A Survey of Zero-Knowledge Succinct Non-Interactive Arguments with preprocessing

Nikitas Paslis
AL1.20.0010

Examination committee:
Aris Pagourtzis, School of Electrical and Computer Engineering, National Technical University of Athens
Stathis Zachos, School of Electrical and Computer Engineering, National Technical University of Athens
Petros Potikas, School of Electrical and Computer Engineering, National Technical University of Athens

Supervisor:
Aris Pagourtzis, Professor,
School of Electrical and Computer Engineering
National Technical University of Athens

Μεταπτυχιακό Πρόγραμμα «Αλγόριθμοι, Λογική και Διακριτά Μαθηματικά» – 2016

ABSTRACT

In this thesis we study the notion of *zero-knowledge succinct non-interactive arguments* (zkSNARGs) with *preprocessing*. Given an arithmetic circuit C, public input x, private input w and public output y, a zkSNARG constitutes a protocol that allows a *computationally bounded* party, called *prover*, to convince another, called *verifier*, that $C(x, w) = y$, i.e. the circuit with inputs x, w evaluates to y, without the verifier gaining any more information beyond the validity of the computation. Moreover, the prover is not required to interact with anyone in order to generate the proof that the verifier will check, while the produced proof is also *succinct*, i.e. short to store and quick to verify. Finally, these protocols are *universal*, in the sense that they can be used for any circuit up to a pre-determined bound on their size, while also being *updatable*, meaning that their SRS can be updated in a verifiable manner by any user.

We present the current landscape of the theory of preprocessing universal and updatable zkSNARKs, which advocates for constructing them in a modular way, separating the information theoretic component of the construction from the cryptographic primitive that is used for the compiling phase. The information theoretic part is a *polynomial holographic proof* (PHP) [Cam+20] for the NP-complete language of Rank 1 Constraint Systems (R1CS), while the cryptographic primitive is a polynomial commitment scheme [KZG10].

Furthermore, we examine the bottleneck of the recent constructions, called *Checkable Subspace Sampling* (CSS) [RZ21], separately and give constructions for it that achieve different efficiency trade-offs. Finally, we present Basilisk [RZ21], which achieves the smallest proof size in the literature.

ΣΥΝΟΨΗ

Σε αυτή τη διπλωματική εργασία μελετάμε την έννοια των *μηδενικής-γνώσεις συνοπτικών μη-διαλογικών επιχειρημάτων* (zkSNARGs) με προεπεξεργασία. Δοθέντος ενός αριθμητικού κυκλώματος C, δημόσιας εισόδου x, ιδιωτικής εισόδου w και δημόσιας εξόδου y, ένα zkSNARG αποτελεί ένα πρωτόκολλο το οποίο επιτρέπει σε ένα *υπολογιστικά φραγμένο* συμμετέχοντα, ονόματι *αποδεικνύοντας*, να πείσει έναν άλλο, ονόματι *επαληθευτής*, ότι $C(x, w) = y$, δηλαδή ότι το κύκλωμα με εισόδους x, w αποτιμάται σε y, χωρίς ο επαληθευτής να αποκομμίσει οποιαδήποτε πληροφορία πέρα από την εγκυρότητα του υπολογισμού. Επιπλέον, ο αποδεικνύοντας δεν απαιτήται να αλληλεπιδράσει με οποιονδήποτε για τη δημιουργία της απόδειξης που ο επαληθευτής θα ελέγξει, ενώ η παραγώμενη απόδειξη είναι συνοπτική, δηλαδή μικρής χωρητικότητας και γρήγορα επαληθεύσιμη. Τέλος, τα πρωτόκολλα αυτά είναι *καθολικά*, υπό την έννοια ότι δουλεύουν για οποιοδήποτε κύκλωμα εώς ένα προκαθορισμένο όριο στο μέγεθος του, ενώ είναι επίσης *επικαιροποιήσιμα*, εννοώντας ότι τα SRS τους μπορούν να επικαιροποιηθούν μέσω ενός επαληθεύσιμου τρόπου από οποιονδήποτε χρήστη.

Παρουσιάζουμε το τρέχον τοπίο της θεωρίας των καθολικών και επικαιροποιήσιμων zkSNARGs με προεπεξεργασία, το οποίο συνηγορεί για την κατασκευή τους με αρθρωτό τρόπο, διαχωρίζοντας το πληροφοριοθεωρητικό σκέλος της κατασκευής από το κρυπτογραφικό θεμελιακό στοιχείο που χρησιμοποιείται κατά τη σύνταξη. Το πληροφοριοθεωρητικό κομμάτι είναι μια *πολυωνυμική ολογραφική απόδειξη* [Cam+20] για την NP-πλήρη γλώσσα των Βαθμού 1 Συστημάτων Περιορισμών, ενώ το κρυπτογραφικό θεμελιακό στοιχείο είναι ένα σχήμα πολυωνυμικών δεσμεύσεων [KZG10].

Επιπρόσθετα, εξετάζουμε το υπολογιστικό κώλυμα των πρόσφατων κατασκευών, ονόματι *ελέγξιμη δειγματοληψία υποχώρου* [RZ21], ξεχωριστά και δίνουμε κατασκευές αυτού που επιτυγχάνουν διαφορετικούς συμβιβασμούς απόδοσης. Τέλος, παρουσιάζουμε το Basilisk [RZ21], το οποίο επιτυγχάνει το μικρότερο μέγεθος απόδειξης στη βιβλιογραφία.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Interactive Proof systems and Zero-Knowledge

"The Knowledge of London", or simple "the Knowledge", is an examination system designed for taxi drivers in London, United Kingdom. Every aspiring taxi driver must undergo training for an extensive period of time, lasting between 2 to 4 years, in order to memorise all the landmarks, streets and places in a 6 mile radius of the Chairing Cross. As part of the examination, the candidate is requested to situate two arbitrary points and find the best route between them without the aid of a map or any other technology. This examination is repeated over the years to assure that the aspiring driver has the necessary Knowledge for the position.

Alice believes she can beat the Knowledge and can find better route than Bob, the Knowledge examiner. Alice, trying to protect the fruits of her hard work, does not want to reveal to Bob what the actual route is, but rather she *merely* wants to convince him that she *knows* one. For this part, Bob does not believe her until she can prove it to him.

Whether Alice's desire can be fulfilled gives birth to the following question: *Is it possible to create a procedure, that can convince someone about the validity of a statement, without him gaining any knowledge beyond the statement's validity?* While considering this question in the mid 80's, Goldwasser, Micali and Rackoff observed that it regards settings where there is some short of *interaction* between two parties. This observation led them to defining, in their landmark paper [GMR89], the complexity class of *Interactive Proofs* (IP).

Informally, an IP is a protocol that allows one party, the *prover*, to convince another, the *verifier* about the validity of a statement. There are two main properties an IP must satisfy:

- *Completeness* states that if the prover and verifier follow the protocol, the verifier accepts the validity of a true statement.

- *Soundness* guarantees that the prover cannot deceive the verifier into accepting the validity of a false statement, even if the prover deviates arbitrarily from the instructions of the protocol.

We can strengthen the soundness property, by considering *knowledge soundness* (IP of knowledge), which guarantees that whenever the prover convinces the verifier, then he *knows* a *witness* that attests to the validity of the statement.

Given an interactive proof, the answer to the previous question corresponds to whether or not the protocol satisfies the following property:

- *Zero knowledge* prevents the verifier to learn anything from an execution of the protocol, apart from the validity of the statement.

Thus, a zero-knowledge interactive proof of knowledge is a cryptographic protocol that allows Alice to convince Bob she has the necessary Knowledge, without disclosing any additional information about it. This is a powerful cryptographic primitive that can be used in many security applications, whenever it is desired to strike a balance between verifiability of information and secrecy. Examples of these applications include manipulation of healthcare data, cloud computing, and public-ledger technologies, e.g. blockchains.

1.2 Complexity classes that admit Zero-Knowledge IPs

Much of the early work on IPs and zero-knowledge was devoted in identifying the classes of languages that admit these kinds of protocols, with what properties and under which cryptographic assumptions. Goldreich, Micali and Wigderson [GMW91] constructed a computational zero-knowledge (bounded verifier) proof for $\mathcal{NP}$, by constructing one for the language of 3-COLORABILITY, based on the existence of one-way functions. Their results were generalized in [IY87], where under the same assumptions and verifier bounds they construct zero-knowledge proofs for the whole IP. Pairing this with the celebrated result by A.Shamir [Sha92], that IP=PSPACE, we get computational zero-knowledge proofs for PSPACE.

Over the last three decades, different constructions have emerged, many of which offer zero-knowledge even against unbounded verifier (*perfect* zero-knowledge). However, unless the Polynomial Hierarchy collapses, NP-complete languages cannot have IP wchich are both sound against unbounded prover and zero-knowledge unbounded verifier [For99]. Therefore, when designing a proof system for NP languages, one must choose between perfect soundness and computational soundness. Proof systems that are sound against computationally bounded provers are called *arguments*.

1.3 Efficiency metrics

Zero-knowledge proofs are ubiquitous in cryptography. For instance they are used in constructions of digital signatures, public-key encryption schemes, voting and auctioning systems, e-cash, secure multiparty computation (MPC), and verifiable outsourced computation. Unfortunately, they are an expensive component of all of these, and it is therefore important for them to be as efficient as possible. Aside from the flavours of their different properties, zero-knowledge proofs can be compared with respect to their efficiency, with the main metrics used to measure their performance being interaction, communication and computational complexity.

The interaction of a proof system measures the number of messages that the prover and verifier exchange. Different constructions require different amounts of interaction

and work on the subject has shown lower bounds on the amount of interaction needed to construct proof systems with different properties for various classes of languages. Nevertheless, these lower bounds can be circumvented by assuming the existence of an honestly generated *common reference string* (CRS), which is shared between prover and verifier. Another way of minimising interaction between prover and verifier is to apply a general transformation to an interactive proof system and turn it into a non-interactive one. This methodology was first illustrated by Fiat and Shamir [FS87], who showed how any public-coin proof system (i.e. the verifier's messages are only random coins) can be transformed into a non-interactive one, where the prover creates the proof of the statement to be verified, without any interaction.

The communication complexity of a proof system corresponds to the overall size of the messages exchanged between prover and verifier and it is usually measured with respect to either the size of the instance, e.g. circuit size, or the size of the witness. Again different constructions achieve different communication complexity, although results from [GK96] show that sublinear communication in unlikely to be achievable for proof systems with statistical soundness.

Fortunately, things are different for arguments, as Kilian [Kil92] constructed the first zero-knowledge argument system with constant round of interaction and poly-logarithmic communication cost, by getting the prover to construct a probabilisticaly checkable proof (PCP) and hash it using Merkle tree to produce short proof for the verifier. But this approach has not been fruitful, as PCPs are expensive to compute. In the non-interactive setting, Micali [Mic94] constructed the first argument with sublinear size using the Fiat-Shamir transform. Both of these novel constructions leverage the computational bound of the prover in order to construct proof systems that are *succinct*

Kilian's work inspired the creation of interactive oracle proofs (IOP) [BCS16], which constitutes a generalization of IP and PCP. In an IOP the verifier does not read the messages outputted by the prover in their entirety, but rather has oracle access to them and can make probabilistic queries to them, leading to reduction in the communication cost of the argument.

Having low verifier complexity can be important even if we are not interested in zero-knowledge and even for languages in P, as is the case of verifiable computation. Verifiable computation has taken two distinct flavours in the literature: in the first the prover wishes to prove that he has correctly computed a public function on private inputs, e.g. private database searches; in the second the verifier wishes to offload a large computation on public inputs to a more powerful prover.

1.4 ZK-SNARKs with pre-processing

In the last decade, mainly due to the increasing demand in public-ledger technologies which require minimizing storage of data and time to verify proofs, there has been a growing interest in zero-knowledge proof systems that additionally are *succinct* and *non-interactive*, the so-called zkSNARKs. These are computationally-sound proof systems that are *succinct*, in that their proofs are short and efficient to verify, i.e. the proof size and verification time should be constant or polylogarithmic in the length of the non-deterministic witness.

In circuit-based arguments for general computations the verifier must at least read the statement to be proven which includes both the description of the computation (i.e. the circuit) and its input (i.e. public input). But this already is not succinct, because

by reading the whole circuit, the verifier runs linearly in the size of the computation. One way to solve this problem is create *preprocessing zkSNARKs*, which leverage the notion of *holography*. Here the verifier (or more generally the *relation encoder* or *indexer*) generates an encoding of the circuit C. He does that *once and for all*. This encoding then can be used at any time to verify an unbounded number of proofs for the computation C. This is indeed a succinct system: while the encoding creation does depend on $|C|$, verification does not.

In contexts with many verifiers, e.g. blockchains, the SRS generation requires a trusted setup. Solutions that minimize this trust (e.g. MPC secure against dishonest majority) are often expensive and impractical to be carried out for every single computation. To mitigate this problem Groth et. al [Gro+18] introduced the model of *universal and updatable SRS*. An SRS is universal if it can be used to generate and verify proofs for all circuits up to some bound (e.g. number of multiplication gates). An SRS is updatable if any user can add randomness to it and a sequence of updates makes it secure if at least one user acted honestly. Groth et al. [Gro+18] proposed the first such proof system, although the SRS size is quadratic to the maximum number of supposed multiplication gates, as well as having quadratic verification and update time. In contrast, recent works have achieved SRS that is linear in the largest supported circuit, first of its kind being Sonic [Mal+19]. Moreover, PLONK [GWC19], Marlin [Chi+19], LUNAR [Cam+20] and Basilisk [RZ21] achieve proving time concretely faster than that of Sonic while also retaining constant-size proofs.

The current landscape of preprocessing zkSNARKs with universal SRS - modular paradigm. There is an important trend in cryptography, that advocates for constructing protocolos 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, while also gives a straightforward way to compare the efficiency and ideas of different protocols. Thus, breaking the constructions in its simplest building blocks is essential to keep track of advancement in this evergrowing area.

Towards this goal, building efficient cryptoraphic arguments works in two distinct steps. First construct an information-theoretic protocol in an abstract model, e.g. interactive proofs, Probabilistically Checkable Proofs (PCPs), Interactive Oracle Proofs (IOPs). In the second step, apply a cryptographic compiler that, taking an abstract protocol as input, transforms in into an efficient computationally sound argument via a cryptographic primitive. This approach has been adopted explicitly or implicitly in most recent works on UaU zkSNARKs. There the information theoretic part is an algebraically-flavored variant of IOPs (e.g. Algebraic Holographic IOPs in [Chi+19], Polynomial Holographic IOPs in [Cam+20], [RZ21]), while the cryptographic primitive are *polynomial commitments* [KZG10]. Using different types of polynomial commitments in the compiling phase will lead to zkSNARKs with different trade-offs.

The overview of these constructions can be seen as follows:

- Run the setup algorithm to produce the SRS.
- Starting with a general computation over a finite field, create an arithmetic argument for verifying the validity of the computation.
- Preprocess the matrices of the constraints, encoding them as polynomials and give oracle access to them to the verifier.

- Transform the constraint system to a PHP, i.e. move from relations between matrices and vectors to equations between prover and encoder polynomials.

- The prover commits to the polynomials and gives oracle access to them to the verifier.

- The verifier validates (with high probability) the correctness of the computation by querying both sets of polynomials at random points and checking that the equations hold for these evaluations.

Of course, this will lead to a succinct *interactive* argument of knowledge. To make it non interactive, since the verifier is public-coin, one can use the Fiat-Shamir transform [FS87].

CHAPTER 2

PRELIMINARIES

We denote by $[n]$ the set $\{1, \ldots, n\} \subseteq \mathbb{N}$. We use boldface for vectors and matrices. For a matrix $\mathbf{M}$, $|\mathbf{M}|$ is the number of non-zero entries. For a set S, $|S|$ is its cardinality and denote $x \leftarrow S$ a sampling of a random $x \in S$. We use $\mathbb{F}$ for fields, $\mathbb{G}$ for groups, and $\mathbb{F}[X]$ for the ring of univariate polynomials with coefficients from $\mathbb{F}$, and $\mathbb{F}[X_1, X_2, \ldots, X_n]$ for polynomials in n variables. Given a finite set S, we denote $\mathbb{F}^S$ the set of vectors indexed by elements in S. For $n \in \mathbb{N}$ we write $\mathbb{F}^n$ instead of $\mathbb{F}^{[n]}$. We denote by $\lambda \in \mathbb{N}$ a security parameter.
For vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}^n$, we write $\mathbf{a} \cdot \mathbf{b}$ for their inner product. We also write $\mathbf{a} \circ \mathbf{b}$ for their Hadamard product, i.e. $(a_1 b_1, a_2 b_2, \ldots, a_n b_n)$.

2.1 Schwartz-Zippel lemma

Lemma 2.1. Let $P \in \mathbb{F}[X_1, \ldots, X_n]$ be a non-zero polynomial of total degree $d \geq 0$ over a field $\mathbb{F}$. Let S be a finite subset of $\mathbb{F}$ and let $r_1, \ldots, r_n$ be uniformly random elements of S. Then

$$\Pr[P(r_1, \ldots, r_n) = 0] \leq \frac{d}{|S|}$$

Corollary 2.2. Let $P \in \mathbb{F}[X_1, \ldots, X_n]$ be a non-zero polynomial over a field $\mathbb{F}$ and d the maximum degree of X_1. Let S be a finite subset of $\mathbb{F}$ and let r_1 be a uniformly random element of S. Then

$$\Pr[P(r_1, X_2 \ldots, X_n) = 0] \leq \frac{d}{|S|}$$

Corollary 2.3. Let $P_1, P_2 \in \mathbb{F}[X]$ be two non-zero polynomials over $\mathbb{F}$ of degrees d_1, d_2 respectively. Then for a uniformly random element $r \in \mathbb{F}$, it holds that:

$$P_1(r) = P_2(r) \Rightarrow \Pr[P_1(X) = P_2(X)] \geq 1 - \frac{max(d_1, d_2)}{|\mathbb{F}|}$$

2.2 Bilinear groups

Throughout the constructions we use a *bilinear group sampler*, which is a probabilistic polynomial-time algorithm SampleGrp, that on input a security parameter λ (represented in unary), outputs a tuple $\langle group \rangle = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, G, H, e)$ where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of prime order $q \in \mathbb{N}$, G, H are generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate, efficiently computable bilinear map. We will use additive notation for the group operations. For the map, it holds that $e(G_1, G_2)$ is a generator of G_T and also that $e(\gamma G_1, \beta G_2) = e(G_1, G_2)^{\gamma\beta}$. We will also use the following notation for group elements:

$$[\gamma]_\alpha \coloneqq \gamma G_\alpha \text{ for } \alpha \in \{1, 2, T\}.$$

With this notation $G_1 = [1]_1, G_2 = [1]_2$ and $e([\gamma]_1, [\beta]_2) = e([1]_1, [1]_2)^{\gamma,\delta} = [\gamma\beta]_T$

2.3 Lagrange Polynomial basis

Let $\mathbb{F}$ be a finite field and $\mathbb{H} \subset \mathbb{F}$ and suppose $\mathbb{H} = \{h_i\}_{i=1}^m$, for some canonical order. Consider the vanishing polynomial in $\mathbb{H}, t(X) \in \mathbb{F}_m[X]$ which is such that $t(h_i) = 0, \quad \forall h_i \in \mathbb{H}$. Clearly $t(X) = \prod_{i=1}^m (X - h_i)$ is of mini. Then, the quotient ring $\mathbb{F}[X]/(t(X))$ is an m-dimensional vector space over $\mathbb{F}$. Consider now the following m polynomials of degree m-1:

$$\lambda_i = \prod_{i \neq j} \frac{X - h_j}{h_i - h_j}$$

Denote $\lambda(\mathbf{X}) = (\lambda_1(X), \ldots, \lambda_m(X))$. For all $i \in [m]$ it holds that

$$\lambda_i(h_j) = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{else} \end{cases}$$

Claim: The λ_i's are linearly independent over $\mathbb{F}$. This is easy to see, as taking an $\mathbb{F}$-linear combination of these polynomials with coefficients $\{a_i\}_{i=1}^m \in \mathbb{F}^m$ and evaluating at h_i, we get $a_i = 0$.
Thus $\{\lambda_i(X)\}_{i=1}^m$ is a basis for the $\mathbb{F}$-vector space of polynomials with degree $< m$.
Now considering these polynomials in the ring $\mathbb{F}[X]/(t(X))$, we get the following properties:

$$(1) \forall i \neq j \in [m] : \lambda_i(X) \cdot \lambda_j(X) \equiv 0 (mod(t(X)))$$
$$(2) \forall i \in [m] : \lambda_i^2(X) \equiv \lambda_i(X) (mod(t(x)))$$

Thus, for

$$a(X) = \mathbf{a}^\intercal \cdot \lambda(X) = \sum_{i=1}^m a_i \cdot \lambda_i(X)$$
$$b(X) = \mathbf{b}^\intercal \cdot \lambda(X) = \sum_{i=1}^m b_i \cdot \lambda_i(X)$$

we have

$$a(X) \cdot b(X) = (\sum_{i=1}^m a_i \cdot \lambda_i(X)) \cdot (\sum_{i=1}^m b_i \cdot \lambda_i(X)) \equiv \sum_{i=1}^m a_i \cdot b_i \cdot \lambda_i(X) \equiv \sum_{i=1}^m (\mathbf{a} \circ \mathbf{b})^\intercal \cdot \lambda(X) (mod(t(X)))$$

Assume now that $\mathbb{H}$ is a multiplicative subgroup of $\mathbb{F}$. Since the cardinality of $\mathbb{H}$ is m, we have that $\forall i \in [m] : h_i^m = 1$ and that these are exactly the $x \in \mathbb{F}$ such that $x^m = 1$ Thus, the vanishing polynomial at $\mathbb{H}$ has the form

$$t(X) = X^m - 1$$

Also, using the Vieta formulae we have the following get the following regarding the i-th Lagrange polynomial

$$\lambda_i(X) = \frac{h_i}{m} \frac{(X^m - 1)}{(X - h_i)}$$
$$\lambda_i(0) = \frac{1}{m}$$

Thus, in the case where $\mathbb{H}$ is a multiplicative subgroup of $\mathbb{F}$, both the vanishing polynomial at $\mathbb{H}$ as well as the Lagrange basis polynomials have a compact representation which can be computed in $\mathcal{O}(logm)$ field operations. This is a critical point, as these can be computed by the verifier without sacrificing the succinctness of the proof system.

2.4 Assumptions

The construction of the polynomial commitment scheme (and subsequently of the zk-SNARK) will achieve its security in the Algebraic Group Model (AGM) [FKL17]. This model replaces specific knowledge assumptions (such us Power Knowledge of Exponent Assumptions). In AGM all algorithms are modeled as *algebraic*, meaning that whenever the algorithm outputs a group element G, it must also provide a description of it based on the group elements it already has. Formally:

Definition 2.4 ([FKL17])**.** (algebraic algorithm). Let $\mathbb{G}$ be a cyclic group of prime order q and $\mathcal{A}_{alg}$ a probabilistic algorithm run on initial inputs including description of $\mathbb{G}$. During its execution $\mathcal{A}_{alg}$ may interact with oracles or other parties and receive further inputs including obliviously sampled group elements (which it cannot sample itself). Let $\mathbf{L} \in \mathbb{G}^n$ be the list of all group elements $\mathcal{A}_{alg}$ has already received such that all other inputs it has been given do not depend in any way on group elements. We call $\mathcal{A}_{alg}$ algebraic if whenever it outputs a group element $G \in \mathbb{G}$ it also outputs a vector $\mathbf{a} = [a_i]_{i=1}^n \in \mathbb{F}_q^n$ such that $G = \sum_{i=1}^n a_i L_i$. The coefficients **a** are called the 'representation' of G with respect to **L**, denoted $G := \langle \mathbf{a}, \mathbf{L} \rangle$.

Definition 2.5 ([Cho+06])**.** The **Strong Diffie-Hellman Assumption** (SDH) states that for every efficient adversary $\mathcal{A}$ and degree bound $d \in \mathbb{N}$ the following probability is negligible in λ:

$$\Pr\left[C = [\tfrac{1}{\beta + c}]_1 \;\middle|\; \begin{array}{c} \langle group \rangle \leftarrow SampleGrp(1^\lambda) \\ \beta \leftarrow \mathbb{F}_q \\ \Sigma \leftarrow \{[\beta^i]_1\}_{i=0}^d, [\beta]_2\} \\ (c, C) \leftarrow \mathcal{A}(\langle group \rangle, \Sigma) \end{array} \right]$$

2.5 zkSNARks

Let $\mathcal{R}$ be a family of universal relations. Given a relation $\mathrm{R} \in \mathcal{R}$ and an instance x we call w a *witness* for x if $(\mathrm{x}, \mathrm{w}) \in \mathrm{R}$, $\mathcal{L}(\mathrm{R})\{x|\exists \mathrm{w} : (\mathrm{x}, \mathrm{w}) \in \mathrm{R}\}$ is the language of all the x that have a witness w in the relation R, while $\mathcal{L}(\mathcal{R})$ is the language of all the pairs (x, R) such that $\mathrm{x} \in \mathcal{L}(\mathrm{R})$.

Definition 2.6. A Universal Succinct Non-Interactive Argument of Knowledge is a tuple of PPT algorithms $(\mathrm{KetGen}, \mathrm{KeyGenD}, \mathrm{Prove}, \mathrm{Verify}, \mathrm{Simulate})$ such that:

- $(\mathrm{srs_u}, \tau) \leftarrow \mathrm{KeyGen}(\mathcal{R})$: On input a family of relations $\mathcal{R}$, KeyGen outputs a universal structured common reference string $\mathrm{srs_u}$ and a trapdoor τ ;
- $\mathrm{srs_R} \leftarrow \mathrm{KeyGenD}(\mathrm{srs_u}, \mathrm{R})$: On input $\mathrm{R} \in \mathcal{R}$, this algorithm outputs a relation dependent SRS that includes $\mathrm{srs_u}$;
- $\pi \leftarrow \mathrm{Prove}(\mathrm{R}, \mathrm{srs_R}, (\mathrm{x}, \mathrm{w}))$: On input the relation $\mathrm{srs_R}$ and a pair $(\mathrm{x}, \mathrm{w}) \in \mathrm{R}$, it outputs a proof π;
- $1/0 \leftarrow \mathrm{Verify}(\mathrm{srs_R}, \mathrm{x}, \pi)$: Verify takes an input $\mathrm{srs_R}$, the instance x and the proof and produces a bit expressing acceptance (1), or rejection (0);
- $\pi_{sim} \leftarrow \mathrm{Simulate}(\mathrm{R}, \tau, \mathrm{x})$: The simulator has the relation R, the trapdoor τ and the instance x as inputs and it generates a simulated proof π_{sim};

and that satisfies the properties of completeness, succinctness and ϵ-knowledge soundness as defined below.

Definition 2.7. Completeness holds if an honest prover will always convince an honest verifier. Formally, $\forall \mathrm{R} \in \mathcal{R}, (\mathrm{x}, \mathrm{w}) \in \mathrm{R}$,

$$\Pr\left[\mathrm{Verify}(\mathrm{srs_R}, \mathrm{x}, \pi) = 1 \;\middle|\; \begin{array}{c} (\mathrm{srs_u}, \tau) \leftarrow \mathrm{KeyGen}(\mathcal{R}) \\ \mathrm{srs_R} \leftarrow \mathrm{KeyGenD}(\mathrm{srs_u}, \mathrm{R}) \\ \pi \leftarrow \mathrm{Prove}(\mathrm{R}, \mathrm{srs_R}, (\mathrm{x}, \mathrm{w})) \end{array}\right] = 1$$

Definition 2.8. Succinctness holds if the size of the proof π is $poly(\lambda + \log|\mathrm{w}|)$ and Verify runs in time $poly(\lambda + |\mathrm{x}| + \log|\mathrm{w}|)$.

Definition 2.9. ϵ-knowledge soundness captures the fact that a cheating prover cannot, with probability at most ϵ, creat a proof π accepted by the verification algorithm unless it has a witness w such that $(\mathrm{x}, \mathrm{w}) \in \mathrm{R}$. Formally, for all PPT adversaries $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}$ such that:

$$\Pr\left[(\mathrm{x}, \mathrm{w}) \notin \mathrm{R} \wedge \mathrm{Verify}(\mathrm{srs_R}, \mathrm{x}, \pi) = 1 \;\middle|\; \begin{array}{c} (\mathrm{srs_u}, \tau) \leftarrow \mathrm{KeyGen}(\mathcal{R}) \\ \mathrm{R} \leftarrow \mathcal{A}(srs_u) \\ srs_R \leftarrow \mathrm{KeyGenD}(\mathrm{srs_u}, \mathrm{R}) \\ (\mathrm{x}, \pi) \leftarrow \mathcal{A}(\mathrm{R}, \mathrm{srs_R}) \\ \mathrm{w} \leftarrow \mathcal{E}(\mathrm{srs_R}, \mathrm{x}, \pi) \end{array}\right] \leq \epsilon$$

Definition 2.10. $(\text{KeyGen}, \text{KeyGenD}, \text{Prove}, \text{Verify}, \text{Simulate})$ is zero-knowledge (a zkSNARK) if for all $\text{R} \in \mathcal{R}$, instances x and PPT adversaries $\mathcal{A}$.

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

Definition 2.11. A universal zkSNARK is *updatable* if anyone can update its SRS in a verifiable manner by adding his own randomness.

Definition 2.12. A family of polynomial time computable relations $\mathcal{R}$ is field dependent if each $\text{R} \in \mathcal{R}$, specifies a unique finite field. More precisely, for any pair $(\text{x}, \text{w}) \in \text{R}$, x specifies the same finite field $\mathbb{F}_\text{R}$ (simply denoted as $\mathbb{F}$ if there is no ambiguity).

CHAPTER 3

POLYNOMIAL COMMITMENTS

Commitment schemes (CS) emulate real world envelopes, i.e. they allow a *party* to conceal a message, with the ability to later reveal it, while being unable to change it. Thus, they can be thought of as functions C, that can have the following properties:

- *Binding.* It is hard (impossible) to find messages m_1, m_2 such that $C(m_1) = C(m_2)$.
- *Hiding.* It is hard (impossible), given commitments c_1, c_2 to any messages m_1, m_2 to distinguish which corresponds to which.

A commitment scheme given by a function that fulfils these properties is called computationally (perfectly) binding CS and computationally (perfectly) hiding CS, respectively. Note that a CS cannot be perfectly binding and perfectly hiding simultaneously.

In the context of interactive oracle proof, where the prover constructs oracles which the verifier can query in order to decide the correctness of a statement, commitments play a crucial part. This is because, by making the prover publish commitments to the oracles he constructs it opens a way to validate that the answers the verifier receives are consistent with the oracles, i.e. the prover did not change the oracles during the proof.

For the construction of pre-processing zkSNARKs, the witness to a statement is modified to witness polynomials that satisfy some set of polynomial equations between them and publicly known ones. Now, the Schwartz-Zippel lemma tells us that if a polynomial equation over a field is satisfied when the polynomials are evaluated at a random point, then the equation holds in general with probability $1 - d/|\mathbb{F}|$, where d is the maximum degree. The basic idea is that, these witness polynomials will be constructed by the prover as oracles and later the verifier can query them at a random point, in order to be convinced about the statement with high-probability. In order to enforce the prover to stick to his original oracle polynomials and not change them along the way, we make him publish commitments to them.

That being said, by leveraging the algebraic flavor of the information theoretic part of the zkSNARKs (e.g. PHPs), what we need for the compiler is a cryptographic primitive that we call *polynomial commitment scheme.* By using different commitment schemes for the compiler, we end up with zkSNARK constructions with different properties.

Informally, a *polynomial commitment scheme* [KZG10] allows a prover to produce a commitment c to a univariate polynomial $p \in \mathbb{F}[X]$, and later "open" $p(X)$ at any point $z \in \mathbb{F}$, providing also an *evaluation proof* π showing that the opened value is consistent with the polynomial "contained" in c at z.

A straightforward polynomial commitment scheme works as follows:

- Pick a cyclic group $\mathbb{G} = \langle g \rangle$ of order q.
- To commit to a polynomial $f(X) = a_0 + a_1 X + \ldots + a_n X^n$, produce commitments to coefficients, $c_0 = g^{a_0}, c_1 = g^{a_1}, \ldots, c_n = g^{a_n}$ and publish them.
- For input an evaluation point $x \in Z_q$, output the evaluation $u = f(x)$ and evaluation proof $\pi = \perp$
- To verify that an input u is the evaluation of the polynomial corresponding to a given commitment at x, calculate $c_0 c_1^x c_2^{x^2} \ldots c_n^{x^n}$ and check if it equals g^u.

The PC scheme is perfectly binding, because by sending commitment to the coefficient of the polynomial, the prover gives the ability to the verifier to evaluate the polynomial at any point, but this evaluation is hidden in the exponent. Then, for a given evaluation point, the only way for the prover to pass the test is to actually send $f(x)$, as this is the only exponent that will produce the same element the verifier computes. The main drawback of this natural construction is that the size of the commitment grows linearly with the degree of the committed polynomials, which makes it impractical. Thus turning this informal goal into a formal definitions requires some care.

3.1 Towards efficient polynomial commitment schemes

When constructing a polynomial commitment scheme we place strong efficiency requirements, namely we require that the commitment, evaluation proof length and verification are much smaller than the polynomial itself, e.g. $polylog(d)$. To achieve these efficiency standards, some initialization phase must take place, where a *common reference string* (CRS) is created. Of course, by embedding structure we can hope for less costly scheme, but this comes with a prize in security, as to produce *structured reference string* (SRS) trust must be put on some party. Another way of trading security for efficiency, is by constructing schemes that are secure based on assumptions about the computational power of the adversary.

Different schemes provide different trade-offs between efficiency and security. FRI polynomial commitment scheme [VP19] uses hash functions and Reed-Solomon codes, to achieve single element commitment, proof of evaluation size and verification of $\mathcal{O}\big(log^2(d)\big)$, while also being *transparent*, i.e. does not require trusted setup, while also being post-quantum secure. Based on the inner product argument (IPA) with the folding technic introduced by Bootle et al. [Boo+16] and later inproved in Bulletproofs [Bün+17], transparent single element polynomial commitment schemes can be build with CRS size d, $\mathcal{O}(d)$ verification time, $log(d)$ size evaluation proof, while if we use a SRS the construction of [Bün+19] requires $\sqrt{d}$ SRS size and $log(d)$ verification time. Finally, in [KZG10] they propose a construction in the SRS model using elliptic curves

and pairings, that achieves the best efficiency results, namely a commitment and evaluation proof consist of a single element, and verification is done through a single pairing, while also keeping an SRS of size d.

Furthermore, a critical point in the construction of these schemes is how they ensure that an adversary that produces a validating tuple of commitment-evaluation-proof actually *knows* a polynomial that respects a certain bound that gives rise to the said tuple. This is the notion of *extractability*.

Another thing to take into consideration is that, in many applications of polynomial commitments, an adversary may need to produce commitments to multiple polynomials within a round of interaction or across multiple rounds. Then he may need to reveal values of all of these polynomials at one point or different locations. This motivates the following considerations. First, we should rely on a single set of public parameters, even if the polynomials to be committed differ in degree. Secondly, it is desirable to have a batching mechanism for the commitments as well as the evaluation proofs to save on communication costs.

3.2 Polynomial Commitment schemes in Marlin [Chi+19]

Because of the aforementioned, in Marlin [Chi+19] they introduce an enhanced version of the polynomial commitments introduced in [KZG10], that captures the desired properties. Their definition of a polynomial commitment scheme PC consists of a tuple of algorithms PC=(Setup,Trim,Commit,Open,Check). The setup algorithm is probabilistic and takes as input a security parameter and maximum degree bound D, and outputs public parameters pp that contain the description of a finite field $\mathbb{F}$. The "trimming" algorithm then deterministically specializes these parameters for a set of given degree bounds and outputs a commiter key ck and a receiver key rk. The sender then can invoke the commit algorithm with input ck and a list of polynomials **p** with respective degree bounds **d**, generating a set of commitments **c**. Then, on input a query set $\mathcal{Q}$, the sender can use the opening algorithm to produce an evaluation proof π that convinces the receiver that the polynomials inside the commitment respect their degree bounds and also that the claimed evaluations **u** are correctly computed. Then, the receiver can use the check algorithm to validate the proof.

Also, they present two different commitment schemes that solve under different assumptions the problem of "knowing a polynomial corresponding to a commitment". The first construction works in the standard model under knowledge assumptions, while the second one works in the Algebraic Group Model. Moreover, they first build these schemes to support one degree bound (the maximum one) and evaluation at one point for all the polynomials. Afterwords, they extend these constructions to support multiple degree bounds and evaluation at different points. For simplicity, we only present the construction at the Algebraic Group Model and we omit its aforementioned extensions. A reason for this is because for the purpose of zkSNARKs we will assume that the adversary is algebraic and also all the polynomials are of maximum degree and we will evaluate them all at one point. Also, for the zkSNARKs, we won't need the polynomial commitment scheme to be hiding (as this is achieved in a different way) but we present this extension because of its usefulness in different settings.

3.3 Definitions for polynomial commitments

Definition 3.1. A polynomial commitment scheme over a field family $\mathcal{F}$ is a tuple of algorithms PC = (Setup, Trim, Commit, Open, Check) with the following syntax.

- PC.Setup$(1^\lambda, D) \to$ pp. On input a security parameter λ (in unary), and a maximum degree bound $D \in \mathbb{N}$, PC.Setup samples public parameters pp. The parameters contain the description of a finite field $\mathbb{F} \in \mathcal{F}$.
- PC.Trim$^{\text{pp}}(1^\lambda, \mathbf{b}) \to (\text{ck}, \text{rk})$. Given oracle access to public parameters pp, and on input a security parameter λ (in unary), and degree bounds $\mathbf{b}$, PC.Trim deterministically computes a key pair (ck, rk) that is specialized to $\mathbf{b}$.
- PC.Commit$(\text{ck}, \mathbf{p}, \mathbf{d}; \omega) \to \mathbf{c}$. On input ck, univariate polynomials $\mathbf{p} = [p_i]_{i=1}^n$ over the field $\mathbb{F}$, and degree bounds $\mathbf{d} = [d_i]_{i=1}^n$ with $deg(p_i) \leq d_i \leq D$, PC.Commit outputs commitments $\mathbf{c} = [c_i]_{i=1}^n$ to the polynomials $\mathbf{p} = [p_i]_{i=1}^n$. The randomness $\omega = [\omega_i]_{i=1}^n$ is used if the commitments $\mathbf{c} = [c_i]_{i=1}^n$ are hiding.
- PC.Open$(\text{ck}, \mathbf{p}, \mathbf{d}, Q, \xi; \omega) \to \pi$. On iput ck, univariate polynomials $\mathbf{p} = [p_i]_{i=1}^n$, degree bounds $\mathbf{d} = [d_i]_{i=1}^n$, a query set Q consisting of tuples $(i, z) \in [n] \times \mathbb{F}$, and opening challenge ξ, PC.Open outputs an evaluation proof π. The randomness ω must equal the one previously used in PC.Commit.
- PC.Check$(\text{rk}, \mathbf{c}, \mathbf{d}, Q, \mathbf{u}, \pi, \xi) \in \{0, 1\}$. On input rk, commitments $\mathbf{c} = [c_i]_{i=1}^n$, degree bounds $\mathbf{d} = [d_i]_{i=1}^n$, query set Q consisting of tuples $(i, z) \in [n] \times \mathbb{F}$, alleged evaluations $\mathbf{u} = (u_{(i,z)})_{(i,z)\in Q}$, evaluation proof π, and opening challenge ξ, PC.Check outputs 1 if π attests that, for every $(i, z) \in Q$, the polynomial p_i committed in c_i has degree at most d_i and evaluates to $u_{(i,z)}$ at z

The polynomial commitment scheme satisfies the properties of completeness and extractability defined below.
The polynomial commitment scheme is (perfectly) hiding if it satisfies the hiding property defined below.

Definition 3.2. (**Completeness**). For every maximum degree bound $D \in \mathbb{N}$ and efficient adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c}\deg(\mathbf{p}) \leq \mathbf{d} \leq D \\ \Downarrow \\ \text{PC.Check}(\text{rk}, \mathbf{c}, \mathbf{d}, Q, \mathbf{u}, \pi, \xi) = 1\end{array} \;\middle|\; \begin{array}{c}\text{pp} \leftarrow \text{PC.Setup}(1^\lambda, D) \\ (\mathbf{p}, \mathbf{d}, Q, \xi, \omega) \leftarrow \mathcal{A}(\text{pp}) \\ (\text{ck}, \text{rk}) \leftarrow \text{PC.Trim}^{\text{pp}}(1^\lambda, \mathbf{d}) \\ \mathbf{c} \leftarrow \text{PC.Commit}(\text{ck}, \mathbf{p}, \mathbf{d}; \omega) \\ \mathbf{u} \leftarrow \mathbf{p}(Q) \\ \pi \leftarrow \text{PC.Open}(\text{ck}, \mathbf{p}, \mathbf{d}, Q, \xi, \omega)\end{array}\right] = 1$$

Informally, PC is *extractable* if for every maximum degree D and every efficient sender adversary $\mathcal{A}$ who produces degree bound d, a commitment c, evaluation u and evaluation proof π such that PC.Check accepts, there exists a corresponding extracor $\mathcal{E}_\mathcal{A}$ that outputs a polynomial of degree at most d that "explains" c so that $p(z) = u$.

Definition 3.3. (**Extraxtability**). For every maximum degree bound $D \in \mathbb{N}$ and efficient adversary $\mathcal{A}$ there exists an efficient extractor $\mathcal{E}$ such that for every round bound $e \in \mathbb{N}$, efficient public-coin challenger $\mathcal{C}$, efficient query sampler $\mathcal{Q}$ and efficient adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ the probability below is negligibly close to 1 (as a function of λ):

$$
\Pr\left[\begin{array}{c|c}
 & \mathrm{pp} \leftarrow \mathrm{PC.Setup}(1^\lambda, D) \\
\hline
 & \begin{array}{c} \text{For } i = 1, \ldots, \mathbf{r}: \\ \rho_i \leftarrow \mathcal{C}(\mathrm{pp}, i) \\ (\mathbf{c}_i, \mathbf{d}_i) \leftarrow \mathcal{A}(\mathrm{pp}, [\rho_j]_{j=1}^{i}) \\ \mathbf{p}_i \leftarrow \mathcal{E}(\mathrm{pp}, [\rho_j]_{j=1}^{i}) \end{array} \\
\hline
\begin{array}{c} \mathrm{PC.Check}(\mathbf{rk}, \mathbf{c}, \mathbf{d}, Q, \mathbf{u}, \pi, \xi) = 1 \\ \Downarrow \\ \deg(\mathbf{p}) \leq \mathbf{d} \leq D \text{ and } \mathbf{u} = \mathbf{p}(Q) \end{array} & \begin{array}{c} Q \leftarrow \mathcal{Q}(\mathrm{pp}, [\rho_j]_{j=1}^{\mathbf{r}}) \\ (\mathbf{u}, \mathbf{st}) \leftarrow \mathcal{B}_1(\mathrm{pp}, [\rho_j]_{j=1}^{\mathbf{r}}, Q) \\ \text{Sample opening challenge } \xi \\ \pi \leftarrow \mathcal{B}_2(\mathbf{st}, \xi) \\ \text{Set } [c_i]_{i=1}^{n} \coloneqq [\mathbf{c}_i]_{i=1}^{\mathbf{r}}, [p_i]_{i=1}^{n} \coloneqq [\mathbf{p}_i]_{i=1}^{\mathbf{r}}, [d_i]_{i=1}^{n} \coloneqq [\mathbf{d}_i]_{i=1}^{\mathbf{r}} \\ \mathbf{ck}, \mathbf{rk} \leftarrow \mathrm{PC.Trim}^{\mathrm{pp}}(1^\lambda, [d_i]_{i=1}^{n}) \\ \text{Define the set of queried polynomials } T \coloneqq \{i \in [n] \mid (i, z) \in Q\} \\ \text{Set } \mathbf{c} \coloneqq [c_i]_{i \in T}, \mathbf{p} \coloneqq [p_i]_{i \in T}, \mathbf{d} \coloneqq [d_i]_{i \in T} \end{array}
\end{array}\right]
$$

There are two notions of efficiency we require for PC. First, the time required to commit to a polynomial and then create an evaluation proof, should be proportional to its degree and not the maximum supported degree. Secondly, on the receiver's side, the commitment size, proof size and time to verify an evaluation should independent of the degrees of the polynomials.

Definition 3.4. (**Efficiency**). We say that a polynomial commitment scheme is:

- **degree − efficient** if the time to run PC.Open is proportional to the maximum degree $\max(\mathbf{d})$ (as opposed to the maximum supported degree D). In particular this implies that $|\mathbf{ck}| = \mathcal{O}(\max(\mathbf{d}))_\lambda$.
- **succinct** if the size of the commitments, the size of the evaluation proofs, and the time to check an opening are all independent of the degree of the committed polynomials. That is, $|\mathbf{c}| = n \cdot \mathrm{poly}(\lambda) \cdot \mathrm{polylog}(d)$, $|\pi| = |Q| \cdot \mathrm{poly}(\lambda) \cdot \mathrm{polylog}(d)$, $|\mathbf{rk}| = \mathcal{O}(n)_\lambda$, and time(Check) $= (n + |Q|) \cdot \mathrm{poly}(\lambda) \cdot \mathrm{polylog}(d)$.

The hiding property of PC states that commitments and proofs of evaluation reveal no information about the committed polynomial beyond the publicly stated degree bound and evaluation itself. Informally, PC is *hiding* if there exists an efficient simulator that outputs simulated commitments and evaluation proofs that cannot be distinguished from their real counterparts by any malicious distinguisher that only knows the degree bound and the evaluation.

Definition 3.5. (**Hiding**). There exists a polynomial-time simulator $\mathcal{S} = (\mathrm{Setup}, \mathrm{Commit}, \mathrm{Open})$ such that, for every maximum degree bound $D \in \mathbb{N}$, and efficient adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, the probability that b=1 in the following two experiments is identical:

$\text{Real}(1^\lambda, D, \mathcal{A})$:

1. $\text{pp} \leftarrow \text{PC.Setup}(1^\lambda, D)$.
2. Letting $\mathbf{c_0} = \perp$, for $i = 1, \ldots, r$
   (a) $(\mathbf{p}_i, \mathbf{d}_i, h_i) \leftarrow \mathcal{A}_1(\text{pp}, \mathbf{c_0}, \mathbf{c_1}, \ldots, \mathbf{c}_{i-1})$
   (b) $(\text{ck}_\text{i}, \text{rk}_\text{i}) \leftarrow \text{PC.Trim}^{\text{pp}}(1^\lambda, \mathbf{d}_i)$
   (c) If $h_i = 0$ sample commitment randomness ω_i
   (d) If $h_i = 1$ set randomness ω_i to $\perp$
   (e) $\mathbf{c}_i \leftarrow \text{PC.Commit}(\text{ck}_i, \mathbf{p}_i, \mathbf{d}_i, \omega_i)$
3. $\mathbf{c} := [\mathbf{c}_i]_{i=1}^r, \mathbf{p} := [\mathbf{p}_i]_{i=1}^r, \mathbf{d} := [\mathbf{d}_i]_{i=1}^r, \omega := [\omega_i]_{i=1}^r$
4. $(\text{ck}, \text{rk}) \leftarrow \text{PC.Trim}^{\text{pp}}(1^\lambda, \mathbf{b})$
5. $([Q_j]_{j=1}^\tau, [\xi_j]_{j=1}^\tau, \mathbf{st}) \leftarrow \mathcal{A}_2(\mathbf{pp}, \mathbf{c})$
6. For $j \in [\tau], \pi_j \leftarrow \text{PC.Open}(\text{ck}, \mathbf{p}, \mathbf{d}, q_j, \xi_j; \omega)$
7. $b \leftarrow \mathcal{A}_3(\mathbf{st}, [\pi]_{j=1}^\tau)$

$\text{Ideal}(1^\lambda, D, \mathcal{A})$:

1. $(\text{pp}, \text{trap}) \leftarrow \mathcal{S}.\text{Setup}(1^\lambda, D)$
2. Letting $\mathbf{c}_0 := \perp$, for $i = 1, \ldots, r$
   (a) $(\mathbf{p}_i, \mathbf{d}_i, h_i) \leftarrow \mathcal{A}_1(\text{pp}, \mathbf{c_0}, \mathbf{c_1}, \ldots, \mathbf{c_{i-1}})$
   (b) $(\text{ck}_\text{i}, \text{rk}_\text{i}) \leftarrow \text{PC.Trim}^{\text{pp}}(1^\lambda, \mathbf{d}_i)$
   (c) If $h_i = 0$, sample randomness ω_i and compute simulated commitments $\mathbf{c}_i \leftarrow \mathcal{S}\text{Commit}(\text{trap}, \mathbf{d}_i; \omega_i)$
   (d) If $h_i = 1$, set $\omega_i := \perp$ and compute (real) commitments $\mathbf{c}_i \leftarrow \text{PC.Commit}(\text{ck}_i, \mathbf{p}_i, \mathbf{d}_i; \omega_i)$
3. $\mathbf{c} := [c_i]_{i=1}^r, \mathbf{p} := [\mathbf{p}_i]_{i=1}^r, \mathbf{d} := [\mathbf{d}_i]_{i=1}^r, \omega := [\omega_i]_{i=1}^r$
4. $(\text{ck}, \text{rk}) \leftarrow \text{PC.Trim}^{\text{pp}}(1^\lambda, \mathbf{d})$
5. $([Q_j]_{j=1}^\tau, [\xi_j]_{j=1}^\tau, \mathbf{st}) \leftarrow \mathcal{A}_2(\text{pp}, \mathbf{c})$
6. Zero out hidden polynomials $\mathbf{p}' := [h_i \mathbf{p}_i]_{i=1}^r$
7. For $j \in [\tau], \pi_j \leftarrow \mathcal{S}.\text{Open}(\text{trap}, \mathbf{p}', \mathbf{p}(Q_j), \mathbf{d}, Q_j, \xi_j; \omega)$
8. $b \leftarrow \mathcal{A}_3(\mathbf{st}, [\pi]_{j=1}^\tau)$

3.4 Overview of the construction in Marlin

We present a polynomial commitment scheme for a single degree bound $D \in \mathbb{N}$ and for query set Q that consists only of a single element $z \in \mathbb{F}_q$. The scheme breaks into two constructions, one that produces non-hiding commitments which we call nhPC_s and another that includes the texts in blue which we call phPC_s for hiding commitments.

The setup phase samples a cryptographically secure bilinear group and then samples a committer key ck and receiver key rk for a given degree bound D. The ck consists of powers of the trapdoor $\beta \in \mathbb{F}_q$ encoded in the first group and also another element $\gamma \in \mathbb{F}_q$ multiplied by the powers of β encoded in the first group, which is going to be used to make the commitment hiding. The receiver key rk consists of the group elements $(D, \langle \text{group} \rangle, [\gamma]_1, [\beta]_2)$. Note that the SRS is updatable, meaning that anyone can produce another one from it by adding his own randomness thus minimizing trust. A way to do this, is by taking the SRS and a $\delta \in \mathbb{F}_q$ and multiplying each element of the SRS by the appropriate power of δ as well as providing a proof that he did this correctly done, e.g. the division of consecutive elements in the SRS have the same discrete logarithm.

To commit to a polynomial $p \in \mathbb{F}_q[X]$, the sender computes $c := [p(\beta)]_1$ (adding

another polynomial to make it hiding). To prove that the committed polynomial evaluates to u at a point z, we rely to the fact that the expression $w(X) := (p(X)-u)/(X-z)$ is a polynomial if and only if $p(z) = u$. Otherwise, it is a rational function and the commitment to it (which is the proof of evaluation π) cannot be computed using ck.

To verify the proof of evaluation, we use the bilinear mapping property (i.e. $e([\alpha]_1, [\alpha_2) = e([1]_1, [1]_2)^{\alpha \cdot \alpha'}$) to check that the equality $e(c - [u]_1, [1]_2) = e(\pi, [\beta]_2 - [z]_2)$ holds.

To commit to multiple polynomials at once, the sender requests from the receiver a random field element ξ, which he uses to take a linear combination of the polynomials: $p = \sum_{i=1}^{n} \xi^i p_i$ and generates a proof of evaluation π for this polynomial. The receiver verifies the proof by using the fact that the commitments are additively homomorphic. The receiver computes $c = \sum_{i=1}^{n} \xi^i c_i$ and $u = \sum_{i=1}^{n} \xi^i u_i$ and checks that the pairing equation holds for c, π, u. Completeness of this batched check is immediate, while soundness follows from the fact that if any polynomial does not match its evaluation, then the combined polynomial will not match its evaluation with high probability due to Schwartz-Zippel lemma.

3.5 PC scheme construction in the AGM

Setup. On input a security parameter λ (in unary), and a maximum degree bound $D \in \mathbb{N}$, $\mathrm{PC_s}$.Setup samples public parameters $(\mathsf{ck}, \mathsf{rk})$ as follows. Sample a bilinear group $\langle \mathsf{group} \rangle \leftarrow \mathrm{SampleGrp}(1^\lambda)$, and parse $\langle \mathsf{group} \rangle$ as a tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, G, H, e)$. Sample random elements $\beta, \gamma \in \mathbb{F}_q$. Then compute the vector

$$\Sigma := \begin{pmatrix} [1]_1 & [\beta]_2 & [\beta^2]_1 & \cdots & [\beta^D]_1 \\ [\gamma]_1 & [\gamma\beta]_1 & [\gamma\beta^2]_1 & \cdots & [\gamma\beta^D]_1 \end{pmatrix} \in \mathbb{G}_1^{2D+2}$$

Set $\mathsf{ck} := (\langle \mathsf{group} \rangle, \Sigma)$ and $\mathsf{rk} := (D, \langle \mathsf{group} \rangle, [\gamma]_1, [\beta]_2)$, and then output the public parameters $(\mathsf{ck}, \mathsf{rk})$. These public parameters will support polynomials over the field $\mathbb{F}_q$ of degree at most D.

Commit. On input ck, univariate polynomials $\mathbf{p} := [p_i]_{i=1}^n$ over $\mathbb{F}_q$ and randomness $\omega := [\omega_i]_{i=1}^n$, $\mathrm{PC_s}$.Commit outputs commitments $\mathbf{c} := [c_i]_{i=1}^n$ that are computed as follows. If for any $p_i \in \mathbf{p}, \deg(p_i) > D$, abort. Else, for each $i \in [n]$, if ω_i is not $\perp$, then obtain randomunivariate polynomial $\bar{p}_i$ of degree $\deg(p_i)$ from ω_i, otherwise $\bar{p}_i$ is set to be a zeropolynomial. For each $i \in [n]$, output $c_i := [p_i(\beta)]_1 + [\gamma \bar{p}_i(\beta)]_1$. Note that because p_i and $\bar{p}_i$ have degree at most D, the above terms are linear combinations of terms in ck.

Open. On input ck, univariate polynomials $\mathbf{p} := [p_i]_{i=1}^n$ over $\mathbb{F}_q$, evaluation point $z \in \mathbb{F}_q$, opening challenge $\xi \in \mathbb{F}_q$, and randomness $\omega := [\omega_i]_{i=1}^n$, which is the same randomness used for $\mathrm{PC_s}$.Commit, $\mathrm{PC_s}$.Open outputs an evaluation proof $\pi \in \mathbb{G}_1$ that is computed as follows. If for any $p_i \in \mathbf{p}, \deg(p_i) > D$, abort. For each $i \in [n]$, if ω_i is not $\perp$, then obtain random univariate polynomial $\bar{p}_i$ of degree $\deg(p_i)$ from ω_i, otherwise $\bar{p}_i$ is set to be a zero polynomial. Then compute the linear combination of polynomials $p(X) := \sum_{i=1}^{n} \xi^i p_i(X)$ and $\bar{p}(X) := \sum \xi^i \bar{p}_i(X)$. Compute witness polynomials $w(X) := \frac{p(X)-p(z)}{X-z}$ and $\bar{w}(X) := \frac{\bar{p}(x)-\bar{p}(z)}{X-z}$. Set $\mathrm{w} := [w(\beta)]_1 + [\gamma \bar{w}(\beta)]_1 \in \mathbb{G}_1$ and $\bar{u} := \bar{p}(z) \in \mathbb{F}_q$. The evaluation proof is $\pi := (\mathrm{w}, \bar{u})$.

Check. On input rk, commitments $\mathbf{c} := [c_i]_{i=1}^n$, evaluation point $z \in \mathbb{F}_q$, alleged evaluations $\mathbf{u} := [u_i]_{i=1}^n$, evaluation proof $\pi := (\mathrm{w}, \bar{u})$, and randmness $\xi \in \mathbb{F}_q$, $\mathrm{PC_s}$.Check proceeds as follows. Compute the linear combination $C := \sum_{i=1}^n \xi^i c_i$. Then compute the linear combination of evaluations $u := \sum_{i=1}^n \xi^i u_i$, and check the evaluation proof via the equality $e(C - [u]_1 - [\gamma \bar{u}]_1, H) = e(\mathrm{w}, [\beta]_2 - [z]_2)$.

3.6 Proofs of properties

Completeness. Fix any maximum degree bound D and efficient adversary $\mathcal{A}$. Let $(\mathrm{ck}, \mathrm{rk})$ be any key pair output by the algorithm $\mathrm{PC_s}.\mathrm{Setup}(1^\lambda, \mathrm{D})$ constructed above. The keys contain a description $\langle group \rangle$ of a bilinear group of some prime order q, which in particular induces a field $\mathbb{F}_q$.
Let $\mathcal{A}(\mathrm{ck}, \mathrm{rk})$ select polynomials $\mathbf{p} = [p_i]_{i=1}^n$ over $\mathbb{F}_q$, location $z \in \mathbb{F}_q$ and opening challenge $\xi \in \mathbb{F}_q$. We only need to consider adversaries $\mathcal{A}$ that make choices for which $\deg(\mathbf{p}) \leq D$. Now consider commitments $\mathbf{c} = [c_i]_{i=1}^n$ and evaluation proof π that are all computed according to the construction above.
We need to show that, for correct evaluations $\mathbf{u} := \mathbf{p}(z)$,

$$\mathrm{PC_s}.(\mathrm{Check})(\mathrm{rk}, \mathbf{c}, z, \mathbf{u}, \pi, \xi) = 1.$$

This amounts to arguing that the pairing equation holds.

$$\begin{aligned}
e(C - [u]_1 - [\gamma \bar{u}]_1, [1]_2) &= e(\sum_{i=1}^n \xi^i c_i - \sum_{i=1}^n \xi^i [u_i]_1 - [\gamma \bar{p}(z)]_1, [1]_2) \\
&= e([\sum_{i=1}^n \xi^i p_i(\beta) + \gamma \sum_{i=1}^n \bar{p}_i(\beta) - \sum_{i=1}^n \xi^i p_i(z) - \gamma \bar{p}(z)]_1, [1]_2) \\
&= e([p(\beta) - \gamma \bar{p}(\beta) - p(z) - \gamma \bar{p}(z)]_1, [1]_2) \\
&= e([w(\beta)(\beta - z) + \gamma \bar{w}(\beta)(\beta - z)]_1, [1]_2) \\
&= e([w(\beta)]_1 + [\gamma \bar{w}(\beta)]_1, [\beta]_2 - [z]_2) \\
&= e(w, [\beta]_2 - [z]_2)
\end{aligned}$$

Succinctness. For a list of n polynomials, the scheme $\mathrm{PC_s}$ requires 2n $\mathbb{G}_1$ elements for the commitment and one $\mathbb{G}_1$ element and one $\mathbb{F}_q$ elementfor the evaluation proof, while the time to check this proof requires two variable-base multi-scalar multiplications of size n and two pairings.

Extractability [Chi+19].

Theorem 3.6. If the bilinear group sampler SampleGrp satisfies SDH assumption against algebraic adversaries, nhPC$_s$ and phPC$_s$ achieve extractability against algebraic adversaries.

Definition 3.7. PC$_s$ satisfies **evaluation binding** if for every maximum degree $D \in \mathbb{N}$ and efficient adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the following probability is negligible in the security parameter λ:

$$\Pr\left[\begin{array}{c} \mathbf{u} \neq \mathbf{u}' \\ \wedge \\ \text{PC}_s.\text{Check}(\text{rk},\mathbf{c}, z, \mathbf{u}, \pi, \xi) = 1 \\ \wedge \\ \text{PC}_s.\text{Check}(\text{rk},\mathbf{c}, z, \mathbf{u}, \pi', \xi) = 1 \end{array} \middle| \begin{array}{c} (\text{ck}, \text{rk}) \leftarrow \text{PC}_s.\text{Setup}(1^\lambda, \text{D}) \\ (\mathbf{c}, z, \mathbf{u}, \mathbf{u}', \mathbf{st}) \leftarrow \mathcal{A}_1(\text{ck}, \text{rk}) \\ \text{Sample opening challenge } \xi \\ (\pi, \pi') \leftarrow \mathcal{A}_2(\mathbf{st}, \xi) \end{array}\right]$$

Lemma 3.8. If the bilinear group sampler SampleGrp satisfies the SDH, nhPC$_s$ and phPC$_s$ achieve evaluation binding.

We don't expect a computationally bounded adversary to be able to produce two different evaluations for a fixed commitment, because this amounts to being able to perform "division" with powers of β, γ, which can't happen (he is bounded to perform only linear algebra computations with the elements of the SRS). Thus, we should be able to use a succesful adversary against evaluation binding to output elements that include some kind of "division" with the unknown parameters.

Proof. Suppose for contradiction that there exists a maximum degree bound D and an efficient adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks evaluation binding with non-negligible probability. We show that either $\mathcal{A}$ can be used to break DL with non-negligible probability or that we can $\mathcal{A}$ to construct an efficient adversary $\mathcal{B}$ that breaks SDH with non-negligible probability. Since SDH assumption implies DL assumption, in either case we obtain a contradiction that SDH holds with respect to SampleGrp. We define $\mathcal{B}$ as follows.

$\mathcal{B}(\langle \text{group} \rangle, \Sigma)$:

1 : Parse Σ as $\{[\beta^i]_1\}_{i=1}^{D}, [\beta]_2\}$

2 : Sample $a \leftarrow \mathbb{F}_q, \gamma \leftarrow \mathbb{F}_q^*$, and set

$\text{ck} = (\langle \text{group} \rangle, \{[\beta^i]_1, [\gamma\beta^i]_1\}_{i=0}^{n})$

$\text{rk} = (\langle \text{group} \rangle, [\gamma]_1, [\beta]_2)$

3 : Compute $(\mathbf{c}, z, \mathbf{u}, \mathbf{u}', \mathbf{st}) \leftarrow \mathcal{A}_1(\text{ck}, \text{rk})$

4 : Sample random opening challenge $\xi \in \mathbb{F}_q$

5 : Compute $(\pi, \pi') \leftarrow \mathcal{A}_2(\mathbf{st}, \xi)$

6 : Parse (π, π') as $((w, \bar{u}), (w', \bar{u}'))$, $\mathbf{u}$ as $[u_i]_{i=1}^n$, $\mathbf{u}'$ as $[u_i']_{i=1}^n$

7 : Compute $u = \sum_{i=1}^{n} \xi^i u_i, \quad u' = \sum_{i=1}^{n} \xi^i u_i'$

8 : If $[z]_1 = [\beta]_1$:

choose $a \in \mathbb{F}_q \setminus \{z\}$, output $(a, [\frac{1}{z+a}]_1)$ breaking SDH

9 : Else if $([z]_1 \neq [\beta]_1) \wedge (w \neq w')$:

output $(-z, \frac{1}{u' - u + \gamma(\bar{u}' - \bar{u})}(w - w'))$ breaking SDH

10 : Else abort.

First, we show that if either predicate in Step 8 or Step 9 is satisfied, then $\mathcal{B}$ does in fact break SDH. Next, we show that one of these predicates is satisfied with non-negligible probability when $\mathcal{A}$ breaks evaluation binding. We do this by showing that if $\mathcal{B}$ aborts but $\mathcal{A}$ still succeeds, then $\mathcal{A}$ can be used to solve the discrete logarithm problem in SampleGrp with non-negligible probability.

$\mathcal{B}$ succeeds if predicates are satisfied. If $\mathcal{A}$ outputs $z = \beta$, then $\mathcal{B}$ can construct an arbitary solution to the SDH problem. If on the other hand $(\beta \neq z) \wedge (w \neq w')$, then if $\mathcal{A}$ breaks evaluation binding, by construction of PC_s.Check the following equations must hold:

$$e(C - [u]_1 - [\gamma\bar{u}]_1, [1]_2) = e(w, [\beta]_2 - [z]_2) \tag{3.6.1}$$

$$e(C - [u']_1 - [\gamma\bar{u}']_1, [1]_2) = e(w', [\beta]_2 - [z]_2) \tag{3.6.2}$$

Since $(\beta \neq z) \wedge (w \neq w')$ the LHS of the equations are not equal, giving us that $u' - u + \gamma(\bar{u}' - \bar{u}) \neq 0$. Assuming some $\mu, \delta, \delta' \in \mathbb{F}_q$ such that $C = [\mu]_1, w = [\delta]_1, w' = [\delta']_1$, then substracting the above equations and using the properties of the bilinear map, we get:

$$
\begin{aligned}
(u' - u + \gamma(\bar{u}' - \bar{u}))e([1]_1, [1]_2) = (\delta - \delta')(\beta - z)e([1]_1, [1]_2) \Rightarrow \\
\frac{1}{\beta - z}e([1]_1, [1]_2) = \frac{1}{u' - u + \gamma(\bar{u}' - \bar{u})}(\delta - \delta')e([1]_1, [1]_2) \Rightarrow \\
e([\frac{1}{\beta - z}]_1, [1]_2) = e(\frac{1}{u' - u + \gamma(\bar{u}' - \bar{u})}(w - w'), [1]_2) \Rightarrow \\
\frac{1}{u' - u + \gamma(\bar{u}' - \bar{u})}(w - w') = [\frac{1}{\beta - z}]_1 \Rightarrow \\
(-z, \frac{1}{u' - u + \gamma(\bar{u}' - \bar{u})}(w - w')) \text{ breaks SDH}
\end{aligned}
$$

Probability that predicates are satisfied. We analyze the probability with which $\mathcal{B}$ aborts. The predicates are not satisfied exactly when $(\beta \neq z) \wedge (w = w')$. From the equations of PC_s.Check this breaks down in the following two cases:

- **Case 1:** $\bar{u} \neq \bar{u}'$. Then from the equations we get $u' - u + \gamma(\bar{u}' - \bar{u}) = 0$. This gives us $\gamma = \frac{u - u'}{\bar{u}' - \bar{u}}$ breaking the discrete logarithm assumption.

- **Case 2:** $\bar{u} = \bar{u}'$. In this case, from the equations it must hold that $u = u'$. Since $\mathbf{u} \neq \bar{\mathbf{u}}$ then the polynomial $\sum_{i=1}^{n} X^n(u_i - u_i')$ is not the 0 polynomial, sampling $\xi \in \mathbb{F}_q$ that is a root of it (and thus getting $u = u'$) occurs with probability at most $\frac{n}{q}$.

Hence, we conclude that if $(\beta \neq z) \wedge (w = w')$ with non-negligible probability and $\mathcal{A}$ still succeeds, then $\mathcal{A}$ can be used to break DL with non-negligible probability, which cannot occur if SDH is hard for SampleGrp.
Thus, if $\mathcal{A}$ succeeds, then with non-negligible probability either $\beta = z$, or $(\beta \neq z) \wedge (w \neq w')$, which in turn implies that $\mathcal{B}$ breaks SDH, contradicting our assumption.

□

Lemma 3.9. If the bilinear group sampler SampleGrp satisfies the SDH assumption against algebraic adversaries, nhPC$_s$ and phPC$_s$ achieve extractability against algebraic adversaries.

From the previous lemma, if a scheme satisfies the SDH assumption, it also achieves evaluation binding. Now, an algebraic adversary can only succeed with non-negligible probability against extractability only if he can produce an accepting evaluation that doesn't correspond to the evaluation of the given polynomials. We show that the latter cannot occur except if the evaluation binding property fails.

Proof. Fix any efficient, algebraic adversary $\mathcal{A}_{alg}$ and maximum degree bound $D \in \mathbb{N}$. We construct an efficient extractor $\mathcal{E}_{\mathcal{A}_{alg}}$ for the polynomial commitment scheme that succeeds with overwhelming probabiiity. In each round $i \in [r]$ algorithm $\mathcal{E}_{\mathcal{A}_{alg}}$ proceeds as follows. We denote by k the number of group elements output by the adversary $\mathcal{A}_{alg}$.

$\mathcal{E}_{\mathcal{A}_{alg}}(\text{ck}, \text{rk}, [\rho_j]_{j=1}^{i}$

1 : Parse ck as $(\langle \text{group} \rangle, \Sigma)$.

2 : Parse Σ as $\begin{pmatrix} [1]_1 & [\beta]_1 & [\beta^2]_1 & \dots & [\beta^D]_1 \\ [\gamma]_1 & [\gamma\beta]_1 & [\gamma\beta^2]_1 & \dots & [\gamma\beta^D]_1 \end{pmatrix}$

3 : Set $\Sigma_1 := ([1]_1, [\beta]_1, [\beta^2]_1, \dots, [\beta^D]_1)$

4 : Set $\Sigma_2 := ([\gamma]_1, [\gamma\beta]_1, [\gamma\beta^2]_1, \dots, [\gamma\beta^D]_1)$

5 : Invoke the adversary:$[\langle \mathbf{a_j}, \Sigma_1 \rangle + \langle \mathbf{b_j}, \Sigma_2 \rangle_{j=1}^{k}] \leftarrow \mathcal{A}_{alg}(\text{ck}, \text{rk}; [\rho_j]_{j=1}^{i})$

6 : Set $\mathbf{X} := (1, X, \dots, X^D)$

7 : For each j in [k], define polynomials $p_j(X) := \langle \mathbf{a_j}, \mathbf{X} \rangle \in \mathbb{F}_q[X]$ and $\bar{p_j} := \langle b_j, X \rangle \in \mathbb{F}_q[X]$

8 : For each j in [k], let the randomness ω_j be the coefficients of $\bar{p_j}$

9 : Output the polynomials $\mathbf{p} = [p_j]_{j=1}^{k}$ and randomness $\omega := [\omega_j]_{j=1}^{k}$

For a given public-coin challenger $\mathcal{C}$, efficient adversary $\mathcal{B} := (\mathcal{B}_1, \mathcal{B}_2)$, efficient query sampler $\mathcal{Q}$, and round bound $r \in \mathbb{N}$, the extractor $\mathcal{E}_{\mathcal{A}_{alg}}$ can fail with non-negligible probability only if there exists a polynomial whose claimed evaluation is incorrect. We will show that if nhPC$_s$ and phPC$_s$ satisfy evaluation binding, then all evaluations a correct with overwhelming probability.
Assume that $\mathcal{E}_{\mathcal{A}_{alg}}$ outputs polynomials that do not match the claimed evaluations with non-negligible probability $\mu(\lambda)$, then we can use $(\mathcal{A}_{alg}, \mathcal{B}_1, \mathcal{B}_2)$, the public-coin challenger $\mathcal{C}$ and the query sampler $\mathcal{Q}$ to construct the following adversary $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ that succeeds in breaking evaluation binding with the same non-negligible probability $\mu(\lambda)$.

$\mathcal{A}'_1(\mathsf{ck}, \mathsf{rk})$:

For $i = 1, \ldots, r$:

(a) Obtain challenge: $\rho_i \leftarrow \mathcal{C}(\mathsf{ck}, \mathsf{rk}, i)$

(b) Obtain commitments: $\mathbf{c}_i \leftarrow \mathcal{A}_{alg}((\mathsf{ck}, \mathsf{rk}, [\rho_j]_{j=1}^{i})$

(c) Extract polynomials and randomness: $(\mathbf{p_i}, \omega_\mathbf{i}) \leftarrow \mathcal{E}_{\mathcal{A}_{alg}}(\mathsf{ck}, \mathsf{rk}, [\rho_j]_{j=1}^{i})$

Sample query set: $\mathrm{Q} \leftarrow \mathcal{Q}(\mathsf{ck}, \mathsf{rk}, [\rho_j]_{j=1}^{i})$

Set $[c_i]_{i=1}^{n} := [\mathbf{c}_i]_{i=1}^{r}, [p_i]_{i=1}^{n} := [\mathbf{p}_i]_{i=1}^{r}, [\omega_i]_{i=1}^{n} := [\omega_i]_{i=1}^{r}$

Parse Q as $T \times \{z\}$ for some $T \subseteq [n]$ and $z \in \mathbb{F}$

Set $\mathbf{c} := [c_i]_{i \in T}, \mathbf{p} := [p_i]_{i \in T}, \omega := [\omega_i]_{i \in T}$

$(\mathbf{c}, \mathbf{st}_\mathcal{B}) \leftarrow \mathcal{B}_1(\mathsf{ck}, \mathsf{rk}, [\rho_j]_{j=1}^{k}, Q)$

Compute alternate evaluations $\mathbf{u}' := \mathbf{p}(z)$

Set $\mathbf{st} = (\mathbf{c}, z, \mathbf{u}, \mathbf{u}', \mathbf{st}_\mathcal{B})$

Output $(\mathbf{c}, z, \mathbf{u}, \mathbf{u}', \mathbf{st})$

$\mathcal{A}'_2(\mathbf{st}, \xi)$:

Parse $\mathbf{st}$ as $(\mathsf{ck}, \mathsf{rk}, \mathbf{p}, z, \omega, \mathbf{st}_\mathcal{B})$

Obtain proof of evaluation: $\pi \leftarrow \mathcal{B}_2(\mathbf{st}_\mathcal{B}, \xi)$

Compute alternate proof: $\pi' \leftarrow \mathrm{PC_s}.\mathrm{Open}(\mathsf{ck}, \mathbf{p}, z, \xi; \omega)$

Output (π, π')

Since the extractor successfully extracts each polynomial and the randomness, and since $\mathrm{PC_s}$ satisfies completeness, $\mathcal{A}'_2$ should be able to produce an alternate valid proof π' that is also accepted by $\mathrm{PC_s}$.Check. Thus, if $\mathcal{A}$ breaks polynomial extractability with non-negligible probability by producing valid proofs for incorrect evaluations, then $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ breaks evaluation binding of $\mathrm{PC_s}$ with non-negligible probability, contradicting the assumption. □

Lemma 3.10 ([Chi+19])**.** $\mathrm{phPC_s}$ is perfectly hiding.

The $\mathrm{phPC_s}$ scheme achieves the hiding property against even unbounded adversaries, because of the masking polynomial $\overline{\mathbf{p}}(X)$. Intuitively, it doesn't matter how many evaluations an adversary gets for the polynomials $\mathbf{p}(X), \overline{\mathbf{p}}(X)$, he cannot deduce an evaluation for an unqueried point, because there are two polynomials contributing to the commitment. The proof of hiding uses trapdoor simulation and argues that since there is a way to produce an accepting proof without having any knowledge of the polynomial, there is no way an adversary can get information about the polynomial from an accepting proof.

Proof. We describe a polynomial-time simulator $\mathcal{S}$ such that, for every maximum degree bound D and efficient adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, the adversary cannot distinguish between the real and ideal world experiments.
We will leverage the fact that by knowing the "trapdoor" the simulator can create the

evaluation proof for arbitary values with respect to the commitment. We build the simulator as follows:

$\mathcal{S}$.Setup$(1^\lambda, D)$:

1 : Run PC$_s$.Setup$(1^\lambda, D)$, get trap $= (\text{ck}, \text{rk}, \beta, \gamma)$

2 : Output $(\text{ck}, \text{rk}, \text{trap})$

$\mathcal{S}$.Commit(trap, k;ω)

1 : Parse ω as $[\omega_i]_{i=1}^k$

2 : For $i = 1, \ldots, k$:

(a) Obtain random polynomials $\bar{p_i}(X)$ from ω_i

(b) Compute $c_i = [\gamma \bar{p_i}(\beta)]_1$

3 : Output $\mathbf{c} = [c_i]_{i=1}^k$

$\mathcal{S}$.Open(trap,$\mathbf{p}$, $\mathbf{u}$,$Q, \xi; \omega)$:

1 : Parse $\mathbf{p} \coloneqq [p_i]_{i=1}^n$, $\mathbf{u} \coloneqq [u_i]_{i=1}^n$, $\omega \coloneqq [\omega_i]_{i=1}^n$

2 : Parse Q as $T \times \{z\}$, for $T \subseteq [n]$, $z \in \mathbb{F}_q$

3 : For $i \in T$:

(a) if $\omega_i \neq \perp$:

(i) Compute $c_i \leftarrow \mathcal{S}$.Commit(trap, 1;$\omega_i$)

(ii) Obtain the random polynomial $\bar{p_i}(X)$ from ω_i

(iii) Set $\bar{u_i} = \bar{p_i}(z) - \dfrac{u_i}{\gamma}$

(b) Else $\omega_i = \perp$

(i) Compute $c_i \leftarrow$ PC$_s$.Commit(ck,p_i; $\perp$)

(ii) Set $\bar{u_i} = 0$

4 : Compute $\bar{u} = \sum_{i=1}^{n} \xi^i \bar{u_i}$, $u = \sum_{i=1}^{n} \xi^i u_i$, $C = \sum_{i=1}^{n} \xi^i c_i$

5 : If $z \neq \beta$:

Compute $w = \dfrac{1}{\beta - z} C - [\dfrac{u + \gamma \bar{u}}{\beta - z}]_1$

6 : Else $z = \beta$:

Set $w = [0]_1$

7 : Output $\pi = (w.\bar{u})$

Associated with each p_i output by $\mathcal{A}$ there is an independently and randomly sampled degree D polynomial $\bar{p_i}$ defined by ω_i. We define a polynomial $\bar{p_i}'$ such that in the real world, $\bar{p_i}' \coloneqq \bar{p_i}$, whereas in the ideal world, if $h_i = 0$ (hence $\omega_i \neq \perp$), then $\bar{p_i}' \coloneqq \frac{p_i(X)}{\gamma}$, and $\bar{p_i}' \coloneqq 0$ otherwise. Then each $\bar{p_i}'$ is independently and randomly distributed if the corresponding polynomial is required to be hiding. It follows that these polynomials are identically distributed in both worlds. Moreover, since $\mathcal{S}$.Setup uses

PC_s.Setup to generate (ck,rk), we see that (ck,rk) is also identically distributed.
We claim that upon fixing (ck,rk) and $\bar{\mathbf{p}}'$, the resulting $\mathbf{c}$ is given by a deterministic function in $\mathbf{p}(\beta)$, and for query point z the corresponding proof π is given by a deterministic function in $(\mathbf{p}(z), z, \xi)$. Since these deterministic functions are parametrized by ck,rk, and $\bar{\mathbf{p}}'$, which we have shown are identically distributed in both worlds, it follows that the outputs of these functions will also be identically distributed, and thus the two worlds are indistinguishable even by unbounded adversaries.
We will show that the following equations hold in both worlds:

$$\begin{aligned} C &= [p(\beta)]_1 + [\gamma \bar{p}'(\beta)]_1 \\ \bar{u} &= \sum_{i=1}^{n} \xi^i \bar{p_i}'(z) \\ w &= \begin{cases} \frac{1}{\beta - z} C - [\frac{u + \gamma \bar{u}}{\beta - z}]_1, \text{ if } z \neq \beta \\ [0]_1, \text{ if } z = \beta \end{cases} \end{aligned}$$

Indistinguishability of commitments.
Real world.

$$c_i = [p_i(\beta)]_1 + [\gamma \bar{p_i}(\beta)]_1 = [p_i(\beta)]_1 + [\gamma \bar{p_i}']_1$$

Ideal world.

$$\bar{p_i}'(X) = \bar{p_i}(X) - \frac{p_i(X)}{\gamma} \Rightarrow \bar{p_i}(X) = \bar{p_i}'(X) + \frac{p_i(X)}{\gamma}$$

So we get the following:

$$c_i = [\gamma \bar{p_i}(\beta)]_1 = [\gamma(\bar{p_i}'(\beta) + \frac{p_i(\beta)}{\gamma})]_1 = [p_i(\beta)]_1 + [\gamma \bar{p_i}']_1$$

Indistinguishability of evaluation proofs.
Real world.

$$\bar{u} = \sum_{i=1}^{n} \xi^i \bar{p_i}(z) = \sum_{i=1}^{n} \xi^i \bar{p_i}'(z)$$

Ideal world.

$$\begin{aligned} \bar{u} &= \sum_{i=1}^{n} \xi^i \bar{u_i} \\ &= \sum_{i=1}^{n} \xi^i (\bar{p_i}(z) - \frac{u_i}{\gamma} \\ &= \sum_{i=1}^{n} \xi^i (\bar{p_i}'(z) + \frac{p_i(z)}{\gamma} - \frac{u_i}{\gamma}) \\ &= \sum_{i=1}^{n} \xi^i \bar{p_i}'(z) \end{aligned}$$

Real world.

$$\begin{aligned} w =& [w(\beta)]_1 + [\gamma \bar{w}(\beta)]_1 \\ &= [\frac{p(\beta) - p(z)}{\beta - z}]_1 + [\frac{\gamma \bar{p}(\beta) - \gamma \bar{p}(z)}{\beta - z}]_1 \\ &= [\frac{p(\beta) + \gamma \bar{p}(\beta)}{\beta - z}]_1 - [\frac{p(z) + \gamma \bar{p}(z)}{\beta - z}]_1 \\ &= \frac{1}{\beta - z} C - [\frac{u + \gamma \bar{u}}{\beta - z}]_1 \end{aligned}$$

Also, note that if $z = \beta$ then $w, \bar{w}$ are not undefined but rather 0.
Ideal World.
w already has the desired form.
We conclude that no adversary can distinguish between the two worlds. □

CHAPTER 4

CONSTRAINT SYSTEMS

Definition 4.1. An *arithmetic circuit* **C** over a field $\mathbb{F}$, set of public inputs $x_1, \ldots, x_n$ and set of witness inputs $w_1, \ldots, w_m$ is a directed acyclic graph as follows. Every node in it with indegree zero is called an *input gate* and is labeled by either a public input x_i, a field element or a witness input. Every other gate is labeled by either $+$ indicating summation of the incoming values or $\times$ indicating multiplication. Gates with no outgoing wires are called output gates. The maximum supported number of input wires of gates is called the *fan-in* of the circuit, while the maximum number of output wires of gates is called the *fan-out*.

Definition 4.2 (Arithmetic Circuit Satisfiability)**.** Let $\mathbb{F}$ be a finite field. The universal arithmetic circuit satisfiability relation $\mathcal{R}_C$ is the set of triples $(C, (\mathbf{x}, \mathbf{y}), \mathbf{w}$, where $C : \mathbb{F}^{l_{in}} \times \mathbb{F}^{l_{wit}} \to \mathbb{F}^{l_{out}}$ is an arithmetic circuit with l_{in} public inputs, l_{wit} private inputs, and l_{out} public outputs, such that $C(\mathbf{x}, \mathbf{w}) = \mathbf{y}$.

Every arithmetic circuit over a field can be transformed into a system of linear algebra constraints, called R1CS (and its variants). The basic idea is the following: By creating a vector that contains $1_{\mathbb{F}}$, the public and private inputs, the public outputs as well as the outputs of the multiplication gates of the circuit, two matrices can be constructed for the specified circuit, that, together with the vector, create a constraint system that whenever it is satisfied for a vector the circuit is also satisfied for the assignment to the input variables and vice versa. These matrices characterize the left and right inputs to multiplication gates. This characterization of arithmetic circuit satisfiability comes in handy, as we can borrow tricks from linear algebra to create efficient proof systems for proving the satisfiability of the circuit. Another reason is that vectors can be naturally encoded as polynomials, giving also the opportunity to pair them with polynomial commitments.

Definition 4.3 (R1CS [Cam+20])**.** Let $\mathbb{F}$ be a finite field and m,l,s $\in \mathbb{N}$ be positive integers. The universal relation $\mathcal{R}_{R1CS}$ is the set of triples

$$(R, x, w) \coloneqq ((\mathbb{F}, m, l, s, \mathbf{F}, \mathbf{G}, \mathbf{O}), \mathbf{x}, \mathbf{w})$$

where $\mathbf{F}, \mathbf{G}, \mathbf{O} \in \mathbb{F}^{m \times m}$, $\max\{||\mathbf{F}||, ||\mathbf{G}||, ||\mathbf{O}||\} \leq s, \mathbf{x} \in \mathbb{F}^{l-1}, \mathbf{w} \in \mathbb{F}^{m-l}$ and for

$\mathbf{c} \coloneqq (1, \mathbf{x}, \mathbf{w})$ it holds

$$(\mathbf{F} \cdot \mathbf{c}) \circ (\mathbf{G} \cdot \mathbf{c}) = \mathbf{O} \cdot \mathbf{c} \tag{4.0.1}$$

Definition 4.4 (R1CS-lite [Cam+20])**.** Let $\mathbb{F}$ be a finite field and m,l,s $\in \mathbb{N}$ be positive integers. The universal relation $\mathcal{R}_{R1CS-lite}$ is the set of triples

$$(R, x, w) \coloneqq ((\mathbb{F}, m, l, s, \mathbf{F}, \mathbf{G}), \mathbf{x}, \mathbf{w})$$

where $\mathbf{F}, \mathbf{G} \in \mathbb{F}^{m \times m}$, $\max\{||\mathbf{F}||, ||\mathbf{G}||\} \leq s$, the first l rows of $\mathbf{G}$ are $(-1, 0, ..., 0) \in \mathbb{F}^{1 \times m}$, $\mathbf{x} \in \mathbb{F}^{l-1}$, $\mathbf{w} \in \mathbb{F}^{m-l}$, and for $\mathbf{c} \coloneqq (1, \mathbf{x}, \mathbf{w})$, it holds

$$(\mathbf{F} \cdot \mathbf{c}) \circ (\mathbf{G} \cdot \mathbf{c}) = \mathbf{c} \tag{4.0.2}$$

We will also use the following equivalent formulation, which we will call $\mathcal{R}'_{R1CS-lite}$ [RZ21]:

$$(R, x, w) \coloneqq ((\mathbb{F}, s, m, l, \mathbf{F}, \mathbf{G}), \mathbf{x}, (\mathbf{a}', \mathbf{b}'))$$

where $\mathbf{F}, \mathbf{G} \in \mathbb{F}^{m \times m}, \mathbf{x} \in \mathbb{F}^{l-1}, \mathbf{a}', \mathbf{b}' \in \mathbb{F}^{m-l}, max\{||\mathbf{F}||, ||\mathbf{G}||\} \leq s$, and for $\mathbf{a} \coloneqq (1, \mathbf{x}, \mathbf{a}'), \mathbf{b} \coloneqq (\mathbf{1}, \mathbf{b}')$ it holds:

$$\begin{pmatrix} \mathbf{I} & \mathbf{0} & -\mathbf{F} \\ \mathbf{0} & \mathbf{I} & -\mathbf{G} \end{pmatrix} \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{a} \circ \mathbf{b} \end{pmatrix} = \mathbf{0} \tag{4.0.3}$$

Definition 4.5 (LongR1CS-lite [Cam+20])**.** Let $\mathbb{F}$ be a finite field and m,l,s $\in \mathbb{N}$ be positive integers. The universal relation $\mathcal{R}_{LongR1CS-lite}$ is the set of triples

$$(R, x, w) \coloneqq ((\mathbb{F}, m, l, s, \mathbf{F}, \mathbf{G}), \mathbf{x}, (\mathbf{a}', \mathbf{b}', \mathbf{c}'))$$

where $\mathbf{F}, \mathbf{G} \in \mathbb{F}^{m \times m}$, $\max\{||\mathbf{F}||, ||\mathbf{G}||\} \leq s, \mathbf{x} \in \mathbb{F}^{l-1}, \mathbf{a}', \mathbf{b}', \mathbf{c}' \in \mathbb{F}^{m-l}$, and for $\mathbf{a} \coloneqq (1, \mathbf{x}, \mathbf{a}'), \mathbf{b} \coloneqq (\mathbf{1}, \mathbf{b}'), \mathbf{c} \coloneqq (1, \mathbf{x}, \mathbf{c}')$, it holds

$$\mathbf{a} \circ \mathbf{b} = \mathbf{c} \quad \wedge \quad \mathbf{a} + \mathbf{F} \cdot \mathbf{c} = 0 \quad \wedge \quad \mathbf{b} + \mathbf{G} \cdot \mathbf{c} = 0 \tag{4.0.4}$$

Lemma 4.6 ([Cam+20])**.** Let R (resp. $\hat{R}$) be a LongR1CS-lite (resp. R1CS-lite) relation with matrices $\{\mathbf{F}, \mathbf{G}\}$. Then for any $\mathbf{x} \in \mathbb{F}^{l-1}$ it holds $\mathbf{x} \in \mathcal{L}(\mathbf{R})$ if and only if $\mathbf{x} \in \mathcal{L}(\hat{\mathbf{R}})$.

Proof. ($\Rightarrow$) Let $(\mathbf{a}', \mathbf{b}', \mathbf{c}')$ be a witness for $\mathbf{x} \in \mathcal{L}(\mathbf{R})$. By definition of LongR1CS-lite it holds that $\mathbf{F} \cdot (\mathbf{a} \circ \mathbf{b}) \circ \mathbf{G} \cdot (\mathbf{a} \circ \mathbf{b}) = \mathbf{a} \circ \mathbf{b}$ for $\mathbf{a} \coloneqq (1, \mathbf{x}, \mathbf{a}')$ and $\mathbf{b} \coloneqq (\mathbf{1}, \mathbf{b})$ and also the first l rows of $\mathbf{G}$ are $(-1, 0, \ldots, 0) \in \mathbb{F}^{1 \times m}$. Then, $\mathbf{w} \coloneqq \mathbf{a}' \circ \mathbf{b}'$ is a witness for $\mathbf{x} \in \mathcal{L}(\mathbf{R})$, as $(1, \mathbf{x}, \mathbf{w}) = \mathbf{a} \circ \mathbf{b}$.

($\Leftarrow$) Let $\hat{\mathbf{c}'}$ be a witness for $\mathbf{x} \in \mathcal{L}(\mathbf{R})$, that is for $\hat{\mathfrak{c}} \coloneqq (1, \mathbf{x}, \hat{\mathbf{c}'})$ it holds $\hat{\mathfrak{c}} = \mathbf{F} \cdot \hat{\mathfrak{c}} \circ \mathbf{G} \cdot \hat{\mathfrak{c}}$.
Let $\mathfrak{a} \coloneqq -\mathbf{F} \cdot \hat{\mathfrak{c}}, \mathfrak{b} \coloneqq -\mathbf{G} \cdot \hat{\mathfrak{c}}$ and $\mathbf{c}' \coloneqq \hat{\mathbf{c}'}$ and let $\mathbf{a}', \mathbf{b}'$ be the last m-l rows of $\tilde{a}, \tilde{b}$ respectively.
By the satisfiability of R1CS-lite we have that

$$\begin{pmatrix} 1 \\ \mathbf{x} \\ \mathbf{c}' \end{pmatrix} = \hat{\mathfrak{c}} = \mathbf{F} \cdot \hat{\mathfrak{c}} \circ \mathbf{G} \cdot \hat{\mathfrak{c}} = \mathfrak{a} \circ \mathfrak{b} = \begin{pmatrix} \mathbf{a}'' \\ \mathbf{a}' \end{pmatrix} \circ \begin{pmatrix} \mathbf{b}'' \\ \mathbf{b}' \end{pmatrix}$$

which implies that $\mathbf{c}' = \mathbf{a}' \circ \mathbf{b}'$ and thus for $\mathbf{a} \coloneqq (1, \mathbf{x}, \mathbf{a}'), \mathbf{b} \coloneqq (\mathbf{1}, \mathbf{b}')$, and $\mathbf{c} \coloneqq (1, \mathbf{x}, \mathbf{c}')$, the Hadamard product constraint of R1CS-lite must hold.
Finally, from the definition of the first l rows of $\mathbf{G}$ it holds that $\mathfrak{b} = (\mathbf{1}, \mathbf{b}')$ and thus $\mathfrak{a} = (1, \mathbf{x}, \mathbf{a}')$. Therefore, for $\mathbf{a}, \mathbf{b}$ as above the linear constraints of R1CS-lite are also satisfied.
This concludes the proof that $(\mathbf{a}', \mathbf{b}', \mathbf{c}')$ is a satisfying witness for $\mathbf{x} \in \mathcal{L}(\mathbf{R})$.

□

Theorem 4.7 ([Cam+20])**.** Let $C : \mathbb{F}^{l_{in}} \times \mathbb{F}_{l_{wit}} \to \mathbb{F}^{l_{out}}$ be an arithmetic circuit with N multiplication gates. Then, there exists an $R1CS - lite\{\mathbf{L}, \mathbf{R}\} \in \mathbb{F}^{n \times n}$ with $l = l_{in} + l_{out} + 1, n = l + N'$, where $N' = N + l_{wit}$, such that for any $\mathbf{x}$, $\exists \mathbf{w} : C(\mathbf{x}, \mathbf{w}) = \mathbf{y}$ only if $\exists \mathbf{w}'$ that makes $(1, \mathbf{x}, \mathbf{y})$ accepted by $\{\mathbf{L}, \mathbf{R}\}$.

Proof. We do the proof by building matrices for the LongR1CS-lite relation. By the equivalence of R1CS-lite and LongR1CS-lite shown in the previous lemma, this shows a reduction to R1CS-lite. Note that in the definition, we regard the private inputs of the non-deterministic circuit as multiplication gates.
Our goal is to define $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and matrices $\mathbf{L}, \mathbf{R}$ such that the satisfiability of C can be expressed as follows:

$$\begin{cases} \mathbf{a} + \mathbf{L} \cdot \mathbf{c} = \mathbf{0} \\ \mathbf{b} + \mathbf{R} \cdot \mathbf{c} = \mathbf{0} \\ \mathbf{a} \circ \mathbf{b} = \mathbf{c} \end{cases} \tag{4.0.5}$$

□

with $\mathbf{a} = (1, \mathbf{x}, \mathbf{y}, \mathbf{a}'), \mathbf{b} = (\mathbf{1}, \mathbf{b}')$ and $\mathbf{c} = (1, \mathbf{x}, \mathbf{y}, \mathbf{c}')$.
Partition [n] into $I_{in} = \{2, \ldots, l_{in}+1\}, I_{out} = \{l_{in}+2, \ldots, l\}, I_{mid} = \{l+1, \ldots, n\}$.
Label all the multiplication gates of C with integers in I_{mid}, and for every such multiplication gate j, denote by a_j, b_j, c_j its left input, right input,output respectively. Then the consistency of every multiplication gate can be checked as:

$$\forall j \in I_{mid} : \begin{cases} a_j + \mathbf{L}_j \cdot \mathbf{c} = 0 \\ b_j + \mathbf{R}_j \cdot \mathbf{c} = 0 \\ a_j \cdot b_j - c_j = 0 \end{cases}$$

for approprite row vectors $\mathbf{L}_j, \mathbf{R}_j$ which express the linear subcircuits for the left and right input wires. An arithmetic circuit is made of fan-in 2 sum gates, fan-in 2 multiplication gates and multiplication by a constant gates. However, we can equivalently batch together the addition gates and multiplication by constant gates and consider circuits of fan-in 2 multiplication gates, whose inputs a linear combination of the outputs of the already computed multiplication gates and input gates. More in detail, a multiplication gate can be described by a list of coefficients $l_0, \ldots, l_k$ (the left inputs) and a list of coefficients $r_0, \ldots, r_{k'}$ (the right inputs) for values $k, k' \in \mathbb{N}$ and computes the function that maps $(x_1, \ldots, x_k, y_1, \ldots, y_{k'})$ (outputs of multiplication gates and input gates) to $(\sum_{i=1}^{k} l_i x_i + l_0) \cdot (\sum_{i=1}^{k'} r_i y_i + r_0)$. Moreover, in the circuit C the inputs of the gates with label j are connected to the outputs of $k + k'$ distinct gates with indexes $j_1, \ldots, j_{k+k'}$. The row vectors of of $\mathbf{L}_j$ and $\mathbf{R}_j$ are thus defined as:

$$\mathbf{L}_{j,i} \coloneqq \begin{cases} -l_0 & \text{if } i = 1 \\ -l_a & \text{if } \exists a : i = j_a \\ 0 & \text{else} \end{cases} \qquad \mathbf{R}_{j,i} \coloneqq \begin{cases} -r_0 & \text{if } i = 1 \\ -r_a & \text{if } \exists a : i = j_{k+a} \\ 0 & \text{else} \end{cases}$$

Next, we add constraints for the public outputs:

$$\forall j \in I_{out} : \begin{cases} a_j + \mathbf{L}_j \cdot \mathbf{c} = 0 \\ b_j - c_1 = 0 \\ a_j \cdot b_j - c_j = 0 \end{cases}$$

The first and second constraints check correctness of outputs obtained from possible linear subcircuits on multiplication gates outputs, namely gates of the form $\sum_i l_i x_i$ for constraints $l_1, \ldots, l_k$ and input variables $x_1, \ldots, x_k$. The row vectors $\mathbf{L}_j$ for $j \in I_{out}$ are thus defined as:

$$\mathbf{L}_{j,i} \coloneqq \begin{cases} -l_a & \text{if } \exists a : i = j_a \\ 0 & \text{else} \end{cases}$$

Finally, recalling that $\mathbf{a} = (1, \mathbf{x}, \mathbf{y}, \mathbf{a}'), \mathbf{b} = (\mathbf{1}, \mathbf{b}')$ and $\mathbf{c} = (1, \mathbf{x}, \mathbf{y}, \mathbf{c}')$ we can add the following (dummy) constraints for the public inputs:

$$\forall j \in \{1\} \cup I_{in} : \begin{cases} a_j - c_j = 0 \\ b_j - c_1 = 0 \\ a_j \cdot b_j - c_j = 0 \end{cases}$$

We conclude by showing how to define matrices $\mathbf{L}, \mathbf{R}$ such that all the constraints above are compactly represented by the equation.

$$\mathbf{L} = \begin{pmatrix} -\mathbf{I}_{l_{in}+1} & | & \mathbf{0} \\ \mathbf{L}_{l_{in}+2} & & \\ \vdots & & \\ \mathbf{L}_n & & \end{pmatrix}$$

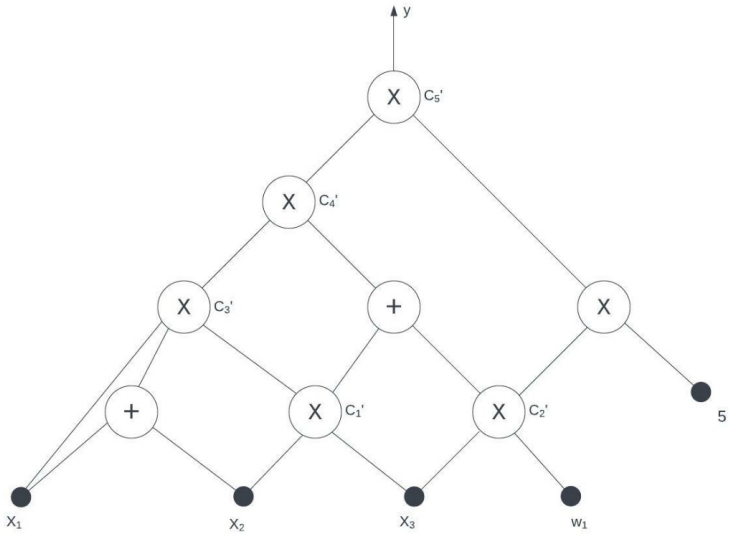
To define matrix **R** we use an auxiliary $l \times n$ matrix **E** where each row is the unit vector $\mathbf{e}_1 \in \mathbb{F}_n$.

$$\mathbf{E} = \begin{pmatrix} 1 & \dots & 0 & \dots & 0 \\ & & \vdots & & \\ 1 & \dots & 0 & \dots & 0 \end{pmatrix}, \quad \mathbf{R} = \begin{pmatrix} -\mathbf{E} \\ \mathbf{R}_{l+1} \\ \vdots \\ \mathbf{R}_n \end{pmatrix}$$

Example. We will now give an example of converting a computation to LongR1CS-lite. Assume we wish to perform the following computation:

$$5x_1x_2w_1(x_1 + x_2)(x_2x_3 + x_3w_1)$$

For some $x_1, x_2.x_3.w_1 \in \mathbb{F}$. Then, regarding x_1, x_2, x_3 as the public inputs, w_1 as the private inputs and as y the public output of the computation, this amounts to computing the circuit below.



The total amount of multiplication (by non-constant) gates are 5. Denoting $a'_1, a'_2, a'_3, a'_4, a'_5$, the left inputs to multiplication gates, $b'_1, b'_2, b'_3, b'_4, b'_5$, we need to define vectors $\mathbf{a} = (1, x_1, x_2, x_3, y, w_1, a'_1, a'_2, a'_3, a'_4, a'_5)$,

$\mathbf{b} = (1, 1, 1, 1, 1, 1, b'_1, b'_2, b'_3, b'_4, b'_5)$, $\mathbf{c} = (1, x_1, x_2, x_3, y, w_1, c'_1, c'_2, c'_3, c'_4, c'_5)$ such that the constraints of LongR1CS-lite hold.
To do so observe that:

$$\begin{array}{ll} a'_1 = x_2 & b'_1 = x_3 \\ a'_2 = x_3 & b'_2 = w_1 \\ a'_3 = x_1 & b'_3 = x_1 + x_2 \\ a'_4 = c'_3 & b'_4 = c'_1 + c'_2 \\ a'_5 = c'_4 & b'_5 = 5c'_2 \end{array}$$

Now we can can use these values to compute the $\mathbf{L}.\mathbf{R}$ matrices of the relation. Specifically we have the following matrices:

$$\mathbf{L} = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{pmatrix}$$

$$\mathbf{R} = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -5 & 0 & 0 & 0 \end{pmatrix}$$

CHAPTER 5

POLYNOMIAL HOLOGRAPHIC PROOFS

Polynomial Holographic Proofs (PHP) [Cam+20] constitute the information theoretic part of the construction. They are a generalization of Algebraic Holographic Proofs (AHP) introduced in [Chi+19]. A proof system for a relation 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 the offline phase. This is a crucial step to achieve succinctness, because by merely reading the description of the circuit the verifier's work becomes linear. A PHP consists of an interaction between a verifier and a prover sending oracle polynomials (which the verifier does not *see*), followed by a decision phase in which the verifier outputs a set of polynomial identities to be checked on the prover's and indexer's polynomials (such as $a(X)b(X) - z \cdot c(X) = 0$) for orancle polynomials a, b, c and some scalar z), as well as a set of degree tests (e.g. $\deg(a(X) < D$).

Definition 5.1. (Polynomial Holographic IOPs (PHP)). A polynomial Holographic IOP for a family of field-dependent relations $\mathcal{R}$ is a tuple PHP $= (\mathrm{rnd}, \mathrm{n}, \mathrm{m}, \mathrm{d}, \mathrm{n_e}, \mathcal{I}, \mathcal{P}, \mathcal{V})$, where $\mathrm{rnd}, \mathrm{n}, \mathrm{m}, \mathrm{d}, \mathrm{n_e} : \{0,1\}^* \to \mathbb{N}$ 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 $\mathcal{I}(\mathrm{R})$ is executed on a relation description R, and it returns $\mathrm{n}(0)$ polynomials $\{p_{0,j}\})j = 1^{\mathrm{n}(0)} \in \mathbb{F}[X]$ encoding the relation R and where $\mathbb{F}$ is a field specified by R.

- **Online phase:** The prover $\mathcal{P}(\mathrm{R}, \mathrm{x}, \mathrm{w})$ and the verifier $\mathcal{V}^{\mathcal{I}(\mathrm{R})}(x)$ are executed for $\mathrm{rnd}(|\mathrm{R}|)$ rounds, the prover has a tuple $(\mathrm{R}, \mathrm{x}, \mathrm{w}) \in \mathcal{R}$, 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 $\mathrm{m}(i)$ messages $\{\pi_{i,j} \in \mathbb{F}\}_{j=1}^{\mathrm{m}(i)}$, and $\mathrm{n}(i)$ oracle polynomials $\{p_{i,j} \in \mathbb{F}[X]\}_{j=1}^{\setminus(i)}$, such that $\deg(p_{i,j} < \mathrm{d}(|\mathrm{R}|, i, j)$.

- **Decision phase:** After the $\mathrm{rnd}(|\mathrm{R}|)$-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 $\mathrm{n}_p = \sum_{k=1}^{\mathrm{rnd}(|\mathrm{R}|)} \mathrm{n}(k)$ and let $p_1, \ldots, p_{n_p}$ be the polynomials sent by $\mathcal{P}$. The verifier specifies a vector of integers $\mathbf{d} \in \mathbb{N}^{n_p}$, which satisfies the following condition

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

– Polynomial checks: to verify that certain polynomial identities hold between the oracle polynomials and the message sent by the prover. Let $\mathrm{n}^* = \sum_{k=0}^{\mathrm{rnd}(|\mathrm{R}|)} \mathrm{n}(k)$ and $\mathrm{m}^* = \sum_{k=0}^{\mathrm{rnd}(|\mathrm{R}|)} \mathrm{m}(k)$, and denote by $(p_1, \ldots, p_{n^*})$ and $(\pi_1, \ldots, \pi_{n^*})$ all the oracle polynomials (including the ones send by the encoder) and all the messages sent by the prover. The verifier can specify a list of $\mathrm{n_e}$ tuples, each of the form $(G, u_1, \ldots, u_{n^*})$, where $G \in \mathbb{F}[X, X_1, \ldots, X_{n^*}, Y_1, \ldots, Y_{m^*}]$ and every $u_k \in \mathbb{F}[X]$. Then a tuple $(G, u_1, \ldots, u_{n^*})$ is satisfied if and only $F(X) \equiv 0$ where

$$F(X) \coloneqq G(X, \{p_k(u_k(X))\}_{k=1,\ldots,\mathrm{n}^*}, \{\pi_k\}_{k=1,\ldots,\mathrm{n}^*})$$

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

Definition 5.2. A PHP is complete if for any triple $(\mathrm{R}, \mathrm{x}, \mathrm{w}) \in \mathcal{R}$, the checks returned by $\mathcal{V}^{\mathcal{I}(\mathrm{R}}$ after interacting with the honest prover $\mathcal{P}(\mathrm{R}, \mathrm{x}, \mathrm{w})$, are satisfied with probability 1.

Definition 5.3. A PHP is ϵ-sound if for every relation-instance tuple $(\mathrm{R}, \mathrm{x}) \notin \mathcal{L}(\mathcal{R})$ and polynomial time prover $\mathcal{P}^*$ we have

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

Definition 5.4. 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

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

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.

Definition 5.5. A PHP is ϵ-zero-knowledge if there exists a PPT simulator $\mathcal{S}$ such that for every triple $(\mathrm{R}, \mathrm{x}, \mathrm{w}) \in \mathcal{R}$, and every algorithm $\mathcal{V}^*$, the following random variables are within ϵ-statistical distance:

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

where $\mathrm{View}(\mathcal{P}((\mathrm{R}, \mathrm{x}, \mathrm{w}), \mathcal{V}^*)$ consists of $\mathcal{V}^*$'s randomness, $\mathcal{P}$'s messages (which do not include the oracles) and $\mathcal{V}^*$'s list of checks, while $\mathrm{View}(\mathcal{S}^{\mathcal{V}^*}(\mathrm{R}, \mathrm{x}))$ 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.

The following definition captures the fact that zero-knowledge should hold even when the verifier has a access to a *bounded amount* of evaluations of the polynomials that contain information about the witness. This fine-grained characterization, formalized in Lunar [Cam+20], allow for a minimization of the requirements needed to achieve zero-knowledge. A straightforward compiler from PHPs to zkSNARKs would require

hiding polynomials; this requirement can be relaxed by leveraging the fact that the verifier does not need to see "many" evaluations of the oracle polynomials, but rather a *bounded* number of them to be convinced.

Let $\mathcal{Q}$ be a list of queries; we say that $\mathcal{Q}$ is $(\mathbf{b}, C)$-bounded for $\mathbf{b} \in \mathbb{N}^{n_p}$ and for C a PT algorithm, if for every $i \in [n_p], |\{(i,z) \in \mathcal{Q}\}| \leq b_i$, and for all $(i,z) \in \mathcal{Q}, C(i,z) = 1$.

Definition 5.6. A PHP is (b, C)-zero-knowledge if for every triple $(\mathrm{R}, \mathrm{x}, \mathrm{w}) \in \mathcal{R}$, and every (b, C)-bounded list $\mathcal{Q}$, the following random variables are within ϵ statistical distance:

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

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

Definition 5.7. 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 $i \in \mathbb{N}$, $\Pr[\mathrm{C}(i,z) = 0]$ is negligible, where z is uniformly sampled over $\mathbb{F}$.

5.1 Checkable Subspace Sampling

In a *Checkable Subspace Sampling* (CSS) argument, the prover and verifier interactively agree on a polynomial $D(X)$ representing a vector **d** in the row space of a matrix **M**. The core of the protocol is that $D(X)$ is calculated as a linear combination of encodings of the rows of **M** with coefficients determined by the verifier, but he does not need to calculate the polynomial himself (this would result in a non-succinct argument). Instead, the prover can calculate $D(X)$ and then convince the verifier that it was computed correctly, according to the verifiers' coins.

Essentially a CSS scheme is similar to a PHP for the relation $\mathrm{R_M}$, except that the statement $(cns, D(X))$ is decided interactively and the verifier has only oracle access to the polynomial $D(X)$. A CSS can be used as a building block in a PHP and the result will also be a PHP.

Definition 5.8. (Checkable Subspace Sampling, CSS). A checkable subspace sampling argument over a field $\mathbb{F}$ defines some $\mathrm{Q}, \mathrm{m} \in \mathbb{N}$, 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 :$\mathcal{C} \leftarrow \mathbb{F}^Q$, and a relation:

$$\mathrm{R_{CSS,\mathbb{F}}} = \left\{ (\mathrm{M}, \mathrm{cns}, D(X)) : \begin{array}{r} \mathbf{M} \in \mathcal{M} \subset \mathbb{F}^{Q\times m}, D(X) \in \mathbb{F}[X], \mathrm{cns} \in \mathcal{C} \\ s = \mathrm{Smp}(\mathrm{cns}), \text{ and } D(X) = \mathbf{s}^\intercal \mathbf{M} \beta(X) \end{array} \right\}$$

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

$$R_{\mathbf{M}} = \left\{ (\mathrm{cns}, D(X)) \quad : \quad (\mathbf{M}, \mathrm{cns}, D(X)) \in R_{\mathrm{CSS},\mathbb{F}} \right\}$$

It consists of three algorithms:

- $\mathcal{I}_{CSS}$ is the indexer: in an offline phase, on input $(\mathbb{F}, \mathbf{M})$ returns a set $\mathcal{W}_{CSS}$ of $\mathrm{n}(0)$ polynomials $\{p_{0,j}(X)\}_{j=1}^{\mathrm{n}(0)} \in \mathbb{F}[X]$. The algorithm is run once for each **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:** $\mathcal{P}_{CSS}$ and $\mathcal{V}_{CSS}$ engage in an interactive protocol. In some round, the verifier sends cns $\leftarrow \mathcal{C}$, and the prover replies with $D(X) = \mathbf{s}^{\intercal}\mathbf{M}\beta(X)$, for $\mathbf{s} = \text{Smp}(\text{cns})$.
  - **ProveSampling:** $\mathcal{P}_{CSS}$ and $\mathcal{V}_{CSS}$ engage in another interactive protocol to prove that $(\text{cns}, D(X)) \in \text{R}_{\mathbf{M}}$.
- When the proving phase is concluded, the verifier outputs a bit indicating acceptance or rejection.

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:

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

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

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.

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

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

For a CSS argument to be useful, we additionally need that the distribution induced by the sampling function is sufficiently "good". This geometric property is captured by the Elusive Kernel property.

Definition 5.9. A CSS argument is ϵ-elusive kernel if

$$\forall \mathbf{t} \in \mathbb{F}^Q, \mathbf{t} \neq \mathbf{0}: \quad \Pr[\mathbf{s} \cdot \mathbf{t} = 0 \mid \mathbf{s} = \text{Smp}(\text{cns}); \text{cns} \leftarrow \mathcal{C}] \leq \epsilon$$

CHAPTER 6

USING LAGRANGE POLYNOMIALS TO PROVE HADAMARD PRODUCT AND INNER PRODUCT RELATIONS

We now present how, by encoding vectors using the Lagrange basis, one can prove Hadamard and Inner Product relations between the vectors.

For any $P_1(X) \in \mathbb{F}[X]$, we see that $P_1(X) \equiv P_2(X)(mod(t(X)))$, where $P_2(X) \coloneqq \sum_{i=1}^{m} P_1(h_1) \cdot \lambda_i(X)$. Evaluating P_2 at a point $u \in \mathbb{F} \backslash \mathbb{H}$ we get $P_2(u) = \sum_{i=1}^{m} P_1(h_1) \cdot \lambda_i(u)$, that is, the sum of the evaluations of P_i at $\{h_i\}_{i=1}^{m}$ scaled by $\{\lambda_i(u)\}_{i=1}^{m}$. The idea is, we can multiply by a normalizing polynomial to get rid of these scalars.

Theorem 6.1 (Generalized Sumcheck.). Let $\mathbb{H}$ be an arbitrary subset of some finite field $\mathbb{F}$ and $t(X)$ the vanishing polynomial at $\mathbb{H}$. For any $P(X) \in \mathbb{F}[X], S \subset \mathbb{H}$, and any $u \in \mathbb{F}, u \notin \mathbb{H}, \sum_{s \in S} P(s) = \sigma$ if and only if there exist polynomials $H(X) \in \mathbb{F}[X], R(X) \in \mathbb{F}_{\leq m-2}[X]$ such that

$$P(X)N_{S,u}(X) - \sigma = (X - u)R(X) + t(X)H(X)$$

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

Proof. From the previous discussion we have that

$$\begin{aligned} P(X)N_{S,u}(X) - \sigma = \\ (\sum_{h \in \mathbb{H}} P(h)\lambda_h(X))(\sum_{s \in S} \lambda_s(u)^{-1}\lambda_s(X)) - \sigma \equiv \\ (\sum_{s \in S} P(s)\lambda_s(u)^{-1}\lambda_s(X)) - \sigma(mod(t(X)) \end{aligned}$$

Setting $Q(X) = (\sum_{s \in S} P(s)\lambda_s(u)^{-1}\lambda_s(X)) - \sigma$, we see that $Q(X)$ has degree at most $m - 1$, it is congruent to $P(X)N_{S,u}(X) - \sigma$ and that $Q(u) = \sum_{s \in S} P(s) - \sigma$. Thus, $\sum_{s \in S} P(s) = \sigma$ if and only if $Q(X)$ is divisible by $X - u$ and the theorem follows. □

Lemma 6.2. If $S = \mathbb{H}$ is a multiplicative subgroup of $\mathbb{F}$, $N_{\mathbb{H},0}(X) = m$

Proof. $N_{\mathbb{H},0}(X) = \sum_{i=1}^{m} \lambda_i(0)^{-1}\lambda_i(X) = m\sum_{i=1}^{m} \lambda_i(X) = m$ □

From the previous lemma and theorem we get the univariate sumcheck.

Theorem 6.3 (Univariate Sumcheck). If $\mathbb{H}$ is a multiplicative subgroup of $\mathbb{F}$, $\sum_{h\in\mathbb{H}} P(h) = \sigma$ if and only if there exist polynomials $R(X), H(X)$ with $\eth(R(X)) \leq m-2$ such that

$$P(X)m - \sigma = XR(X) + t(X)H(X) \tag{6.0.1}$$

Hadamard product relation. From the previous discussion, if we encode **a,b,c** as $A(X) = \mathbf{a}^\intercal\lambda(X), B(X) = \mathbf{b}^\intercal\lambda(X), C(X) = \mathbf{c}^\intercal\lambda(X)$, then it holds that $\mathbf{a} \circ \mathbf{b} = \mathbf{c}$ if and only if $A(X)B(X) - C(X) = H(X)t(X)$ for some $H(X) \in \mathbb{F}[X]$.

Theorem 6.4 (Inner Product Polynomial Relation.). For some $k \in \mathbb{N}$, let $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_k), \mathbf{y}_i = (y_{ij}), \mathbf{d} = (\mathbf{d}_1, \ldots, \mathbf{d}_k)$ be two vectors in $\mathbb{F}^{km}, \mathbf{y}_i, \mathbf{d}_i \in \mathbb{F}^m$, and $\mathbb{H}$ a multiplicative subgroup 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:

$$\mathbf{Y}(X) \cdot \mathbf{D}(X) - \frac{\sigma}{m} = XR(X) + t(X)H(X) \tag{6.0.2}$$

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

Proof. Since $Y_i(h_j) = y_{ij}$, for all $i, j, Y_i(X) = \mathbf{y}_i^\intercal\lambda(X)(mod(t(X)))$. Thus, from the aforementioned properties of the Langrange basis, $Y_i(X)D_i(X) \equiv (\mathbf{y}_i \circ \mathbf{d}_i)^\intercal\lambda(X)(mod(t(X)))$. Hence,

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

By the previous theorem, $((\sum_{i=1}^{k}(\mathbf{y}_i \circ \mathbf{d}_i)^\intercal)\lambda(X))m - \sigma$ 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 j-th 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. □

Corollary 6.5. Let $k, m, \mathbf{y}, \mathbf{d}, \mathbb{F}, \mathbb{H}$ be as in the inner product theorem 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:

$$\mathbf{Y}(X) \cdot \mathbf{D}(X)(X-u) - \frac{\sigma}{m}(X-u) = XR(X) + t(X)H(X)(X-u) \tag{6.0.3}$$

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

CHAPTER 7

PHP FOR R1CS-LITE' FROM SIMPLER BLOCKS

7.1 From CSS to Linear Argument

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

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

using a CSS scheme as a building block. That is given a vector **y**, the argument allows to prove membership in the linear space $\mathbf{W}^{\perp} = \{\mathbf{y} \in \mathbb{F}^{km} : \mathbf{W}\mathbf{y} = \mathbf{0}\}$. Afterwards, we build a PHP for R1CS, using the PHP for Linear Argument as a black-box and forcing the polynomials outputted to satisfy the Hadamard product relation.

To prove $\mathbf{y} \in \mathbf{W}^{\perp}$ a standard way is to let the verifier sample a *sufficiently random* vector **d** in the row space of **W** and then prove $\mathbf{y} \cdot \mathbf{d} = 0$. The trick to make this succinct is to let **d** be computed by the prover according to the verifier's coins, after **y** is declared, through a CSS argument.

The argument goes as follows. The prover sends a vector of polynomials $\mathbf{Y}(X)$ encoding **y**. The CSS argument is used to delegate to the prover the sampling of $\mathbf{d}_i^{\intercal}, i = 1, \ldots, k$ in the row space of $\mathbf{W}_i$. Then the prover sends $\mathbf{D}(X)$ together with a proof that $\mathbf{y} \cdot \mathbf{d} = 0$. Because of the soundness property of the CSS argument, the prover cannot influence the distribution of **d**, which is sampled according to the verifier's coins. Therefore, if $\mathbf{Y}(X)$ passes the test of the verifier, **y** is orthogonal to **d**. By the Elusive Kernel property of the CSS argument, **d** will be sufficiently random. As it is sampled after **y** is declared, this will imply that $\mathbf{y} \in \mathbf{W}^{\perp}$.

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

Online Phase: $\mathcal{P}_{LA}$: On input a witness $\mathbf{y} \in W_Y \subset (\mathbb{F}^m)^k$, output $\mathbf{Y}(X)$, such that $Y_i(h_j) = y_{ij}$

$\mathcal{P}_{LA}$ and $\mathcal{V}_{LA}$ run in parallel k instances of CSS argument, with inputs $(\mathbb{F}, \mathbf{W}_i)$ and $\mathbb{F}$, respectively, and where the verifier is given oracle access to $\mathcal{W}_{CSS_i}$. The output is a set $\{(\text{cns}, D_i(X))\}_{i=1}^{k}$, where cns are the same for all k instances. Define $\mathbf{D}(X) = (D_1(X), \ldots, D_k(X))$.

$\mathcal{P}_{LA}$: Outputs $R_t(X) \in \mathbb{F}_{\leq m-2}[X], H_t(X)$ such that

$$\mathbf{Y}(X) \cdot \mathbf{D}(X) = XR_t(X) + t(X)H_t(X) \tag{7.1.2}$$

Decision Phase: Accept if and only if (1) $\deg(R_t) \leq m-2$, (2) $\mathcal{V}_{CSS}^{i}$ accepts (cns,$D_i(X)$), and (3) the following equation holds: $\mathbf{Y}(X) \cdot \mathbf{D}(X) = XR_t(X) + t(X)H_t(X)$

Theorem 7.1. When instantiated using a CSS scheme with perfect completeness, the previous PHP for membership in $\mathbf{W}^{\perp}$ has perfect completeness.

Proof. By definition, $\mathbf{D}(X) = (\mathbf{s}^{\intercal}\mathbf{W_1}\lambda(X), \ldots, \mathbf{s}^{\intercal}\mathbf{W_k}\lambda(X))$, for $\mathbf{s} = \text{Smp(cns)}$. Thus, $\mathbf{D}(X)$ is the polynomial encoding of $\mathbf{d} = (\mathbf{s}^{\intercal}\mathbf{W_1}, \ldots, \mathbf{s}^{\intercal}\mathbf{W_k}) = \mathbf{s}^{\intercal}\mathbf{W}$. Therefore, for $\mathbf{y} \in \mathbf{W}^{\perp}$, we get $\mathbf{d} \cdot \mathbf{y} = \mathbf{s}^{\intercal}\mathbf{W}\mathbf{y} = \mathbf{0}$. By the aforementioned characterization of the inner product, this implies the existence of polynomials $H_t(X), R_t(X)$ satisfying the verification equation. □

Theorem 7.2. Let CSS be $\epsilon-$sound and ϵ'-Elusive Kernel. Then for any polynomial time adversary $\mathcal{A}$ against the soundness of the PHP:

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

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 $\mathcal{P}_{LA}^*$ and $\mathbf{y}^* = (\mathbf{y_1^*}, \ldots, \mathbf{y_k^*})$ the vector such that $Y_i^*(h_j) = y_{ij}^*$. As a direct consequence of the previous theorems, $\mathbf{Y}^*(X) \cdot \mathbf{D}(X) = XR_t(X) + t(X)H_t(X)$ only if $\mathbf{y}^* \cdot \mathbf{d} = 0$, where $\mathbf{d}$ is the unique vector such that $\mathbf{D}(X) = (\mathbf{d_1^{\intercal}}\lambda(X), \ldots, \mathbf{d_k^{\intercal}}\lambda(X))$.
Now, 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 $\mathbf{y}^*\mathbf{d} = 0$ 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 $\mathbf{y}^*\mathbf{d} = 0$. Since $\mathbf{d}^{\intercal} = \mathbf{s}^{\intercal}\mathbf{W}$, and in a successful attack $\mathbf{y}^* \notin \mathbf{W}^{\perp}$, we can bound this possibility by the probability:

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

Since $\mathbf{s}^{\intercal}\mathbf{W}\mathbf{y}^* = \mathbf{s} \cdot (\mathbf{W}\mathbf{y}^*)$, 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 $\{h_j\}_{j=1}^{m}$ 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. □

7.2 From Linear Argument to R1CS-lite'

We now present a PHP for R1CS-lite', using a PHP for Linear Argument as a building block and assuming that the outputted polynomials satisfy the Hadamard product relation.

Offline Phase: $\mathcal{I}_{lite}(\mathbf{W}, \mathbb{F})$ runs $\mathcal{I}_{LA}(\mathbf{W}, \mathbb{F})$ to obtain a list of polynomials $\mathcal{W}_{LA}$ and outputs $\mathcal{W}_{lite} = \mathcal{W}_{LA}$

Online Phase: $\mathcal{P}_{lite}(\mathbb{F}, \mathbf{W}, \mathbf{x}, (\mathbf{a}', \mathbf{b}'))$ defines $\mathbf{a} = (1, \mathbf{x}, \mathbf{a}'), \mathbf{b} = (\mathbf{1_l}, \mathbf{b}')$ and computes

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

for $t_l(X) = \prod_{i=1}^{l}(X - h_i)$. It outputs $(A'(X), B'(X))$.

$\mathcal{V}_{lite}$ and $\mathcal{P}_{lite}$ instantiate $\mathcal{V}_{LA}^{\mathcal{W}_{LA}}$ and $\mathcal{P}_{LA}(\mathbb{F}, \mathbf{W}, (\mathbf{a}, \mathbf{b}, \mathbf{a} \circ \mathbf{b}))$. Let $\mathbf{Y}(X) = (A(X), B(X), A(X)B(X))$ be the polynomials outputted by $\mathcal{P}_{LA}$ in the first round.

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

Theorem 7.3. When instantiated with a complete, sound and knowledge sound linear argument, the PHP satisfies completeness, soundness and knowledge-soundness.

7.3 Adding Zero-Knowledge

In [RZ21] they show how to add zero-knowledge to the PHP for R1CS-lite' without sending additional masking polynomials, with a minimal increase to the prover's cost. Let $(b_A, b_B, b_{R_t}, b_{H_t})$ be the tuple of bounds on the number of polynomial evaluations seen by the verifier after compiling for the polynomials $A(X), B(X), R_t(X), H_t(X)$. To commit to a vector $\mathbf{y} \in \mathbb{F}^m$, we sample some randomness $\mathbf{r} \in \mathbb{F}^{\mathbf{n}}$, where n is a function of the bounds to be specified (a small constant when compiling). The cardinal og $\mathbb{H}$ is denoted $\tilde{m}$ here. The encoding of the vector $(\mathbf{y}, \mathbf{r})$ is done in the usual way. The main idea is of [RZ21] is to consider related randomness for $A(X), B(X)$ so that the additional randomness sums to 0 and so does not interfere with the inner product argument. Their nover approach is to enforce this behaviour on the prover by adding additional constraint to $\mathbf{W}$.

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

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

Online Phase: $\mathcal{P}_{lite}$ 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 $\mathfrak{a} := (1, \mathbf{x}, \mathbf{a}', \mathbf{r}_a), \mathfrak{b} := (\mathbf{1_l}, \mathbf{b}', \mathbf{r}_b)$, to construct $\tilde{A}(X)$ and $\tilde{B}(X)$, $\tilde{A}'(X)$ and $\tilde{B}'(X)$ as before.

Theorem 7.4. With the previous modification the PHP is perfectly complete, sound, knowledge-sound, perfect zero-knowledge and $(b_A, b_B, b_{R_t}, b_{H_t})$ - bounded honest-verifier zero-knowledge if
$n \geq (b_A + b_B + b_{R_t} + b_{H_t} + 1)/2$, and $n \geq max(b_A, b_B)$.

Proof. For completeness, the additional constraint makes sure that $\sum_{i=1}^{n} r_{a,i} + r_{b,i} = 0$, and an honest prover chooses the randomness so that this holds. Moreover, the sum-check theorem together with this equation, makes sure that the randomness does not affect divisibility at 0 of $(\tilde{A}(X), \tilde{B}(X), \tilde{A}(X)\tilde{B}(X)) \cdot \mathbf{D}(X)(mod(t(X)))$.
For soundness, observe that $\mathfrak{W}(\mathfrak{a}^\intercal, \mathfrak{b}^\intercal, (\mathfrak{a} \circ \mathfrak{b})^\intercal) = 0$ is equivalent to (1) $\mathbf{a} = \mathbf{F}(\mathbf{a} \circ \mathbf{b})$,

(2) $\mathbf{b} = \mathbf{G}(\mathbf{a} \circ \mathbf{b})$ and (3) $\sum_{i=1}^{n} r_{a,i} + r_{b,i} = 0$, for $\mathbf{a} \coloneqq (1, \mathbf{x}, \mathbf{a}'), \mathbf{b} \coloneqq (\mathbf{1_l}, \mathbf{b}')$. 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 $(A(X), B(X), A(X)B(X))^\intercal \cdot \mathbf{D}(X)(mod(t(X)))$.
Perfect zero-knowledge of the PHP follows from the fact that, all messages in the CSS procedure contain only public information and the rest of information are oracle polynomials.
For the honest-verifier bounded zero-knowledge, 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 they are independent of the witness.
For an oracle query at some point γ, the simulator samples uniform random values $A'_\gamma, B'_\gamma, R_{\gamma,t} \in \mathbb{F}$ and declares them as $A'(\gamma), B'(\gamma), R_t(\gamma)$. It then defines the rest of the values to be consistent with them. More in detail, let $\mathbf{D}(X)^\intercal = \mathbf{s}^\intercal \mathbf{W} \lambda(X) = (D_a(X), D_b(X), D_{ab}(X))$ be the output of the CSS argument. Then the simulator sets:

$$A_\gamma = A'_\gamma t_l(\gamma) + \sum_{i=1}^{l} x_i \lambda_i(\gamma) B_\gamma = B'_\gamma t_l(\gamma) + 1 \tag{7.3.1}$$

$$p_\gamma = D_a(\gamma) A_\gamma + D_b(\gamma) B_\gamma + D_{ab}(\gamma) A_\gamma B_\gamma H_{t\gamma} = (p_\gamma - \gamma R_{t,\gamma})/t(\gamma) \tag{7.3.2}$$

The simulator keeps a table of the computed values to answer consistently the oracle queries.
Now, since the verifier is honest and $|\mathbb{H}|$ is assumed to be a fraction of the field elements, we can assume that the verifier chooses a $\gamma \in \mathbb{F} \setminus \mathbb{H}$. In this case, the polynomial encoding of $\mathbf{r}_a, \mathbf{r}_b$ acts as a masking polynomial for $A'(X), B'(X), R_t(X), H_t(X)$ and taking into account that $\sum_{i=1}^{n} r_{a,i} + r_{b,i} = 0$ to have the same distribution it suffices to take $2n-1 \geq b_A + b_B + b_{R_t} + b_{H_t}$, and $n \geq max(b_A, b_B)$ as stated in the theorem. □

CHAPTER 8

CHECKABLE SUBSPACE SAMPLING [RZ21]

8.1 Overview

We focus our attention now to construction of CSS arguments for the matrix of R1CS-lite', namely:

$$\mathbf{W} = \begin{pmatrix} \mathbf{I} & \mathbf{0} & -\mathbf{F} \\ \mathbf{0} & \mathbf{I} & -\mathbf{G} \end{pmatrix} \in \mathbb{F}^{2m \times 3m}$$

A fundamental observation made in [RZ21] is that the matrix $\mathbf{W}$ can be seen as a matrix with three blocks $(\mathbf{W_a}, \mathbf{W_b}, \mathbf{W_c})$ and sampling in each of these blocks must be done separately (using the same coins from the verifier), as the prover needs to receive $(D_a(X), D_b(X), D_c(X))$ to do the inner product with $(A(X), B(X), C(X))$, where $C(X) = A(X)B(X)$. A naive approach is to obtain these polynomials by running one CSS scheme for each of the matrices, but a more careful approach can save elements in communication complexity:

1. For a matrix of 2m rows, we can split it into two blocks of m rows $\mathbf{M} = \begin{pmatrix} \mathbf{M}_1 \\ \mathbf{M}_2 \end{pmatrix}$, use the same CSS argument for each marrix with the smae coins and combine them to save on communication. More in detail, if $\mathbf{s} = \text{Smp}(\text{cns})$, and $D_1(X) = \mathbf{s}^\intercal \mathbf{M_1} \lambda(X)$ and $D_2(X) = \mathbf{s}^\intercal \mathbf{M_2} \lambda(X)$, are the polynomials associated to $\mathbf{M}_1, \mathbf{M}_2$, the argument is modified so that it sends $D_1(X) + zD_2(X)$ for some challenge z chosen by the verifier. Note that this polynomial equals $(\mathbf{s}^\intercal, z\mathbf{s}^\intercal)\mathbf{M}\lambda(X)$, that is, it corresponds to a CSS argument where the sampling coefficients depend on z also. It is important to note here that since this modification corresponds to implicitly constructing a CSS argument for $\mathbf{M}_1 + z\mathbf{M}_2$, it is not always the case that this can be done. The first reason is that the polynomials computed by the indexer of the CSS argument for the matrices must be able to be combined upon receiving z to a CSS indexer polynomial for $\mathbf{M}_1 + z\mathbf{M}_2$, and secondly that $\mathbf{M}_1 + z\mathbf{M}_2$ must be an admissible matrix for the CSS argument. We will see later that these conditions are met for sparse matrices and matrices with at most V non-zero entries per column.

2. When a block of size $m \times m$ is trivial, i.e. 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 $\mathbf{I} \in \mathbb{F}^{m\times m}$, the value of $D(y)$ will end up being $(t(x)y - xt(y))$ where x are the verifier coins, and thus can be computed by the verifier in $\mathcal{O}(logm)$ field operations. Thus, when $\mathbf{W}_a, \mathbf{W}_b$ are as in R1CS-lite', we can use the approach of (a) to cut them into blocks of $m \times m$ size and then the verifier can open $D_a(X), D_b(X)$ himself, so there is no use for a CSS scheme to prove correct sampling.

After the previous discussion we need to turn our focus to constructing CSS schemes for matrices $\mathbf{M} \in \mathbb{F}^{m\times m}$ and then use one for each block of $\mathbf{W}$. We will give in this section CSS arguments presented in [RZ21] for different types of sparse matrices. Two different constructions are going to be presented:

- A construction that works for general sparse matrices.

- A construction that assumes a maximum bound on the non-zero elements of a column of the matrix.

For this section we consider two disjoint sets of roots of unity, $\mathbb{H}, \mathbb{K}$. For $\mathbb{H}$ we will use the established notation. for $\mathbb{K}$, assume a canonical order and let $K, k_l, \mu_l, u(X)$ be its cardinal, the l-th element, the l-th Lagrange basis polynomial associated to $\mathbb{K}$ and the vanishing polynomial respectively.

Matrices $\mathbf{M} \in \mathbb{F}^{m\times m}$ can be naturally encoded as a bivariate polynomial $P(X, Y) = \mathbf{a}(Y)^\intercal \mathbf{M}\beta(X)$, for some $\mathbf{a}(Y) \in \mathbb{F}[Y]^m, \beta(X) \in \mathbb{F}[X]^m$, which can be thought of as a zipping of the rows followed by a zipping of the columns. Then, for some $x \in \mathbb{F}$:

$$P(X, x) = \mathbf{a}(x)^\intercal \mathbf{M}\beta(X) = \textstyle\sum_{i=1}^{m} a_i(x)\mathbf{m}_i^\intercal \beta(X)$$

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 $\beta(X)$ with coefficients $a_i(x)$. This suggests to construct a CSS scheme where, in the proving phase, the verifier sends the challenge x and the prover replies with $D(X) = P(X, x)$ and subsequently in the proving phase, the prover convinces the verifier that $D(X)$ is correctly computed according to the coins x.

In Marlin, Lunar and [RZ21] they set $\mathbf{a}(Y) = \lambda(Y), \beta(X) = \lambda(X)$. The common strategy used was introduced by [Mal+19] and goes as follows: 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 $\sigma = P(y, x)$. This proves (with overwhelming probability) that $D(X) = P(X, x)$. The last step is the most challenging and is in fact the main technical novelty of the aforementioned works in pre-processing zkSNARKs. In all of them this is achieved by restricting the set of admissible matrices of the CSS argument.

Assuming we have a mareix $\mathbf{M} \in \mathbb{F}^{m\times m}$ with $|\mathbf{M}| = K$ and that its non-zero entries are ordered, the matrix can be represented as proposed in [Chi+19], by three functions $\mathrm{r} : \mathbb{K} \to [m], \mathrm{c} : \mathrm{K} \to [m], \mathrm{v} : \mathbb{K} \to \mathbb{F}$, specifying the row, column and value of the l-th non-zero entry. Then the bivariate polynomial can be written as:

$$P(X, Y0 = \textstyle\sum_{l=1}^{K} v(k_l)\lambda_{r(k_l)}(Y)\lambda_{c(k_l)}(X)$$

Then, the authors of [RZ21] observed, that in order to see if $P(y,x)$ is correctly computed, it can be written as:

$$P(y,x) = (\lambda_{r(k_1)}(x), \ldots, \lambda_{r(k_K)}(x)) \cdot (v(k_1)\lambda_{c(k_1)}(y), \ldots, v(k_K)\lambda_{c(k_K)}(y))$$

Then defining the low degree extentions of each of these vectors respectively as:

$$e_x(X) = \sum_{l=1}^{K} \lambda_{r(k_l)}(x)\mu_l(X), \quad e_y(X) = \sum_{l=1}^{K} v(k_l)\lambda_{c(k_l)}(y)\mu_l(X)$$

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

The key observation made in [RZ21] is that $e_x(X) = \lambda(x)^\intercal \mathbf{M}_x \mu(X)$ and $e_y(X) = \lambda(y)^\intercal \mathbf{M}_y \mu(X)$, for some matrices $\mathbf{M}_x, \mathbf{M}_y$ with at most one non-zero element per column. Thus to prove they are correctly computed it is sufficient to design a CSS argument for these simple matrices. The idea behind the construction for these types of matrices is that given an arbitrary $e_x(X) = \sum_{l=1}^{K} v(k_l)\lambda_{f(k_l)}(x)\mu_l(X)$ for some $f : \mathbb{K} \to [m]$, we can "complete" the Lagrange $\lambda_{f(k_l)}(x)$ with the missing term $(x - h_{f(k_l)})$ to get the vanishing polynomial t(x). Now, the low degree extension of these "completing terms" is $x - u_1(X) = x - \sum_{l=1}^{K} h_{f(k_l)}\mu_l(X)$ can be computed by the indexer

8.2 CSS Argument for Simple Matrices

The basic building block is a CSS argument for matrices $\mathbf{M} = (m_{ij}) \in \mathbb{F}^{m \times K}$ with at most one non-zero value in each column. We define two functions associated to $\mathbf{M}$, $v : \mathbb{K} \to \mathbb{F}$, $f : \mathbb{K} \to [m]$. Given an element $k_l \in \mathbb{K}$, $v(k_l) = m_{f(k_l),l} \neq 0$, i.e. the function v outputs the only non zero value of column l and f the corresponding row; if such a value does not exist set $v(k_l) = 0$ and $f(k_l)$ arbitarily. We define the polynomial $P(X,Y)$ such that $D(X) = P(X,x)$ as $P(X,Y) = \lambda(Y)\mathbf{M}\mu(X)$. Now, by definition of the functions v and f, it holds that $P(X,Y) = v(k_l)\lambda_{f(k_l)}(Y)\mu_l(X)$.

Online phase: $\mathcal{I}_{CSS}(\mathbb{F}, \mathbf{M})$ outputs $\mathcal{W}_{CSS} = \{u_1(X), u_2(X)\}$, where

$$u_1(X) = \sum_{l=1}^{K} h_{f(k_l)}\mu_l(X), \quad u_2(X) = m^{-1}\sum_{l=1}^{K} v(k_l)h_{f(k_l)}\mu_l(X)$$

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

$$D(X)(x - u_1(X)) = t(x)u_2(X) + H_u(X)u(X)$$

Decision Phase: Accept if and only if (1) $\deg D(X) \leq K - 1$, and (2) $D(X)(x - u_1(X)) = t(x)u_2(X) + H_u(X)u(X)$

Theorem 8.1. The argument satisfies completeness and perfect soundness.

Proof. When evaluated at any $k_l \in \mathbb{K}$, the RHS of the verification equation becomes $t(x)u_2(k_l) = t(x)v(k_l)h_{f(k_l)}m^{-1}$. Now the LHS computes to:

$$D(k_l)(x - u_1(k_l)) = (v(k_l)\lambda_{f(k_l)}(x))(x - h_{f(k_l)}) = t(x)v(k_l)h_{f(k_l)}m^{-1}$$

For soundness, since the degree of $D(X)$ is at most $K - 1$ and for $\forall k_l \in \mathbb{K}$ it holds that $D(k_l) = v(k_l)\lambda_{f(k_l)}$, it holds that $D(X) = \sum_{l=1}^{K} v(k_l)\lambda_{f(k_l)}\mu_l(X)$. □

8.3 CSS argument for Sparse Matrices

Offline Phase: $\mathcal{I}_C SS$ outputs $\mathcal{W}_{CSS} = (u_r(X), u_{1,c}(X), u_{2,c}(X)$ where:

$$u_r(X) = \sum_{l=1}^{K} h_{r(k_l)}\mu_l(X)$$
$$u_{1,c}(X) = \sum_{l=1}^{K} h_{c(k_l)}\mu_l(X), \quad u_{2,c}(X) = m^{-1}\sum_{l=1}^{K} v(k_l)h_{c(k_l)}\mu_l(X)$$

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

ProveSampling: $\mathcal{V}_{CSS}$ sends $y \leftarrow \mathbb{F}$ and $\mathcal{P}_{CSS}$ outputs $\sigma = D(y)$ and $e_x(X), e_y(X)$ where $e_x(X) = \sum_{l=1}^{K} \lambda_{r(k_l)}(x)\mu_l(X), e_y(X) = \sum_{l=1}^{K} v(k_l)\lambda_{c(k_l)}(y)\mu_l(X), \mathcal{V}_{CSS}$ sends $z \leftarrow \mathbb{F}$ and $\mathcal{P}_{CSS}$ computes $H_{u,x}(X), H_{u,y}(X), R_u(X), H_{u,x,y}(X)$ such that:

$$e_x(X)(x - u_r(X)) = m^{-1}t(x)u_r(X) + H_{u,x}(X)u(X)$$
$$e_y(X)(y - u_{1,c}(X)) = t(y)u_{2,c}(X) + H_{u,y}(X)u(X)$$
$$Ke_x(X)e_y(X) - \sigma = XR_u(X) + u(X)H_{u,x,y}$$

It also defines $H_u(X) = H_{u,x,y}(X) + zH_{u,x}(X) + z^2 H_{u,y}(X)$ and outputs $(R_u(X), H_u(X))$.

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

$$e_x(X) + z^2 i_y(X))(e_y(X) + zi_x(X)) - z^3 i_x(X)i_y(X) - z^2 t(y)u_{2,c}(X) - \sigma/K - zt(x)m^{-1}u_r(X)$$
$$=$$
$$XR_u(X) + H_u(X)u(X)$$

Theorem 8.2. The argument satisfies completeness and ϵ-soundness.

Proof. Completeness follow immediately. For soundness, the prover is showing, in a batched form, that the following equations are satisfied,

$$e_x(X)(x - u_r(X)) = m^{-1}t(x)u_r(X) + H_{u,x}(X)u(X)$$
$$e_y(X)(y - u_{1,c}(X)) = t(y)u_{2,c}(X) + H_{u,y}(X)u(X)$$
$$Ke_x(X)e_y(X) - \sigma = XR_u(X) + u(X)H_{u,x,y}$$

Since all the left terms of the equations are defined before the verifier sends z, by the Schwartz-Zippel lemma, with all but $3/|\mathbb{F}|$ probability, the verifier accepts if and only if such $H_{u,x}(X), H_{u,y}(X), R_u(X), H_{u,x,y}(X)$ exist.
If they do, the rest of the proof is a consequence of (1) soundness of the previous protocol, which implies that $e_x(X), e_y(X)$ correspond to the correct polynomials modulo $u(X)$, and (2) the following lemma, which shows that if the last equation is satisfied, and $e_x(X), e_y(X)$ are the correct polynomials, then $\sigma = P(y, x)$. Because the prover sends $D(X)$ before receiving y and $D(y) = \sigma$, again from the Schwartz-Zippel lemma, except with negligible probability, it holds that $P(X, x) = D(X)$ □

Lemma 8.3. Given $e_x(X), e_y(X)$ such that $e_x(X) = \sum_{l=1}^{K} \lambda_{r(k_l)}(x)\mu_l(X), e_y(X) = \sum_{l=1}^{K} v(k_l)\lambda_{c(k_l)}(y)\mu_l(X)$ then $P(y, x) = \sum_{l=1}^{K} v(k_l)\lambda_{c(k_l)}(y)\lambda_{r(k_l}(x) = \sigma$ if and only if there exist polynomials $R_u(X) \in \mathbb{F}_{\leq m-2}[X], H_{u,x,y}(X)$ such that:

$$e_x(X)e_y(X) - \sigma/K = XR_u(X) + H_{u,x,y}(X)u(X)$$

Proof. Note that $e_x(X)e_y(X) = \sum_{l=1}^{K} v(k_l)\lambda_{c(k_l)}(y)\lambda_{r(k_l)}(x)\mu_l(X)(mod(u(X)))$. By the univariate sumcheck $e_x(X)e_y(X) - \sigma/K$ is divisible by X if and only if $P(y, x) = \sigma$, which concludes the proof. □

8.4 CSS Argument for Sums of Simple Matrices

We now give a CSS argument for a matrix $\mathbf{M} \in \mathbb{F}^{m \times K}$ that can be written as $\sum_{i=1}^{V} \mathbf{M}_i$, each of which has at most one non-zero element in each column. Note that any matrix with at most V non-zero entries per column can be written this way, and in particular this holds for constraint matrices coming from circuits with fun-out bounded by V. We define two functions associated to each $\mathbf{M}_i, v_i \mathbb{K} \to \mathbb{F}, f_i : \mathbb{K} \to [m]$ as in the section for simple matrices.

Define $P(X, Y) = \lambda(Y)\mathbf{M}\mu(X)$ and $D(X) = P(X, x)$. Using the above defined functions we can write $P(X, Y) = \sum_{i=1}^{V} \sum_{l=1}^{K} v_i(k_l)\lambda_{f_i(k_l)}(Y)\mu_l(X)$. Define $S_l = \{f_i(k_l) : i \in [V]\}$ and $S_l^c = [m] - S_l$. The intuition is that, since there are at most V non zero $v_i(k_l)$ for each l, we can factor as:

$$P(k_l, x) = \sum_{i=1}^{V} v_i(k_l)\lambda_{f_i(k_l)}(x) = \prod_{s \in S_l^c}(x - h_s)R_l(x)$$

where $R_l(X)$ is a polynomial of degree $V - 1$. So, to "complete" $P(k_l, x)$ to be a multiple of $t(x)$, we need to multiply it by $\prod_{s \in S_l}(x - h_s)$, and the result will be $t(x)R_l(x)$. The trick is that $\hat{I}_l(Y) = \prod_{s \in S_l}(Y - h_s)$ and $R_l(X)$ are polynomials of degrees V, V-1 respectively. So, the indexer can publish the coefficients of these polynomials in the monomial basis and they can be reconstructed by the verifier with coefficients $1, x, \ldots, x^V$.

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

$$\hat{R}_l(Y) = \frac{1}{m}\sum_{i=1}^{V} v_i(k_l)h_{f_i(k_l)} \prod_{s \in S_l - \{f_i(k_l)\}}(Y - h_s) = \sum_{j=0}^{V-1} \hat{R}_{lj}Y^j$$
$$\hat{I}_l(Y) = \prod_{s \in S_l}(Y - h_s) = \sum_{j=0}^{V} \hat{I}_{lj}Y^j$$

Define

$$u_j^{\hat{R}}(X) = \sum_{l=1}^{K} \hat{R}_{lj}\mu_l(X) \quad u_j^{\hat{I}}(X) = \sum_{l=1}^{K} \hat{I}_{lj}\mu_l(X)$$

Outuput $\mathcal{W}_{CSS} = \{\{u_j^{\hat{I}}(X)\}_{j=0}^{V}, \{u_j^{\hat{R}}(X)\}_{j==0}^{V-1}\}$

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

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

$$D(X)\hat{I}_x(X) = t(x)\hat{R}_x(X) + H_u(X)u(X)$$

Decision Phase: Accept if and only if (1) $\deg(D) \leq K - 1$, and (2) $D(X)\hat{I}_x(X) = t(x)\hat{R}_x(X) + H_u(X)u(X)$

Theorem 8.4. The previous argument satisfies completeness and perfect soundness.

Proof. When evaluated in any $k_l \in \mathbb{K}$, the RHS of the verification equation is:

$$t(x)\hat{R}_x(x) = \frac{t(x)}{m}\sum_{i=1}^{V} v_i(k_l)h_{f_i(k_l)} \prod_{s \in S_l - \{f_i(k_l)\}}(x - h_s)$$
$$= \sum_{i=1}^{V} v_i(k_l)\frac{h_{f_i(k_l)}}{m}\frac{t(x)}{x - h_{f_i(k_l)}} \prod_{s \in S_l}(x - h_s) =$$
$$\prod_{s \in S_l}(x - h_s)\sum_{i=1}^{V} v_i(k_l)\lambda_{f_i(k_l)}(x)$$

The LHS of the equation is the same if we substitute the terms, so completeness follows.

For soundness, if the verifier accepts $D(X)$, then because $\hat{I}_x(k_l) = \hat{I}_l(x)$, we get:

$$D(k_l) = \hat{I}_l(x)^{-1} t(x) \hat{R}_l(x) = (\prod_{s \in S_l^c} (x - h_s)) \hat{R}_x(x) = \sum_{i=1}^{V} v_i(k_l) \lambda_{f_i(k_l)}(x)$$

We conclude that $D(X) = P(X, x)(mod(u(x)))$ and since these are both at most $K-1$ degree polynomials, soundness is proven. □

CHAPTER 9

CONCRETE CONSTRUCTION OF ZKSNARK: BASILISK [RZ21]

For the readers convenience we give a full representation of the zkSNARK construction Basilisk of [RZ21] for 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. This construction combines the building blocks presented in this paper. More in detail, it runs the CSS argument for matrices that are sums of simple matrices, instantiates the PHP for linear argument for the outputs of the CSS scheme and pairs it with the Hadamard product relation in order to instantiate the PHP for R1CS-lite'. Moreover it expands the matrix of constraints and the witness vectors, in the way described previously, in order to achieve zero-knowledge.

As already discussed, since the other blocks of m rows are the identity or the zero matrices, 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, first write $\mathbf{W}_c = \sum_{i=1}^{V} \begin{pmatrix} \mathbf{F}_i \\ \mathbf{G}_i \end{pmatrix}$ where each $\begin{pmatrix} \mathbf{F}_i \\ \mathbf{G}_i \end{pmatrix}$ is a simple matrix. As pointed out, we will implicitly construct the scheme for $\hat{\mathbf{W}} = \mathbf{F} + z_1\mathbf{G}$, that can be written as $\sum_{i=1}^{V} \mathbf{F}_i + z_1\mathbf{G}_i$, with each summand being a matrix with at most one non-zero element per column. We define two functions associated to each $\mathbf{W}_{c,i} = \begin{pmatrix} \mathbf{F}_i \\ \mathbf{G}_i \end{pmatrix}$. The function $r_i : \mathbb{H} \to [2m]$ that, given an element $h_l \in \mathbb{H}$ it outputs the row corresponding to the only non-zero element in column l of matrix $\mathbf{W}_{c,i}$ and the function $v_i : \mathbb{H} \to \mathbb{F}$ that outputs the value of this non-zero entry. For simplicity of notation, we also define the sets $V_l^1 = \{i \in [V] : 1 \leq r_i(h_l) \leq m\}, V_l^2 = \{i \in [V] : m+1 \leq r_i(h_l) \leq 2m\}, S_l = \{\{r_i(h_l) : i \in V_l^1\} \cup \{r_i(h_l) - m : i \in V_l^2\}\}$ and $\hat{V}_i^1 = \{l \in [m] : 1 \leq r_i(h_l) \leq m\}, \hat{V}_i^2 = \{l \in [m] : m+1 \leq r_i(h_l) \leq 2m\}$.

If $\iota(1) = a, \iota(2) = b, \iota(3) = c$, and for $i = 1, 2, k = 1, 2$, let $(P')^k_{\iota(i)}(X, Y) = \lambda(Y)^\intercal(\mathbf{W}^1_{\iota(i)} + z_1\mathbf{W}^2_{\iota(i)})\lambda(X)$, and $(D')^k_{\iota(i)}(X) = (P')^k_{\iota(i)}(X, x)$.

Elements in blue are added to achieve zero-knowledge.

KeyGen($\mathcal{R}$: Sample $\tau \leftarrow \mathbb{F}$ and output $\tau, srs_u = (\{[\tau^i]_1\}_{i=0}^{m-1}, \{[\tau^i]_1\}_{i=m}^{m+5}, [\tau]_2)$. Choose an arbitrary $u \in \mathbb{F}^*, u \notin \mathbb{H}$.

KeyGenD$(srs_u, \mathbf{W}', \mathbf{w}')$: Parse $\mathbf{W}' = (\mathbf{W}'_a, \mathbf{W}'_b, \mathbf{W}'_c)$ and $\mathbf{W}'_c$ as $\begin{pmatrix} \mathbf{F} & 0_{m\times 6} \\ \mathbf{G} & 0_{m\times 6} \end{pmatrix}, \mathbf{F}, \mathbf{G} \in \mathbb{F}^{m\times m}$. For $i \in [V], k = 1, 2$ define $\hat{R}_l^k(Y)$ and its coefficients $\hat{R}_{lj}^k$ as:

$$\hat{R}_l^k(Y) = \frac{1}{m}\sum_{i\in V_l^k} v_i(h_l) h_{r_i(h_l)-(k-1)m} \prod_{s\in S_l - \{r_i(h_l)-(k-1)m\}}(Y - h_s) = \sum_{j=0}^{V-1} \hat{R}_{lj}^k Y^j,$$

Also, let $\hat{I}_l(Y)$ and $\hat{I}_{lj}$ be such that $\hat{I}_l(Y) = \prod_{s\in S_l}(Y - h_s) = \sum_{j=0}^{V} \hat{I}_{lj} Y^j$.
Finally, for $j = 0, \ldots, V-1$ define $u_j^{\hat{R},1}(X) = \sum_{l=1}^{m} \hat{R}_{lj}^1 \lambda_l(X)$, $u_j^{\hat{R},2}(X) = \sum_{l=1}^{m} \hat{R}_{lj}^2 \lambda_l(X)$, and, for $j = 0, \ldots, V u_j^{\hat{I}}(X) = \sum_{l=1}^{m} \hat{I}_{lj}\lambda_l(X)$. Compute $[u_j^{\hat{I}}]_1 = [u_j^{\hat{I}}(\tau)]_1, [u_j^{\hat{R},1}]_1 = [u_j^{\hat{R},1}(\tau)]_1, [u_j^{\hat{R},2}]_1 = [u_j^{\hat{R},2}(\tau)]_1$.
Output $srs_W = (srs_u, \{[u_j^{\hat{I}}]_1\}_{j=0}^{V}, \{[u_j^{\hat{R},1}]_1, [u_j^{\hat{R},2}]_1\}_{j=0}^{V-1})$.
Prove$(\mathbf{W}, srs_W, (\mathbf{x}, (\mathbf{a}', \mathbf{b}')))$: Sample $r_a \leftarrow \mathbb{F}^4, r_b \leftarrow \mathbb{F}^2$ 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 $A(X) = \sum_{j=1}^{m+6} a_j\lambda_j(X), B(X) = \sum_{j=1}^{m+6} b_j\lambda_j(X), B'(X) = (B(X) - 1)/(t_l(X)\prod_{i=1}^{4}(X - hm + 1))$ and

$$A'(X) = ((\sum_{j=l+1}^{m+6} a_j\lambda_j(X)) - t_l(X))/(t_l(X)(X - h_{m+5})(X - h_{m+6}))$$

Output $\pi_1 = ([A']_1 = [A'(\tau)]_1, [B']_1 = [B'(\tau)]_1)$.
Verify$(srs_W, \mathbf{x}, \pi_1)$: Send $x, z_1, z_2 \leftarrow \mathbb{F}$.
Prove$(\mathbf{W}, srs_W, (\mathbf{x}, (\mathbf{a}', \mathbf{b}')), x, z, z_2)$: For $i = 1, 2, 3$ let $D'_{\iota(i)}(X) = (D')^1_{\iota(i)}(X) + z_1(D')^2_{\iota(i)}(X)$. Let $D_a(X) = D'_a(X) + z_1^2\sum_{j=m+1}^{m+6}\lambda_j(X)$, $D_b(X) = D'_b(X) + z_1^2\sum_{j=m+1}^{m+6}\lambda_j(X)$ and $D_c(X) = D'_c(X)$.
Find $R(X), H_1(X), H_2(X)$ such that:

$$A(X)D_a(X)(X - u) + B(X)D_b(X)(X - u) - D_c(X)A(X)B(X)(X - u) = XR(X) + t(X)H_1(X)(X - u)$$

and, if $\hat{R}_x(X) = \sum_{j=0}^{V-1} x^j(u_j^{\hat{R},1}(X) = z_1 u_j^{\hat{R},2}(X))$ and $\hat{I}_x(X) = \sum_{j=0}^{V} x^j u_j^{\hat{I}}(X)$,

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

Output $\pi_2 = ([D_c]_1 = [D_c(\tau)]_1, [H]_1 = [H(\tau)]_1 + z_2[H_(\tau)]_1, [R]_1 = [R(\tau)]_1$.
Verify$(srs_W, \mathbf{x}, \pi_1, \pi_2)$: Send $y, \gamma \leftarrow \mathbb{F}$.
Prove$(\mathbf{W}, srs_W, (\mathbf{x}, \mathbf{a}', \mathbf{b}')), x, z_1, z_2, y, \gamma)$: Define $\sigma = D_c(y)$ and, for

$$E(X) = A(y)D_a(y)(y - u) + B(X)D_b(y)(y - u) + \sigma(-A(y)B(X)(y - u) + z_2\hat{I}_x(X)) - yR(X) - z_2t(x)\hat{R}_x(X) - t(y)H(X)(y - u)$$

$\mathbf{p}(X) = (A(X), D_c(X), E(X))$ and $\mathbf{d} = (m-1, m-1, m-1)$, calculate $([w]_1, (\alpha, \sigma, 0)) \leftarrow$ PC.Open$(srs_u, \mathbf{p}(X), \mathbf{d}, y, \gamma)$
Output $\pi_3 = ([w]_1, (\alpha, \sigma))$.
Verify $(srs_W, \mathbf{x}, \pi_1, \pi_2, \pi_3)$: Define $s = \alpha + \gamma\sigma$ and $D_a = D_b = (t(x)y - xt(y))/(x -$

$y) - \sum_{j=m+1}^{m+6} \lambda_j(x)\lambda_j(x)$. Compute $[A]_1 = ([A']_1(y-h_{m+5})(y-h_{m+6})+1)t_l(y)+ \sum_{i=1}^{l} x_i[\lambda_i(\tau)]_1$, $[B]_1 = ([B']_1 t_l(y) \prod_{i=1}^{4}(y-h_{m+i})+1)$, $[\hat{R}_x]_1 = \sum_{j=0}^{V-1} x^j([u_j^{\hat{R},1}]_1+ z_1[u_j^{\hat{R},2}]_1)$, $[\hat{I}_x]_1 = \sum_{j=0}^{V} x^j[u_j^{\hat{I}}]_1$ and

$$[p]_1 = [A]_1 + \gamma[D_c]_1 + \gamma^2(\alpha D_a + z_1 D_b[B]_1 + \sigma(-\alpha[B]_1(y-u) + z_2[\hat{I}_x]_1) - y[R]_1 - z_2 t(x)[\hat{R}_x]_1 - t(y)[H]_1(y-u))$$

Output 1 if and only if

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

BIBLIOGRAPHY

[FS87] Amos Fiat and Adi Shamir. ``How to prove yourself: Practical solutions to identification and signature problems". אנגלית. In: *Advances in Cryptology — CRYPTO 1986 - Proceedings*. Ed. by Andrew M. Odlyzko. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Publisher Copyright: © 1987, Springer-Verlag Berlin Heidelberg.; null ; Conference date: 11-08-1986 Through 15-08-1986. Springer Verlag, 1987, pp. 186–194. ISBN: 9783540180470. DOI: 10.1007/3-540-47721-7_12.

[IY87] Russell Impagliazzo and Moti Yung. ``Direct Minimum-Knowledge Computations." In: vol. 293. Aug. 1987, pp. 40–51. DOI: 10.1007/3-540-48184-2_4.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. ``The Knowledge Complexity of Interactive Proof Systems". In: *SIAM J. Comput.* 18.1 (Feb. 1989), pp. 186–208. ISSN: 0097-5397. DOI: 10.1137/0218012. URL: https://doi.org/10.1137/0218012.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. ``Proofs That Yield Nothing but Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems". In: *J. ACM* 38.3 (July 1991), pp. 690–728. ISSN: 0004-5411. DOI: 10.1145/116825.116852. URL: https://doi.org/10.1145/116825.116852.

[Kil92] Joe Kilian. ``A Note on Efficient Zero-Knowledge Proofs and Arguments (Extended Abstract)". In: *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*. STOC '92. Victoria, British Columbia, Canada: Association for Computing Machinery, 1992, pp. 723–732. ISBN: 0897915119. DOI: 10.1145/129712.129782. URL: https://doi.org/10.1145/129712.129782.

[Sha92] Adi Shamir. ``IP = PSPACE". In: *J. ACM* 39.4 (Oct. 1992), pp. 869–877. ISSN: 0004-5411. DOI: 10.1145/146585.146609. URL: https://doi.org/10.1145/146585.146609.

[Mic94] S. Micali. ``CS Proofs". In: *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. SFCS '94. USA: IEEE Computer Society, 1994, pp. 436–453. ISBN: 0818665807. DOI: 10.1109/SFCS.1994.365746. URL: https://doi.org/10.1109/SFCS.1994.365746.

[GK96] Oded Goldreich and Hugo Krawczyk. ``On the Composition of Zero-Knowledge Proof Systems". In: *SIAM Journal on Computing* 25 (Jan. 1996). DOI: 10.1007/BFb0032038.

[For99] Lance Fortnow. ``The Complexity of Perfect Zero-Knowledge". In: *Conference Proceedings of the Annual ACM Symposium on Theory of Computing* 5 (Apr. 1999). DOI: 10.1145/28395.28418.

[Cho+06] Sherman S. M. Chow et al. ``Ring Signatures without Random Oracles". In: *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*. ASIACCS '06. Taipei, Taiwan: Association for Computing Machinery, 2006, pp. 297–302. ISBN: 1595932720. DOI: 10.1145/1128817.1128861. URL: https://doi.org/10.1145/1128817.1128861.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. ``Constant-Size Commitments to Polynomials and Their Applications". In: *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security*. Vol. 6477. Lecture Notes in Computer Science. Springer, 2010, pp. 177–194. DOI: 10.1007/978-3-642-17373-8_11. URL: https://www.iacr.org/archive/asiacrypt2010/6477178/6477178.pdf.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. *Interactive Oracle Proofs*. Cryptology ePrint Archive, Paper 2016/116. https://eprint.iacr.org/2016/116. 2016. URL: https://eprint.iacr.org/2016/116.

[Boo+16] Jonathan Bootle et al. *Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting*. Cryptology ePrint Archive, Paper 2016/263. https://eprint.iacr.org/2016/263. 2016. URL: https://eprint.iacr.org/2016/263.

[Bün+17] Benedikt Bünz et al. *Bulletproofs: Short Proofs for Confidential Transactions and More*. Cryptology ePrint Archive, Paper 2017/1066. https://eprint.iacr.org/2017/1066. 2017. URL: https://eprint.iacr.org/2017/1066.

[FKL17] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. *The Algebraic Group Model and its Applications*. Cryptology ePrint Archive, Paper 2017/620. https://eprint.iacr.org/2017/620. 2017. URL: https://eprint.iacr.org/2017/620.

[Gro+18] Jens Groth et al. *Updatable and Universal Common Reference Strings with Applications to zk-SNARKs*. Cryptology ePrint Archive, Paper 2018/280. https://eprint.iacr.org/2018/280. 2018. URL: https://eprint.iacr.org/2018/280.

[Bün+19] Benedikt Bünz et al. *Proofs for Inner Pairing Products and Applications*. Cryptology ePrint Archive, Paper 2019/1177. https://eprint.iacr.org/2019/1177. 2019. URL: https://eprint.iacr.org/2019/1177.

[Chi+19] Alessandro Chiesa et al. *Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS*. Cryptology ePrint Archive, Paper 2019/1047. https://eprint.iacr.org/2019/1047. 2019. URL: https://eprint.iacr.org/2019/1047.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. *PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge*. Cryptology ePrint Archive, Paper 2019/953. https://eprint.iacr.org/2019/953. 2019. URL: https://eprint.iacr.org/2019/953.

[Mal+19] Mary Maller et al. *Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updateable Structured Reference Strings*. Cryptology ePrint Archive, Paper 2019/099. https://eprint.iacr.org/2019/099. 2019. URL: https://eprint.iacr.org/2019/099.

[VP19] Alexander Vlasov and Konstantin Panarin. *Transparent Polynomial Commitment Scheme with Polylogarithmic Communication Complexity*. Cryptology ePrint Archive, Paper 2019/1020. https://eprint.iacr.org/2019/1020. 2019. URL: https://eprint.iacr.org/2019/1020.

[Cam+20] Matteo Campanelli et al. *Lunar: a Toolbox for More Efficient Universal and Updatable zkSNARKs and Commit-and-Prove Extensions*. Cryptology ePrint Archive, Paper 2020/1069. https://eprint.iacr.org/2020/1069. 2020. URL: https://eprint.iacr.org/2020/1069.

[Beh+21a] Pourandokht Behrouz et al. ``Designated-Verifier Linkable Ring Signatures". In: *Information Security and Cryptology - ICISC 2021 - 24th International Conference, Seoul, South Korea, December 1-3, 2021, Revised Selected Papers*. Ed. by Jong Hwan Park and Seung-Hyun Seo. Vol. 13218. Lecture Notes in Computer Science. Springer, 2021, pp. 51–70. DOI: 10.1007/978-3-031-08896-4_3. URL: https://doi.org/10.1007/978-3-031-08896-4%5C_3.

[Beh+21b] Pourandokht Behrouz et al. ``Designated-Verifier Linkable Ring Signatures". In: *Information Security and Cryptology - ICISC 2021 - 24th International Conference, Seoul, South Korea, December 1-3, 2021, Revised Selected Papers*. Ed. by Jong Hwan Park and Seung-Hyun Seo. Vol. 13218. Lecture Notes in Computer Science. Springer, 2021, pp. 51–70. DOI: 10.1007/978-3-031-08896-4_3. URL: https://doi.org/10.1007/978-3-031-08896-4%5C_3.

[Gro+21] Panagiotis Grontas et al. ``Publicly auditable conditional blind signatures". In: *J. Comput. Secur.* 29.2 (2021), pp. 229–271. DOI: 10.3233/JCS-181270. URL: https://doi.org/10.3233/JCS-181270.

[RZ21] Carla Ràfols and Arantxa Zapico. *An Algebraic Framework for Universal and Updatable SNARKs*. Cryptology ePrint Archive, Paper 2021/590. https://eprint.iacr.org/2021/590. 2021. URL: https://eprint.iacr.org/2021/590.