the algebraic group model and random oracle model if the qSDH, qDHE, and qSFrac assumptions hold.*

Proof. We proceed through a series of games to show that the protocol defined in 5.3 satisfies knowledge soundness. We set Game_0 to be the knowledge soundness game as defined in 1 and consider an algebraic adversary $\mathcal{A}$ against it which has advantage $\mathsf{Adv}^{\mathsf{k\text{-}sound}}_{\mathcal{A}}(\lambda)$. We define Game_1 and Game_2 and specify reductions $\mathcal{B}_1$ and $\mathcal{B}_2$ such that

$$\begin{aligned}\mathsf{Adv}^{\mathsf{k\text{-}sound}}_{\mathcal{A}}(\lambda) &= \mathsf{Adv}^{\mathsf{Game}_0}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{qSDH}}_{\mathcal{B}_1}(\lambda) \\ &\leq \mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{qDHE}}_{\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{qSDH}}_{\mathcal{B}_1}(\lambda) \\ &\leq \mathsf{Adv}^{\mathsf{Game}_3}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{qSDH}}_{\mathcal{B}_3}(\lambda) + \mathsf{Adv}^{\mathsf{qDHE}}_{\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{qSDH}}_{\mathcal{B}_1}(\lambda) \\ &\leq \mathsf{Adv}^{\mathsf{qSFrac}}_{\mathcal{B}_3}(\lambda) + \mathsf{Adv}^{\mathsf{qDHE}}_{\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{qSDH}}_{\mathcal{B}_1}(\lambda) + \mathsf{negl}(\lambda)\,.\end{aligned}$$

In Game_0 the adversary will return $[F]_1 = [f(\tau)]$ along with a proof. We define Game_1 identically to Game_0, but after the adversary returns $[F]_1$ and a proof, Game_1 additionally checks whether for $f(X), \bar{U}_\alpha(X), p(X)$ the algebraic representations of $[F]_1, [\bar{U}_\alpha]_1$, $[P]_1$, it is true that $u(\alpha) = v_1$, $\bar{U}_\alpha(1) = 0$, $\bar{U}_\alpha(\beta) = v_2$, $\bar{U}_\alpha(\beta\nu) = v_3$, and $p(\beta) = 0$; and it aborts if one of the conditions does not hold.

The redution $\mathcal{B}_1$ takes as input the challenge $[y_1]_1, \ldots, [y_q]_1$. It runs the following reduction $\mathcal{B}_{\mathsf{KZG}}$ as a subroutine. The $\mathcal{B}_{\mathsf{KZG}}$ runs the adversary $\mathcal{A}$ against Game_0 over an srs in which $[\tau]_1 = [y_1]_1$. Whenever $\mathcal{A}$ returns an output which wins Game_0, if $(p(X), \mathbf{v}, \mathbf{z})$ for some $(p(X), \mathbf{v}, \mathbf{z}) \in \{(f(X), v_1, \alpha), (\bar{U}_\alpha(X), (1, \beta, \nu\beta), (0, v_2, v_3)), (p(X), \beta, 0)\}$ is such that $f(v_i) \neq z_i$, then $\mathcal{B}_{KZG}$ computes $f(z) = v'$ and a valid proof π'. It outputs $([f(\tau)]_1, \mathbf{z}, \mathbf{v}, \pi)$ and $([f(\tau)]_1, \mathbf{z}, \mathbf{v}', \pi')$ and wins evaluation binding as they are both proofs that verify and open to different elements. Then $\mathcal{B}_{\mathsf{qSDH}}$ can extract a qSDH solution from these openings following the proof in Theorem 3 of [KZG10]. Thus

$$\mathsf{Adv}^{\mathsf{k\text{-}sound}}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{Game}_0}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{qSDH}}_{\mathcal{B}_1}(\lambda)$$

Now Game_2 behaves identically as Game_1 except that it additionally checks whether $\deg(\bar{U}_\alpha) \leq n - 1$ and $\deg(h_2) \leq n - 1$. If it is not the case, it aborts. Suppose $\mathcal{A}$ returns either $\deg(\bar{U}_\alpha) = n - 1 + d$ or $\deg(h_2) = n - 1 + d$ for some $d > 0$. We argue the advantage of $\mathcal{A}$ in Game_1 and Game_2 is the same unless we can build an adversary $\mathcal{B}_2$ that succeeds against qDHE. The $\mathcal{B}_2$ takes as input the challenge $[y_1]_1, \ldots, [y_{q+d-1}]_1$ and runs the adversary $\mathcal{A}$ against Game_1 over an srs in which $[\tau]_1 = [y_1]_1$. Whenever $\mathcal{A}$ returns an output which wins the Game_1 game, if $(p(X), \mathbf{v}, \mathbf{z})$ for

$$(p(X), \mathbf{v}, \mathbf{z}) \in \{(\bar{U}_\alpha(X), (1, \beta, \nu\beta), (0, v_2, v_3)), (p(X), \beta, 0)\}$$

is such that $p(X)$ has degree greater than $n-1$, then the corresponding proof $\pi = [q(\tau)]_1$ has a representation $q(X)$ has degree $q+1$. Thus $\mathcal{B}_2$ succeeds in returning $[\pi - \sum_{i=0}^{q-1} \tau^i]_1$ and

$$\mathsf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{qDHE}}_{\mathcal{B}_2}(\lambda)$$

We define Game_3 identically to Game_2, but after the adversary returns $[F]_1$ and a proof, Game_3 additionally checks whether for $\bar{U}(X), h_2(X), \bar{U}_\alpha(X), h_{2,\alpha}(X)$ the algebraic representations of $[\bar{U}]_1, [\bar{U}_\alpha]_1, [h_2]_1, [h_{2,\alpha}]_1$, it is true that

$$\bar{U}_\alpha(X) = \sum_{i,j} \alpha^i \bar{U}_{ij} X^j \text{ and } h_{2,\alpha}(X) = \sum_{i,j} \alpha^i h_{2,ij} X^j$$

and it aborts if one of the conditions does not hold.

The redution $\mathcal{B}_3$ against qSFrac [GG17] takes as input the challenge $[y_1]_1, \ldots, [y_q]_1$ and runs the adversary $\mathcal{A}$ against Game_2 over an srs in which $[\tau]_1 = [y_1]_1$. Whenever $\mathcal{A}$ returns an output which wins the Game_2 game, if $(p(X), V, z)$ for

$$(p(X), \phi(X), z) \in \{(\bar{U}(X), \bar{U}_\alpha(X), \alpha), (h_2(X), h_{2,\alpha}(X), \alpha)\}$$

is such that $\phi(X) \neq \phi'(X) = \sum_{i,j} \alpha^i f_{ij} X^j$, then set π be the proof for $(p(X), \phi(X), z)$. Then $\mathcal{B}_3$ returns

$$\phi(X) - \phi'(X), (X^m - z), \pi - \left[\frac{f(\tau) - \phi'(\tau)}{\tau^n - z}\right]_1$$

We have that $\deg(\phi(X) - \phi'(X)) < \deg(X^m - z)$ because $\phi(X)$ has degree bounded by $n-1$. Hence this is as a valid solution and

$$\mathsf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{Game}_3}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{qSFrac}}_{\mathcal{B}_3}(\lambda)$$

Lets see that the advantage of $\mathcal{A}$ in Game_3 is negligible.

Consider $h_1(X), h_2(X, Y)$ the algebraic representations of $[h_1]_1$, $[h_2]_1$. We can use the equations verified by Game_1 and replace the corresponding values in $p(X)$, obtaining

$$\begin{aligned}
p(X) &= (v_1\mu_i(\beta) + v_2)^2 - h_1(X) z_U(\beta) - (v_3 + \mathsf{id}(\alpha)\mu_n(\beta)) - z_H(\alpha) h_{2,\alpha}(X) \\
&= \left(u(\alpha)\mu_i(\beta) + \bar{U}_\alpha(\beta)\right)^2 - h_1(X) z_U(\beta) - (\bar{U}_\alpha(\beta\nu) + \mathsf{id}(\alpha)\mu_n(\beta)) \\
&\quad - z_H(\alpha) h_{2,\alpha}(X) \\
&= \left(u(\alpha)\mu_i(\beta) + \bar{U}(\alpha, \beta)\right)^2 - h_1(X) z_U(\beta) - (\bar{U}(\alpha, \beta\nu) + \mathsf{id}(\alpha)\mu_n(\beta)) \\
&\quad - z_H(\alpha) h_2(\alpha, X)
\end{aligned}$$

From the fact that $p(\beta) = 0$ we get that

$$\left(u(\alpha)\mu_i(\beta) + \bar{U}(\alpha, \beta)\right)^2 - (\bar{U}(\alpha, \beta\nu) + \mathsf{id}(\alpha)\mu_n(\beta)) - z_H(\alpha) h_2(\alpha, \beta) - h_1(\beta) z_U(\beta)$$

equals 0. Since $[F]_1, [\bar{U}]_1$, $[h_1]_1$, $[h_2]_1$ have been sent by the prover before it sees challenges β, we have that except in the case where $(Y = \beta)$ is a root of the polynomial below, which happens with negligible probability, for all Y,

$$\begin{aligned} 0 \quad &= \big(u(\alpha)\mu_i(Y) + \bar{U}(\alpha, Y)\big)^2 - (\bar{U}(\alpha, Y\nu) + \mathsf{id}(\alpha)\mu_n(Y)) \\ &\quad -z_H(\alpha)h_2(\alpha, Y) - h_1(Y)z_U(Y) \end{aligned} \tag{5.8}$$

Thus we have that

$$\begin{aligned} i = 0 &\Rightarrow 0 = f^2(\alpha) - \bar{U}(\alpha, \nu^1) - z_H(\alpha)h_2(\alpha, \nu^1) \\ 1 \leq i \leq n-1 &\Rightarrow 0 = \bar{U}^2(\alpha, \nu^i) - \bar{U}(\alpha, \nu^{i+1}) - z_H(\alpha)h_2(\alpha, \nu^i) \\ i = n &\Rightarrow 0 = \bar{U}^2(\alpha, \nu^{n-1}) - \mathsf{id}(\alpha) - z_H(\alpha)h_2(\alpha, \nu^1) \end{aligned}$$

Since $[F]_1, [\bar{U}]_1$, $[h_2]_1$ have been sent by the prover before it sees challenges α, we have that except in the case where $(X = \alpha)$ is a root of the polynomial below, which happens with negligible probability, for all X,

$$\begin{aligned} i = 0 &\Rightarrow 0 = f^2(X) - \bar{U}(X, \nu^1) - z_H(X)h_2(X, \nu^1) \\ 1 \leq i \leq n-1 &\Rightarrow 0 = \bar{U}^2(X, \nu^i) - \bar{U}(X, \nu^{i+1}) - z_H(X)h_2(X, \nu^i) \\ i = n &\Rightarrow 0 = \bar{U}^2(X, \nu^{n-1}) - \mathsf{id}(X) - z_H(X)h_2(X, \nu^1) \end{aligned}$$

Over $\mathsf{h}_j \in \mathbb{H}$ we thus have that

$$\begin{aligned} i = 0 &\Rightarrow 0 = f^2(\mathsf{h}_j) - \bar{U}(\mathsf{h}_j, \nu^1) \\ 1 \leq i \leq n-1 &\Rightarrow 0 = \bar{U}^2(\mathsf{h}_j, \nu^i) - \bar{U}(\mathsf{h}_j, \nu^{i+1}) \\ i = n &\Rightarrow 0 = \bar{U}^2(\mathsf{h}_j, \nu^{n-1}) - 1 \end{aligned}$$

Together these gives us the desired requirement that $f^N(\mathsf{h}_j) = 1$ for all $j = 1, \ldots, m$ except with negligible probability. □

Theorem 23. *The protocol in Fig. 5.3 and 5.4 implies position-hiding linkability between the vector commitment schemes of* C *and* cm*, provided that the zk proof for* $\mathsf{R}_{\mathsf{unity}}$ *is instantiated with the protocol in 5.4 and provided that* $\log(N) > 6$.

Proof. We first define a simulator $\mathsf{Simulate}$ and then argue that their transcript is indistinguishable from an honest provers transcript. The $\mathsf{Simulate}$ subverts the setup algorithm such that it knows the secret τ contained in $[\tau]_1, [\tau^2]_1, \ldots, [\tau^d]$. It takes as input some instance (C, cm) and aims to generate a verifying transcript.

It samples s_1, s_2, s_3, s_4, s_5, $s_6, s_7, s_8 \leftarrow \mathbb{F}$ at random and outputs $[C_I]_1 = [s_1]_1, [z_I]_1 = [s_2]_1$, $[F]_1 = [s_3]_1$, $[Q_1]_2 = [(C - s_1)/s_2]_2$ and a simulated proof π_{unity} that we describe in the next paragraph. After receiving χ it outputs $[Q_2]_1 = [s_4]_1$. After receiving α it outputs $v_1 = s_5$, $v_2 = s_6$. and

$$\begin{aligned} \pi_1 &= [(u - v_1)/(\tau - \alpha)]_1 \\ \pi_2 &= [(z_I + \chi C_I)/(\tau - v_1)]_1 \\ \pi_3 &= [(v_2 - \chi\mathsf{cm} - z_H(\alpha)Q_2)/(\tau - \alpha)] \end{aligned}$$

To simulate π_{unity} the simulate $\mathsf{Simulate}$ outputs $[\bar{U}]_1 = [s_7]_1$, $[h_2]_1 = [s_8]_1$. After receiving α it outputs $[h_1]_1 = [s_9]_1$. After receiving β it outputs $[\bar{U}_\alpha]_1 = [s_{10}]_1$, $[h_{2,\alpha}] = [s_{11}]_1$ and $v_1 = s_{12}$, $v_2 = s_{13}$, $v_3 = s_{14}$ and

$$\pi_1 = [(u - v_1)/(\tau - \alpha)]_1$$
$$\pi_2 = [(\bar{U} + \bar{U}_\alpha)/(\tau - \alpha)]_1$$
$$\pi_3 = [(h_2 - h_{2,\alpha})/(\tau - \alpha)]_1$$
$$\pi_4 = [\tau^{\mathsf{max\_deg}-n}(\bar{U}_\alpha + \ell(\tau))/(\tau - 1)(\tau - \beta)(\tau - \beta\nu)]_1$$
$$\pi_5 = [\tau^{\mathsf{max\_deg}-n}((v_1\mu_i(\beta) + v_2)^2 - h_1 z_U(\beta) - (v_3 + \mathsf{id}(\alpha)\mu_n(\beta)) - z_H(\alpha)h_{2,\alpha})/(\tau - \beta)]_1$$

where $\ell(\tau)$ is the polynomial that interpolates to $(0, v_2, v_3)$ at $(1, \beta\beta\nu)$.

We now argue $\mathsf{Simulate}$'s output is indistinguishable from an honest prover's output.

We consider each of the elements in 5.3 separately and argue they are identically distributed with overwhelming probability.

- $[C_I]_1$ is blinded by r_2 for the prover and s_1 for the simulator.
- $[z_I]_1$ is blinded by r_1 for the prover and s_2 for the simulator.
- $[F]_1$ is blinded by r_5 for the prover and s_3 for the simulator.
- $[Q_1]_2$ is the unique element satisfied by the pairing check for both the prover and simulator given $[C_I]_1$ and $[z_I]_1$.
- $[Q_2]_1$ is blinded by r_3 for the prover and s_4 for the simulator. Note that $r_3\frac{\chi u(\tau) z_I(u(\tau))}{z_H(\tau)}$ is non-zero with overwhelming probability.
- v_1 is blinded by r_6 for the prover and s_5 for the simulator. Note that $r_6\alpha z_H(\alpha)$ is non-zero with overwhelming probability.
- v_2 is blinded by r_4 for the prover and s_6 for the simulator. Note that $r_4 f^2 \alpha z_I(u(\alpha))$ is non-zero with overwhelming probability.
- π_1, π_2, π_3 are the unique element satisfied by the KZG opening checks for both the prover and the simulator.

Finally we consider each of the elements in 5.4 separately and argue they are identically distributed with overwhelming probability.

- $[\bar{U}]_1$ is blinded by t_1 for the prover and s_7 for the simulator.
- $[h_2]_1$ is blinded by t_2 for the prover and s_8 for the simulator. Note that there exists a $\mu_2(\tau)t_2$ term in the provers $[h_2]_1$ which is linearly independent from all other terms and thus not cancelled with overwhelming probability.

- $[h_1]_1$ is blinded by t_3 for the prover and s_9 for the simulator. Note that there is a $t_3^2 z_H^2(\alpha)\frac{\mu_i 4^2(\tau)-\mu_i 4(\tau)}{z_U(\tau)}$ term in the provers $[h_1]_1$ which is linearly independent from all other terms.

- $[\bar{U}_\alpha]_1$ is blinded by t_4 for the prover and s_{10} for the simulator. Note that there is a $t_4 z_H(\alpha)\mu_5(\tau)$ term in the provers $[\bar{U}_\alpha]_1$ which is linearly independent from all other terms.

- $[h_{2,\alpha}]_1$ is blinded by t_5 for the prover and s_{11} for the simulator. Note that there is a $\mu_2(\tau)t_2$ term in the provers $[h_{2,\alpha}]_1$ which is linearly independent from all other terms.

- v_1 is blinded by r_7 for the prover and s_{12} for the simulator.

- v_2 is blinded by t_5 for the prover and s_{13} for the simulator. Note that there is a $t_5 z_H(\alpha)\mu_6(\beta)$ term in the provers v_2 which is linearly independent from all other terms.

- v_3 is blinded by t_6 for the prover and s_{14} for the simulator. Note that there is a $t_6 z_H(\alpha)\mu_7(\beta)$ term in the provers v_3 which is linearly independent from all other terms.

- $\pi_1, \pi_2, \pi_3, \pi_4, \pi_5$ are the unique elements satisfied by the KZG opening checks for both the prover and the simulator.

□

5.5 Efficiency

In this section we describe some optimizations we apply to the protocols in Fig. 5.3 and 5.4 in order to achieve the efficiency claimed in Table 5.1.

Opening t polynomials in one point. As noted in [GWC19],[CHM+20], whenever we have t openings of different polynomials at the same point i.e. for $t = 2$ this would be of the form

$$\pi_1 \leftarrow \mathsf{PC.Open}(\mathsf{srs}_{\mathsf{PC}}, f_1(X), \deg = d, \alpha)$$
$$\pi_2 \leftarrow \mathsf{PC.Open}(\mathsf{srs}_{\mathsf{PC}}, f_2(X), \deg = d, \alpha)$$

then we can send a single opening proof π as opposed to t opening proofs $\pi_1, \ldots, \pi_t$.

Batching Pairings. We also apply standard techniques to batch pairings that share the same elements in one of the two groups. Namely, we can aggregate the equations

$$e([a]_1, [b_1]_2) = e([c_1]_1, [d]_2) \text{ and } e([a]_1, [b_2]_2) = e([c_2]_1, [d]_2),$$
$$\text{as } e([a]_1, [b_1 + \gamma b_2]_2) = e([c_1 + \gamma c_2]_1, [d]_2)$$

for γ some random field element sampled by the verifier.

Note that we can adapt KZG openings equations so they can be batched further, namely if we parse the verification pairing as $e\big([F_1]_1 - s_1 + [Q_1]_1\alpha, [1]_2\big) = e\big([Q_1]_1, [\tau]_2\big)$, then two openings of different polynomials at different points can be verified by two pairings.

Fig. 5.1 and 5.2: In Fig. 5.1 proofs have the form $([z]_2, [T]_1, [S]_2, \pi_{\mathsf{ped}}, \pi_{\mathsf{unity}})$. See that π_{ped} consists of 1 $\mathbb{G}_1$ and 2$\mathbb{F}$. In Fig. 5.2 proofs have the form $([F]_1, [Q]_1, v_1, v_2, \pi_1, \pi_2)$ which amounts to 4$\mathbb{G}_1$ and 2$\mathbb{F}$. Thus we have a total of 6$\mathbb{G}_1$, 2$\mathbb{G}_2$ and 4$\mathbb{F}$.

For the verifier, their first pairing check in Fig. 5.1 uses pairings of the form $e(*, [1]_2)$, $e(*, [z]_2)$, and $e([h]_1, *)$ amounting to 3 pairings. The Pedersen verifier uses no pairings. In Fig. 5.2 we have a KZG verifier which uses pairings of the form $e(*, [1]_2)$, $e(*, [\tau]_2)$, and a pairing check that uses pairings of the form $e(*, [1]_2)$, $e(*, [z]_2)$, and $e(*, [\tau]_2)$. Thus we can batch the pairing checks to get a total of 4 unique pairings over the two constructions.

Fig. 5.3 and 5.4: In Fig. 5.3 proofs have the form $([C_I]_1, [z_I]_1, [u]_1, [Q_1]_2, [Q_2]_1, v_1, v_2, \pi_1, \pi_2, \pi_3, \pi_{\mathsf{unity}})$. Here the π_1, π_3 are both openings at the same α and can be batched into one proof. Thus there are 7$\mathbb{G}_1$, 1$\mathbb{G}_2$ and 2$\mathbb{F}$ in addition to the π_{unity}. In 5.2 proofs have that form $\big([\bar{U}]_1, [h_2]_1, [h_1]_1, [\bar{U}_\alpha]_1, [h_{2,\alpha}]_1, v_1', v_2', v_3', \pi_1', \pi_2', \pi_3', \pi_4', \pi_5'\big)$. Here we can send the same verifier challenge α in both 5.3 and 5.4 (assuming we run the protocols in parallel) which allows us to avoid sending v_1', π_1' in 5.4. Further, this allows us to batch the proofs (π_2', π_3') with the proof for (π_1, π_3) because these all use the same α. Thus π_{unity} contributes 7$\mathbb{G}_1$, and 2$\mathbb{F}$ Thus we have a total of 14$\mathbb{G}_1$, 1$\mathbb{G}_2$ and 4$\mathbb{F}$.

For the verifier, their pairing check in Fig. 5.3 uses pairings of the form $e(*, [1]_2)$ and $e([z_I]_1)$. We also have 3 KZG verifiers which use pairings of the form $e(*, [1]_2)$, $e(*, [\tau]_2)$. This amounts to 2 batched pairings. In Fig. 5.2 we have a 5 KZG verifiers. Two use a degree check and thus use pairings of the form $e(*, [1]_2)$, $e(*, [\tau]_2)$, and $e(*, [\tau^{d-n+1}]_2)$. The others have the usual pairings as these do not have degree checks. Thus we can batch the pairing checks to get a total of 4 unique pairings over the two constructions. In the protocol of Fig. 5.3, the work of the prover is dominated by the computation of $Q(X)$ and $p_2(X)$ which have degree m^2, because $[Q_1]$ is formed in time m by using the pre-computed individual proofs, and all the other proof elements are commitments to polynomials of degree m. In the protocol

of Fig. 5.4, prover work is dominated by the computation of $[\bar{U}]_1$ and $[h_2]_1$ that are commitments to polynomials of degree $m\log(N)$. The lookup proof has preprocessing time for C of $N \log N$ $\mathbb{G}_1$, for N the size of the table, and update of proofs can be done in $O(N)$ $\mathbb{G}_2$ operations as described in [TAB+20].

5.6 Implementation

Caulk has been implemented by Dmtry Khovratovich and Mary Maller in Rust using the arkworks library [ac22], and have released the implementation in open source[3]. The code contains a subroutine that computes all KZG openings, which we need for fast proof preprocessing and which can be used in other projects. For all the schemes different from Caulk, we used the Legosnark implementation[4]. All the benchmarks included in this section have been obtained by running the corresponding codes in a laptop with CPU i7-8565U and 8GB of RAM.

In Figure 5.5 we compare Caulk's prover time, in the y axis, with its alternatives in the scenario where $m = 1$ and for different values of N, represented in the x axis on a logarithmic scale. We consider the following schemes:

- Caulk: the $m = 1$ version;
- MT-Pos: SNARKed Merkle Poseidon tree with N elements.
- MT-SHA: SNARKed Merkle SHA-2 tree with N elements.
- Harisa([CFH+21]): RSA-2048 accumulator of N elements.

We see that Caulk is almost 100 times as fast as Merkle trees instantiated with a Poseidon Hash and Groth16 zkSNARK on top, and 10 times as fast as the RSA accumulator. Although the latter stays constant while Caulk's time grows slowly, we claim Caulk will still perform better for all values N that can be cosnider practical.

We compare Caulk's performance for lookup tables in Figure 5.6. The y axis represent prover time, while the x axis represent the value of m. The size of the vector is diferent for every color line. We consider the following schemes:

- MT-Pos-20: SNARKed Merkle tree with Poseidon poseidon hashes and $N = 2^{20}$ elements.
- MT-Pos-8: SNARKed Merkle tree with Poseidon poseidon hashes and $N = 2^8$ elements.

[3] `https://github.com/caulk-crypto/caulk`
[4] `https://github.com/matteocam/libsnark-lego/`

- Caulk-8: Caulk for vectors of size $N = 2^8$.
- Caulk-20: Caulk for vectors of size $N = 2^{20}$.
- Harisa([CFH+21]): RSA-2048 accumulator for vectors of size $N = 2^{16}$ elements. The performance of the prover in RSA accumulators is independent on the size of the vector.

Caulk is faster than Harisa for all the values of N we were able to compute, but approaches as N grows, and will perform worse for bigger tables. Both constructions are significantly faster than Merkle-SNARK.

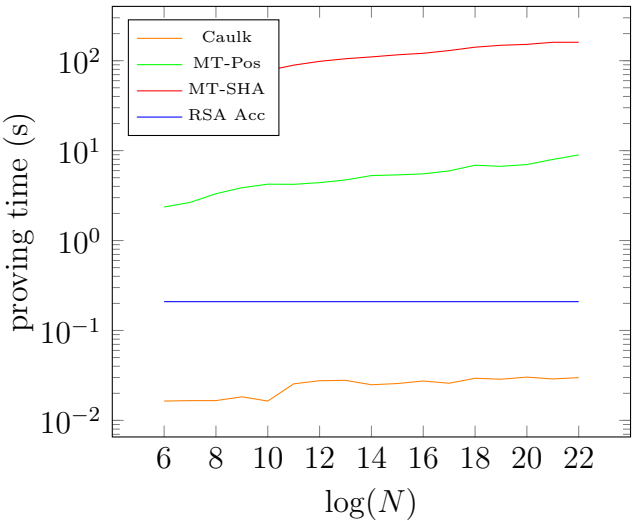

Figure 5.5: Comparison of performance of Caulk and other arguments for zero-knowledge single openings.

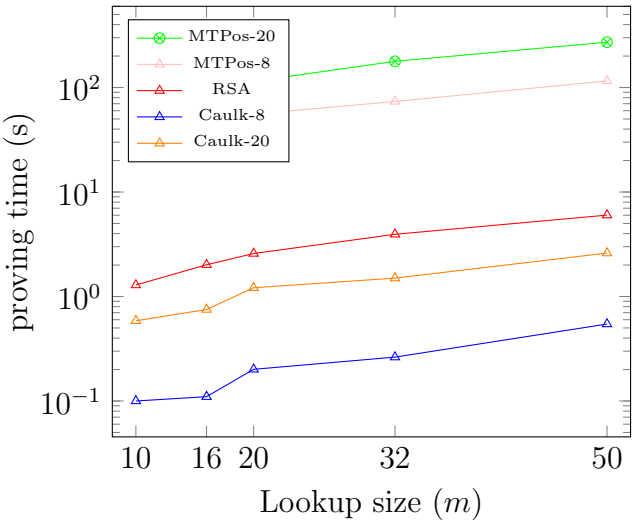

Figure 5.6: Comparison of performance between Caulk and other schemes for lookup tables.

Bibliography

[AAC+17] Hamza Abusalah, Joël Alwen, Bram Cohen, Danylo Khilko, Krzysztof Pietrzak, and Leonid Reyzin. Beyond hellman's time-memory trade-offs with applications to proofs of space. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 357–379. Springer, Heidelberg, December 2017. 73

[ABLZ17] Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, and Michal Zajac. A subversion-resistant SNARK. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2017. 32

[ac22] arkworks contributors. `arkworks` zksnark ecosystem, 2022. 136

[AGR+16] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity. In *ASIACRYPT 2016*, volume 10031 of *LNCS*, pages 191–219, 2016. 104, 107

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2087–2104. ACM Press, October / November 2017. 2, 32

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 2004. 23

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018. 2, 32

[BBF19] Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 561–586. Springer, Heidelberg, August 2019. 16, 72, 73, 74, 77

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018. `https://eprint.iacr.org/2018/046`. 2, 32

[BBHR19] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 701–732. Springer, Heidelberg, August 2019. 2

[BCC+15] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable ring signatures based on DDH. In Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl, editors, *ESORICS 2015, Part I*, volume 9326 of *LNCS*, pages 243–265. Springer, Heidelberg, September 2015. 107

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357. Springer, Heidelberg, May 2016. 2, 32, 33

[BCF+21] Daniel Benarroch, Matteo Campanelli, Dario Fiore, Kobi Gurkan, and Dimitris Kolonelos. Zero-knowledge proofs for set membership: Efficient, succinct, modular. In Nikita Borisov and Claudia Díaz, editors, *FC 2021, Part I*, volume 12674 of *LNCS*, pages 393–414. Springer, 2021. 107, 108

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP, 2014*, pages 459–474. IEEE Computer Society, 2014. 2, 107

[BCG+15] Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure sampling of public parameters for succinct zero knowledge proofs. In *2015 IEEE Symposium on Security and Privacy*, pages 287–304. IEEE Computer Society Press, May 2015. 3

[BCR+19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Heidelberg, May 2019. 30, 32, 33, 36, 38

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

[BCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 276–294. Springer, Heidelberg, August 2014. 2

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988. 2

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 677–706. Springer, Heidelberg, May 2020. 30

[BG12] Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 263–280. Springer, Heidelberg, April 2012. 64, 107

[BG18] Jonathan Bootle and Jens Groth. Efficient batch zero-knowledge arguments for low degree polynomials. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 561–588. Springer, Heidelberg, March 2018. 107

[BGG19] Sean Bowe, Ariel Gabizon, and Matthew D. Green. A multi-party protocol for constructing the public parameters of the pinocchio zk-SNARK. In Aviv Zohar, Ittay Eyal, Vanessa Teague, Jeremy Clark, Andrea Bracciali, Federico Pintore, and Massimiliano Sala, editors, *FC 2018 Workshops*, volume 10958 of *LNCS*, pages 64–77. Springer, Heidelberg, March 2019. 3

[BGM17] Sean Bowe, Ariel Gabizon, and Ian Miers. Scalable multi-party computation for zk-SNARK parameters in the random beacon model. Cryptology ePrint Archive, Report 2017/1050, 2017. `https://eprint.iacr.org/2017/1050`. 3

[BMM+21] Benedikt Bünz, Mary Maller, Pratyush Mishra, Nirvan Tyagi, and Psi Vesely. Proofs for inner pairing products and applications. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 65–97. Springer, Heidelberg, December 2021. 75, 108

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993. 22

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006. 7

[CCs08] Jan Camenisch, Rafik Chaabouni, and abhi shelat. Efficient protocols for set membership and range proofs. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 234–252. Springer, Heidelberg, December 2008. 107

[CF13] Dario Catalano and Dario Fiore. Vector commitments and their applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 55–72. Springer, Heidelberg, February / March 2013. 3, 15, 71, 73

[CFF+21] Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: A toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2021. 19, 20, 29, 31, 34, 38, 45, 46, 49, 62, 66

[CFG+20] Matteo Campanelli, Dario Fiore, Nicola Greco, Dimitris Kolonelos, and Luca Nizzardo. Incrementally aggregatable vector commitments and applications to verifiable decentralized storage. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 3–35. Springer, Heidelberg, December 2020. 72, 73, 74, 75, 76, 77

[CFH+21] Matteo Campanelli, Dario Fiore, Semin Han, Jihye Kim, Dimitris Kolonelos, and Hyunok Oh. Succinct zero-knowledge batch proofs for set accumulators. Cryptology ePrint Archive, Paper 2021/1672, 2021. `https://eprint.iacr.org/2021/1672`. 108, 136, 137

[CHM+20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020. 19, 20, 25, 29, 30, 31, 34, 38, 45, 49, 107, 134

[CL02] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 61–76. Springer, 2002. 107

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002. 3

[CNR+22] Matteo Campanelli, Anca Nitulescu, Carla Ràfols, Alexandros Zacharakis, and Arantxa Zapico. Linear-map vector commitments and their practical applications. Cryptology ePrint Archive, Paper 2022/705, 2022. `https://eprint.iacr.org/2022/705`. 71, 95

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 769–793. Springer, Heidelberg, May 2020. 2

[CPZ18] Alexander Chepurnoy, Charalampos Papamanthou, and Yupeng Zhang. Edrax: A cryptocurrency with stateless transaction validation. Cryptology ePrint Archive, Report 2018/968, 2018. `https://eprint.iacr.org/2018/968`. 72

[Dam93] Ivan Damgård. Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with proprocessing. In Rainer A. Rueppel, editor, *EUROCRYPT'92*, volume 658 of *LNCS*, pages 341–355. Springer, Heidelberg, May 1993. 2

[DFKP15] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 585–605. Springer, Heidelberg, August 2015. 73

[DRZ20] Vanesa Daza, Carla Ràfols, and Alexandros Zacharakis. Updateable inner product argument with logarithmic verifier and applications. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 527–557. Springer, Heidelberg, May 2020. 29, 75

[Fil20] Filecoin. Filecoin powers of tau ceremony attestations, 2020. `https://github.com/arielgabizon/perpetualpowersoftau`. 4

[Fis18] Ben Fisch. PoReps: Proofs of space on useful data. Cryptology ePrint Archive, Report 2018/678, 2018. `https://eprint.iacr.org/2018/678`. 73

[Fis19] Ben Fisch. Tight proofs of space and replication. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 324–348. Springer, Heidelberg, May 2019. 73

[FK] Dankrad Feist and Dmitry Khovratovich. Fast amortized kate proofs. 26, 105

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018. 4, 23, 32, 74

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. SIAM Journal of Computing, 1999. 2

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987. 22

[Fuc18] Georg Fuchsbauer. Subversion-zero-knowledge SNARKs. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 315–347. Springer, Heidelberg, March 2018. 32

[Gab19] Ariel Gabizon. AuroraLight: Improved prover efficiency and SRS size in a sonic-like system. Cryptology ePrint Archive, Report 2019/601, 2019. `https://eprint.iacr.org/2019/601`. 29

[GG17] Essam Ghadafi and Jens Groth. Towards a classification of non-interactive computational assumptions in cyclic groups. *IACR Cryptol. ePrint Arch.*, page 343, 2017. 23, 24, 131

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013. 29, 38, 45

[GK15] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 253–280. Springer, Heidelberg, April 2015. 107

[GKM+18] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, August 2018. 11, 12, 13, 29, 32

[GKR+21] Lorenzo Grassi, Dmitry Khovratovich, Arnab Roy, Christian Rechberger, and Markus Schofnegger. Poseidon: A new hash function for zero-knowledge proof systems. *Usenix Security 2021*, 2021. 104, 107

[GMMM18] Sanjam Garg, Mohammad Mahmoody, Daniel Masny, and Izaak Meckler. On the round complexity of OT extension. In Hovav Shacham and

Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 545–574. Springer, Heidelberg, August 2018. 3

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proofs. In *SIAM Journal on Computing*, pages 186–208, 1989. 2

[Gro09] Jens Groth. Linear algebra with sub-linear zero-knowledge arguments. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 192–208. Springer, Heidelberg, August 2009. 33

[Gro10] Jens Groth. Short non-interactive zero-knowledge proofs. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 341–358. Springer, Heidelberg, December 2010. 2

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016. 2, 104

[GRWZ20] Sergey Gorbunov, Leonid Reyzin, Hoeteck Wee, and Zhenfei Zhang. Pointproofs: Aggregating proofs for multiple vector commitments. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 2007–2023. ACM Press, November 2020. 72, 74, 76

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011. 4

[GW20] Ariel Gabizon and Zachary J. Williamson. plookup: A simplified polynomial protocol for lookup tables. *IACR Cryptol. ePrint Arch.*, page 315, 2020. 104, 106

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. `https://eprint.iacr.org/2019/953`. 19, 20, 25, 29, 31, 66, 134

[Ish20] Yuval Ishai. Zero-knowledge proofs from information theoretic proof systems. In *Zkproofs Blog, https://zkproof.org/2020/08/12/information-theoretic-proof-systems/*, 2020. 29

[JR13] Charanjit S. Jutla and Arnab Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2013. 33

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992. 1

[KMSV21] Markulf Kohlweiss, Mary Maller, Janno Siim, and Mikhail Volkhov. Snarky ceremonies. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 98–127. Springer, Heidelberg, December 2021. 3

[KP98] Joe Kilian and Erez Petrank. An efficient noninteractive zero-knowledge proof system for np with general assumptions. Journal of Cryptology, 1998. 2

[KPV19] Assimakis Kattis, Konstantin Panarin, and Alexander Vlasov. RedShift: Transparent SNARKs from list polynomial commitment IOPs. Cryptology ePrint Archive, Report 2019/1400, 2019. `https://eprint.iacr.org/2019/1400`. 30

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 177–194. Springer, 2010. 3, 13, 24, 25, 30, 32, 114, 118, 124, 130

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Heidelberg, March 2012. 2

[Lip13] Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 41–60. Springer, Heidelberg, December 2013. 2

[LM19] Russell W. F. Lai and Giulio Malavolta. Subvector commitments with application to succinct arguments. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 530–560. Springer, Heidelberg, August 2019. 16, 17, 72, 74, 75

[LRY16] Benoît Libert, Somindu C. Ramanna, and Moti Yung. Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *ICALP 2016*, volume 55 of *LIPIcs*, pages 30:1–30:14. Schloss Dagstuhl, July 2016. 72, 74

[LSZ22] Helger Lipmaa, Janno Siim, and Michal Zajac. Counting vampires: From univariate sumcheck to updatable zk-snark. Cryptology ePrint Archive, Paper 2022/406, 2022. `https://eprint.iacr.org/2022/406`. 31, 32

[LY10] Benoît Libert and Moti Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 499–517. Springer, Heidelberg, February 2010. 3, 71, 74

[Mau09] Ueli M. Maurer. Unifying zero-knowledge proofs of knowledge. In *AFRICACRYPT*, volume 5580 of *Lecture Notes in Computer Science*, pages 272–286. Springer, 2009. 28

[MBKM19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019. xv, 29, 30, 31, 60

[Mer88] Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 369–378. Springer, Heidelberg, August 1988. 72

[MGGR13] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 397–411. IEEE Computer Society, 2013. 107

[Mic00] Silvio Micali. The knowledge complexity of interactive proofs. In *SIAM Journal on Computing 30 (4)*, pages 1253–1298, 2000. 1

[MRV16] Paz Morillo, Carla Ràfols, and Jorge Luis Villar. The kernel matrix Diffie-Hellman assumption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 729–758. Springer, Heidelberg, December 2016. 41

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008. 2

[Nao03] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109. Springer, Heidelberg, August 2003. 4

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992. 3

[PFM+22] Luke Pearson, Joshua Fitzgerald, Héctor Masip, Marta Bellés-Muñoz, and Jose Luis Muñoz-Tapia. Plonkup: Reconciling plonk with plookup. *IACR Cryptol. ePrint Arch.*, page 86, 2022. 106

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013. 2

[PST13] Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. Signatures of correct computation. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 222–242. Springer, Heidelberg, March 2013. 30, 49, 75

[RD16] Ling Ren and Srinivas Devadas. Proof of space from stacked expanders. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part I*, volume 9985 of *LNCS*, pages 262–285. Springer, Heidelberg, October / November 2016. 73

[RZ21] Carla Ràfols and Arantxa Zapico. An algebraic framework for universal and updatable SNARKs. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 774–804, Virtual Event, August 2021. Springer, Heidelberg. 29, 32

[Sch80] Jacob T Schwartz. Fast probabilistic algorithms for verification of polynomial identities. In *Journal of the ACM 24.7*, pages 701–717. ACM, 1980. 8

[SCP+22] Shravan Srinivasan, Alexander Chepurnoy, Charalampos Papamanthou, Alin Tomescu, and Yupeng Zhang. Hyperproofs: Aggregating and maintaining proofs in vector commitments. In *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA, August 2022. USENIX Association. 75

[Set20] Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 704–737. Springer, Heidelberg, August 2020. 29, 32

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997. 23

[SZ20] Alan Szepieniec and Yuncong Zhang. Polynomial iops for linear algebra relations. Cryptology ePrint Archive, Report 2020/1022, 2020. `https://eprint.iacr.org/2020/1022`. 29, 31, 45

[TAB+20] Alin Tomescu, Ittai Abraham, Vitalik Buterin, Justin Drake, Dankrad Feist, and Dmitry Khovratovich. Aggregatable subvector commitments for stateless cryptocurrencies. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20*, volume 12238 of *LNCS*, pages 45–64. Springer, Heidelberg, September 2020. 26, 74, 75, 76, 88, 93, 94, 105, 136

[TCZ+20] Alin Tomescu, Robert Chen, Yiming Zheng, Ittai Abraham, Benny Pinkas, Guy Golan-Gueta, and Srinivas Devadas. Towards scalable threshold cryptosystems. In *2020 IEEE Symposium on Security and Privacy*, pages 877–893. IEEE Computer Society Press, May 2020. 75, 94

[Tor21] Tornado cash privacy solution version 1.4, 2021. `https://tornado.cash/Tornado.cash_whitepaper_v1.4.pdf`. 107

[WTs+18] Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *2018 IEEE Symposium on Security and Privacy*, pages 926–943. IEEE Computer Society Press, May 2018. 2, 32

[XZZ+19] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 733–764. Springer, Heidelberg, August 2019. 2

[ZBK+22] Arantxa Zapico, Vitalik Buterin, Dmitry Khovratovich, Mary Maller, Anca Nitulescu, and Mark Simkin. Caulk: Lookup arguments in sublinear time. Cryptology ePrint Archive, Paper 2022/621, 2022. `https://eprint.iacr.org/2022/621`. 103

[ZCary] ZCash protocol specification, 2022, 1st February. `https://github.com/zcash/zips/blob/master/protocol/protocol.pdf`. 4, 104, 106

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *International symposium on symbolic and algebraic manipulation.*, pages 216–226. Springer, 1979. 8

[ZkS21] Zksync rollup protocol, 2021. `https://github.com/matter-labs/zksync/blob/master/docs/protocol.md`. 107

Appendix A

Publications

Conference proceedings

2021 Ràfols, C., & Zapico, A. An Algebraic Framework for Universal and Updatable SNARKs. *Annual International Cryptology Conference*.

Abstract. We introduce Checkable Subspace Sampling Arguments, a new information theoretic interactive proof system in which the prover shows that a vector has been sampled in a subspace according to the verifier's coins. We show that this primitive provides a unifying view that explains the technical core of most of the constructions of universal and updatable pairing-based (zk)SNARKs. This characterization is extended to a fully algebraic framework for designing such SNARKs in a modular way. We propose new constructions of CSS arguments that lead to SNARKs with different performance tradeoffs. Our most efficient construction, Basilisk, seems to have the smallest proof size in the literature, although it pays a price in terms of structure reference string for the number of multiplicative gates whose fan-out exceeds a certain bound.

2022 Daza, V., Haque, A., Scafuro, A., Zacharakis, A. & Zapico, A. Mutual Accountability Layer: Accountable Anonymity within Accountable Trust. *International Symposium on Cyber Security Cryptology and Machine Learning.*

Abstract. Anonymous cryptographic primitives reduce the traces left by the users when interacting over a digital platform. However, they also prevent a platform owner to hold users accountable in case of malicious behaviour. Revocable anonymity offers a compromise by allowing only the manager (and not the other users) of the digital platform to de-anonymize user's activities when necessary. However, such de-anonymization power can be abused too, as a misbehaving manager can de-anonymize all the activities without user's awareness. Previous work propose to mitigate this issue by distributing the de-anonymization power across several entities. However, there is no comprehensive and formal treatment where both accountability and non-frameability (i.e., the inability to falsely accuse a party of misbehavior) for both the user and the manager are explicitly defined and provably achieved.

In this paper we formally define mutual accountability: a user can be held accountable for her otherwise anonymous digital actions and a manager is held accountable for every de-anonymization attempt; plus, no honest party can be framed – regardless of what malicious parties do.

Instead of distributing the de-anonymization power across entities, instead, we decouple the power of de-anonymization from the power of monitoring de-anonymization attempts. This allows for greater flexibility, particularly in the choice of the monitoring entities.

We show that our framework can be instantiated generically from threshold encryption schemes and succinct non-interactive zero-knowledge. We also show that the highly-efficient threshold group signature scheme by Camenisch et al.(SCN'20) can be modified and extended to instantiate our framework.

Preprints

2022 Campanelli, M., Nitulescu, A., Ràfols, C., Zacharakis, A., & Zapico, A. Linear-map Vector Commitments and their Practical Applications.

Abstract. Vector commitments (VC) are a cryptographic primitive that allow one to commit to a vector and then open some of its positions efficiently. Vector commitments are increasingly recognized as a central tool to scale highly decentralized networks of large size and whose content is dynamic. In this work, we examine the demands on the properties that an ideal vector commitment should satisfy in the light of the emerging plethora of practical applications and propose new constructions that improve the state-of-the-art in several dimensions and offer new tradeoffs. We also propose a unifying framework that captures several constructions and show how to generically achieve some properties from more basic ones. On the practical side, we focus on building efficient schemes that do not require new trusted setup (we can reuse existing ceremonies for pairing-based powers of tau run by real-world systems such as ZCash or Filecoin). Our implementation demonstrates that our work over-performs in efficiency prior schemes with same properties.

2022 Zapico, A., Buterin, V., Khovratovich, D., Maller, M., Nitulescu, A., & Simkin, M.. Caulk: Lookup Arguments in Sublinear Time.

Abstract. We present position-hiding linkability for vector commitment schemes: one can prove in zero knowledge that one or m values that comprise commitment cm all belong to the vector of size N committed to in C. Our construction Caulk can be used for membership proofs and lookup arguments and outperforms all existing alternatives in prover time by orders of magnitude.

For both single- and multi-membership proofs Caulk beats SNARKed Merkle proofs by the factor of 100 even if the latter instantiated with Poseidon hash. Asymptotically our prover needs $O(m^2 + m \log N)$ time to prove a batch of m openings, whereas proof size is $O(1)$ and verifier time is $O(\log(\log N))$.

As a lookup argument, Caulk is the first scheme with prover time sublinear in the table size, assuming $O(N \log N)$ preprocessing time and $O(N)$ storage. It can be used as a subprimitive in verifiable computation schemes in order to drastically decrease the lookup overhead.

Our scheme comes with a reference implementation and benchmarks.