An Algebraic Framework for Universal and Updatable zkSNARKs

No Author Given

No Institute Given

Abstract. We introduce Verifiable Subspace Sampling Arguments, a new information theoretic interactive proof system in which the prover shows that a vector has been sampled in a subspace according to the verifier's coins. We show that this primitive provides a unifying view that explains the technical core of most of the constructions of universal and updatable pairing-based zkSNARKs.

1 Introduction

We propose an algebraic framework that takes a step further in achieving modular constructions of universal and updatable SNARKs. We identify the technical core of previous work as instances of a *Verifiable Subspace Sampling Argument*(VSS).

Starting from Sonic [13], most works on universal and updatable zkSNARKs ([8,5,3]) follow the same print: they first build an information-theoretic proof system, and then compile it into a zkSNARK using a polynomial commitment as main ingredient. Our proposal is to decompose the information theoretic component of these works even further, and identify more prominently the underlying algebraic ideas. Thus, achieving zero-knowledge or details on the compilation step are out of the scope of this document.

We believe that a more modular, unified view of the contributions of existing constructions on SNARKs seems essential for a clearer understanding of the techniques behind these works. This, in turn should allow for a better comparison, more flexibility in combining the different methods, and give insights on current limitations.

Motivation. As an alternative to a trusted SRS, Groth et al. [11] proposed to use *updatable* zkSNARKs, in which the SRS can be updated by any party, non-interactively, and in a verifiable way, resulting in a properly generated structured reference string where the simulation trapdoor is unknown to all parties if at least one is honest. Further, the SRS is universal and can be used for arbitrary circuits up to a maximum given size.

Arithmetic Circuit Satisfiability can be reduced to a set of quadratic and affine constraints over a finite field. The quadratic ones are universal and can be easily proven in the pairing-based setting with a Hadamard product argument, the basic core of most zkSNARKS constructions starting from [9]. On the other hand, affine constraints are circuit-dependent, and it is a challenging task to efficiently prove them with a universal SRS [13,5,8,3,6,16,7,4].

In Groth et al. [11] they are proven via a very expensive subspace argument that requires a SRS quadratic in the circuit size and a preprocessing step that is cubic. Sonic [13], the first efficient, universal, and updatable SNARK, gives two different ways to prove the affine constraints, a fully succinct one (not so efficient) and another one in the amortized setting (very efficient). Follow-up work has significantly improved the efficiency in the fully succinct mode. In particular, Marlin [5], Plonk [8] and Lunar [3]. As it is mentioned in Plonk, underlying this progress there are new techniques to check correctness of the evaluation of certain bivariate polynomial that encodes affine constraints, but this common core is hard to isolate from other components, and there is not a clear algebraic interpretation of why this component is necessary.

From VSS to zkSNARKs We propose Verifiable Subspace Sampling arguments, an information-theoretic proof system through which two parties, prover and verifier, on input a field $\mathbb{F}$ and a matrix $\mathbf{W} \in \mathbb{F}^{Q \times m}$ agree on a polynomial $D(X)$ encoding a vector $\boldsymbol{d}$ in the rowspace of $\mathbf{W}$. The interesting part is that, even though the coefficients of the linear combination that define $\boldsymbol{d}$ are chosen by the verifier's coins, the latter does not need to perform a linear number of computations in order to trust $D(X)$. Instead, in a proving phase, the prover shows that $D(X)$ is correctly sampled.

With this algebraic formulation, it is immediate to see that a VSS argument can be used as a building block for an argument of membership in linear spaces. Basically, given a matrix $\mathbf{W}$, we can prove that some vector $\boldsymbol{y}$ is in the orhogonal space to the rows of $\mathbf{W}$ by sampling, after $\boldsymbol{y}$ is declared, a *sufficiently random* vector $\boldsymbol{d}$ in the row space of $\mathbf{W}$ and checking whether $\boldsymbol{d} \cdot \boldsymbol{y} = 0$. A VSS guarantees that the sampling process can be run by the verifier in sublinear time without sacrificing soundness.

Naturally, for building succinct proofs, instead of $\boldsymbol{y}, \boldsymbol{d}$, the argument uses polynomial encodings $Y(X)$ and $D(X)$ (which are group elements after the compilation step). Thus, we introduce an inner product argument in Section 3, which is specific to the case where the polynomials are encoded in the Lagrangian basis, but can be easily generalized to the monomial basis. The argument is a straightforward application of the sumcheck protocol of Aurora [1]. However, we contribute a generalized sumcheck (that works not only for multiplicative subgroups of finite fields), with a completely new security proof that relates it with polynomial evaluation at some fixed point v.

These building blocks can be put together as an argument for the language of Rank1 constraint systems. For efficiency, we stick to a variant recently proposed by Lunar [3], which is slightly simpler but still NP-complete. Our final construction can be instantiated with any possible choice of VSS schemes, so in particular it can essentially recover the construction of Marlin [5] and Lunar by isolating the VSS argument implicit in these works, or the amortized construction of Sonic. We show these two instantiations as examples. We hope that this serves to better identify the challenge behind building updatable and universal SNARKs, and allow for new steps in improving efficiency, as well as more easily combining the techniques.

In summary, our R1CS proof uses three main ingredients: an inner product, a Hadamard product and a VSS argument. We think this algebraic formulation is very clear, and also makes it easier to relate advances in universal and updatable SNARKS with other works that have used a similar language, for example, the inner product argument of [2], or the arguments of membership in linear spaces [12], or the arguments for linear algebra relations [10].

2 Preliminaries

2.1 Constraint Systems

Formally, we will construct an argument for the universal relation $\mathcal{R}'_{R1CS\text{-lite}}$, an equivalent of the relation $\mathcal{R}_{R1CS\text{-lite}}$ introduced in Lunar [3]. The latter is a simpler version of Rank 1 Constraint Systems, it is still NP complete and encodes circuit satisfiability in a natural way:

Definition 1. *(R1CS-lite) Let $\mathbb{F}$ be a finite field and $n, m, l \in \mathbb{N}$. We define the universal relation R1CS-lite as:*

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

As an equivalent formulation of this relation, we use the following:

$$\mathcal{R}'_{R1CS\text{-lite}} = \left\{ \begin{array}{c} (\mathsf{R}, \mathsf{x}, \mathsf{w}) := \big((\mathbb{F}, s, m, l, \mathbf{F}, \mathbf{G}), \boldsymbol{x}, (\boldsymbol{a}', \boldsymbol{b}')\big) : \\ \mathbf{F}, \mathbf{G} \in \mathbb{F}^{m \times m}, \boldsymbol{x} \in \mathbb{F}^{l-1}, \boldsymbol{a}', \boldsymbol{b}' \in \mathbb{F}^{m-l}, s = \max\{|\mathbf{F}|, |\mathbf{G}|\}, \\ \text{and for } \boldsymbol{a} := (1, \boldsymbol{x}, \boldsymbol{a}'), \boldsymbol{b} := (\mathbf{1}, \boldsymbol{b}') \\ \begin{pmatrix} \mathbf{I} & \mathbf{0} & -\mathbf{F} \\ \mathbf{0} & \mathbf{I} & -\mathbf{G} \end{pmatrix} \begin{pmatrix} \boldsymbol{a} \\ \boldsymbol{b} \\ \boldsymbol{a} \circ \boldsymbol{b} \end{pmatrix} = \mathbf{0} \end{array} \right\}.$$

To see they are equivalent, observe that, if in $\mathcal{R}'_{R1CS\text{-lite}}$ we define the vector $\boldsymbol{c} = \boldsymbol{a} \circ \boldsymbol{b}$, the linear equation reads as $\boldsymbol{a} = \mathbf{F}\boldsymbol{c}$ and $\boldsymbol{b} = \mathbf{G}\boldsymbol{c}$. A formal proof is a direct consequence of the proof that arithmetic circuit satisfiability reduces to R1CS-lite found in Lunar([3]).

2.2 Polynomial Holographic Proofs

In this paper, we use the notion of Polynomial Holographic Interactive Oracle Proofs (PHP), recently introduced by Campanelli et al. [3]. It is a refinement and quite similar to other notions used in the literature to construct SNARKs in a modular way, such as Algebraic Holographic Proofs (AHP) [5] or idealized polynomial protocols [8].

A proof system for a relation R is holographic if the verifier does not read the full description of the relation, but rather has access to an encoding of the statement produced by some holographic relation encoder, also called indexer, that outputs oracle polynomials. In all these models, the prover is restricted to send oracle polynomials or field elements, except that, for additional flexibility, the PHP model of [3] also lets the prover send arbitrary messages. In PHPs, the queries of the verifier are algebraic checks over the polynomials sent by the verifier, as opposed to being limited to polynomial evaluations as in AHPs.

The following definitions are taken almost verbatim from [3].

Definition 2. *A family of polynomial time computable relations $\mathcal{R}$ is field dependent if each relation $\mathsf{R} \in \mathcal{R}$, specifies a unique finite field. More precisely, for any pair $(\mathsf{x}, \mathsf{w}) \in \mathsf{R}$, x specifies the same finite field $\mathbb{F}_{\mathsf{R}}$ (simply denoted as $\mathbb{F}$ if there is no ambiguity).*

Definition 3 (Polynomial Holographic IOPs (PHP)). *A Polynomial Holographic IOP for a family of field-dependent relations $\mathcal{R}$ is a tuple $\mathsf{PHP} = (\mathsf{rnd}, \mathsf{n}, \mathsf{m}, \mathsf{d}, \mathsf{n}_e, \mathcal{I}, \mathcal{P}, \mathcal{V})$, where $\mathsf{rnd}, \mathsf{n}, \mathsf{m}, \mathsf{d}, \mathsf{n}_e : \{0,1\}^* \to \mathbb{N}$ are polynomial-time computable functions, and $\mathcal{I}, \mathcal{P}, \mathcal{V}$ are three algorithms that work as follows:*

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

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

  - *Polynomial checks: to verify that certain polynomial identities hold between the oracle polynomials and the messages sent by the prover. Let $\mathsf{n}^* = \sum_{k=0}^{\mathsf{rnd}(|\mathsf{R}|)} \mathsf{n}(k)$ and $\mathsf{m}^* = \sum_{k=0}^{\mathsf{rnd}(|\mathsf{R}|)} \mathsf{m}(k)$, and denote by $(p_1, \ldots, p_{\mathsf{n}^*})$ and $(\pi_1, \ldots, \pi_{\mathsf{n}^*})$ all the oracle polynomials (including the $\mathsf{n}(0)$ ones frrom the encoder) and all the messages sent by the prover. The verifier can specify a list of n_e tuples, each of the form $(G, v_1, \ldots, v_{\mathsf{n}^*})$, where $G \in \mathbb{F}[X, X_1, \ldots, X_{\mathsf{n}^*}, Y_1, \ldots, Y_{\mathsf{m}^*}]$ and every $v_k \in \mathbb{F}[X]$. Then a tuple $(G, v_1, \ldots, v_{\mathsf{n}^*})$ is satisfied if and only if $F(X) \equiv 0$ where*

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

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

Definition 4. PHP *is complete if for any triple $(\mathsf{R}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}$, the checks returned by $\mathcal{V}^{\mathcal{I}(\mathsf{R})}$ after interacting with the honest prover $\mathcal{P}(\mathsf{R}, \mathsf{x}, \mathsf{w})$, are satisfied with probability 1.*

Definition 5. PHP *is ϵ-sound if for every relation-instance tuple $(\mathsf{R}, \mathsf{x}) \notin \mathcal{L}(\mathcal{R})$ and polynomial time prover $\mathcal{P}^*$ we have*

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

Definition 6. PHP *is ϵ-zero-knowledge if there exists a PPT simulator $\mathcal{S}$ such that for every triple* $(\mathsf{R}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}$*, and every algorithm $\mathcal{V}$, the following random variables are within ϵ-statistical distance:*

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

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

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

The following definition captures de fact that zero-knowledge should hold even when the verifier has access to *a bounded amount* of evaluations of the polynomials that contain information about the witness. Let $\mathcal{Q}$ be a list of queries; we say that $\mathcal{Q}$ is (b, C)-bounded for $\mathsf{b} \in \mathbb{N}^{\mathsf{n}_p}$ and C is a PT algorithm if for every $i \in [\mathsf{n}_p]$, $|\{(i, z) : (i, z \in \mathcal{Q})\}| \leq \mathsf{b}_i$ and for all $(i, z) \in \mathcal{Q}$, $\mathsf{C}(i, z) = 1$.

Definition 7. PHP *is* (b, C)*-zero-knowledge if for every triple* $(\mathsf{R}, \mathsf{x}, \mathsf{w}) \in \mathcal{R}$*, and every* (b, C)*-bounded list $\mathcal{Q}$, the follow random variables are within ϵ statistical distance:*

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

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

Definition 8. PHP *is honest-verifier zero-knowledge with query bound* b *if there exists a PT algorithm* C *such that* PHP *is* (b, C)*-zero-knowledge and for all $i \in \mathbb{N}$, $Pr[\mathsf{C}(i, z) = 0]$ is negligible, where z is uniformly sampled over $\mathbb{F}$.*

3 Generalized Univariate Sumcheck

In this section, we present the core sub-arguments of our modular PHP for R1CS-lite. This is composed by a Hadamard and inner product arguments. For the latter, we revisit the sumcheck by Aurora [1]. As presented there, this argument allows to prove that the sum of the evaluations of a polynomial in some multiplicative[1] set $\mathbb{H}$ of a finite field $\mathbb{F}$ sum to 0. Because our aim is to create a framework as flexible as possible, we generalize the argument to arbitrary sets of $\mathbb{H}$, solving an open problem posed there. Additionally, we give a simpler proof of the same result by connecting the sumcheck to polynomial evaluation and other simple properties of polynomials.

Given some finite field $\mathbb{F}$, $\mathbb{H}$ is an arbitrary set of cardinal m. The set $\mathbb{H}$ is assumed to have some defined canonical order, and h_i refers to the ith element in this order. The ith Lagrangian basis polynomial associated to $\mathbb{H}$ is denoted by $\lambda_i(X)$, although for simplicity, occasionally it might also be indexed by the ith element of $\mathbb{H}$ as $\lambda_{\mathsf{h}_i}(X)$. The vector $\boldsymbol{\lambda}(X)$ is defined as $\boldsymbol{\lambda}(X)^\top = (\lambda_1(X), \ldots, \lambda_m(X))$. The vanishing polynomial of $\mathbb{H}$ will be denoted by $t(X)$. When $\mathbb{H}$ is a multiplicative subgroup, the following properties are known to hold:

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

for any $i = 1, \ldots, m$. This representation makes their computation particularly efficient: both $t(X)$ and $\lambda_i(X)$ can be evaluated in $O(\log m)$ field operations.

We prove a generalized sumcheck theorem below, and derive the sumcheck of Aurora as a corollary for the special case where $\mathbb{H}$ is a multiplicative subgroup. The intuition is simple: let $P_1(X), P_2(X)$ be two

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

polynomials congruent modulo $t(X)$, and the degree of $P_2(X)$ be at most $m-1$. Then, if we define $P_2(X) = \sum_{i=1}^{m} \lambda_i(X) P_1(\mathsf{h}_i)$, when $P_2(X)$ is evaluated at an arbitrary point $v \in \mathbb{F}$, $v \notin \mathbb{H}$, $P_2(v) = \sum_{i=1}^{m} \lambda_i(v) P_1(\mathsf{h}_i)$. Thus, $P_2(v)$ is "almost" (except for the constants $\lambda_i(v)$) the sum of the evaluations of $P_1(\mathsf{h}_i)$. Multiplying by a normalizing polynomial, we get rid of the constants and obtain a polynomial that when evaluated at v leads to the sum of any set of evaluations of interest. The sum will be zero if the normalized polynomial has a root at v.

Theorem 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 $P(X) \in \mathbb{F}[X]$, $\mathcal{S} \subset \mathbb{H}$, and any $v \in \mathbb{F}, v \notin \mathbb{H}$, $\sum_{s\in\mathcal{S}} P(s) = \sigma$ if and only if there exist polynomials $H(X) \in \mathbb{F}[X]$, $R(X) \in \mathbb{F}_{\leq m-2}[X]$ such that*

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

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

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

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

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

Lemma 1. *If $\mathcal{S} = \mathbb{H}$, then $v = 0$ and $N_{\mathbb{H},0}(X) = m$.*

Proof. Recall that, as $\mathbb{H}$ is a multiplicative subgroup, $\lambda_i(0) = 1/m$ for all $i = 1, \ldots, m$. Therefore, $N_{\mathbb{H},0}(X) = \sum_{i=1}^{m} \lambda_i(0)^{-1}\lambda_i(X) = m\sum_{i=1}^{m}\lambda_i(X) = m \mod t(X)$. Since $N_{\mathbb{H},0}(X)$, by definition, has degree at most $m-1$, we conclude that $N_{\mathbb{H},0}(X) = m$. $\square$

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

3.1 Application to Linear Algebra Arguments

Several works [1,5] have observed that R1CS languages can be reduced to proving a Hadamard product relation and a linear relation, where the latter consists on showing that two vectors $\boldsymbol{x}, \boldsymbol{y}$ are such that $\boldsymbol{y} = \mathbf{M}\boldsymbol{x}$, or equivalently, that the inner product of $(\boldsymbol{y}, \boldsymbol{x})$ with all the rows of $(\mathbf{I}, -\mathbf{M})$ is zero. When matrices and vectors are encoded as polynomials for succinctness, for constructing a PHP it is necessary to express these linear algebra operations as polynomial identities.

The following Theorem explicitly derives a polynomial identity that encodes the inner product relation from the univariate sumcheck. This connection in a different formulation is implicit in previous works [5,3].

Theorem 2 (Inner Product Polynomial Relation). *For some $k \in \mathbb{N}$, let $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k)$, $\boldsymbol{y}_i = (y_{ij})$, $\boldsymbol{d} = (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_k)$ be two vectors in $\mathbb{F}^{km}$, $\boldsymbol{y}_i, \boldsymbol{d}_i \in \mathbb{F}^m$, and $\mathbb{H}$ a multiplicative subgroup of $\mathbb{F}^*$ of order m. Then, $\boldsymbol{y} \cdot \boldsymbol{d} = 0$ 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:*

$$(\boldsymbol{Y}(X) \cdot \boldsymbol{D}(X)) N_{\mathbb{H},0}(X) = XR(X) + t(X)H(X), \tag{1}$$

where $Y_i(X)$ is a polynomial of arbitrary degree such that $Y_i(\mathsf{h}_j) = y_{ij}$ for all $j = 1, \ldots, m$, and $D_i(X) = \boldsymbol{d}_i^\top \boldsymbol{\lambda}(X)$.

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

$$\boldsymbol{Y}(X) \cdot \boldsymbol{D}(X) = \sum_{i=1}^{k} Y_i(X)D_i(X) = \sum_{i=1}^{k} (\boldsymbol{y}_i \circ \boldsymbol{d}_i)^\top \boldsymbol{\lambda}(X) = \big(\sum_{i=1}^{k} (\boldsymbol{y}_i \circ \boldsymbol{d}_i)^\top\big) \boldsymbol{\lambda}(X). \tag{2}$$

By Theorem 1, $\big(\sum_{i=1}^{k} (\boldsymbol{y}_i \circ \boldsymbol{d}_i)^\top\big)\boldsymbol{\lambda}(X)\big) N_{\mathbb{H},0}(X)$ can be written as in the right hand side of equation (1) if and only if the sum of the coordinates of $\sum_{i=1}^{k} (\boldsymbol{y}_i \circ \boldsymbol{d}_i)$ is 0. The jth coordinate of $\sum_{i=1}^{k} (\boldsymbol{y}_i \circ \boldsymbol{d}_i)$ is $\sum_{i=1}^{k} y_{ij} d_{ij}$, thus the sum of all coordinates is $\sum_{j=1}^{m} \sum_{i=1}^{k} y_{ij} d_{ij} = \boldsymbol{y} \cdot \boldsymbol{d}$, which concludes the proof. □

For the Hadamard product, the basic observation is that two polynomials $P_1(X), P_2(X)$ of arbitrary degree are identical in a set of interpolation points $\mathbb{H}$ if and only if there exists $H(X)$ such that $P_1(X) - P_2(X) = H(X)t(X)$. Thus, the polynomial identity

$$(\boldsymbol{a}^\top \boldsymbol{\lambda}(X))(\boldsymbol{b}^\top \boldsymbol{\lambda}(X)) - (\boldsymbol{c}^\top \boldsymbol{\lambda}(X)) = H(X)t(X), \tag{3}$$

holds for some $H(X)$ if and only if $\boldsymbol{a} \circ \boldsymbol{b} - \boldsymbol{c} = 0$. This Hadamard product argument is one of the main ideas behind the zkSNARK of Gentry et al. [9].

4 Verifiable Subspace Sampling: Definition and Implications

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

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

Definition 9 (Verifiable Subspace Sampling, VSS). *A verifiable subspace sampling argument over a field $\mathbb{F}$, defines some $Q, m \in \mathbb{N}$, a set of admissible matrices $\mathcal{M}$, a vector of polynomials $\boldsymbol{\lambda}(X) \in (\mathbb{F}[X])^m$, a coinspace $\mathcal{C}$, a sampling function $\mathsf{Smp} : \mathcal{C} \to \mathbb{F}^Q$, and a relation:*

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

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

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

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

– *When the proving phase is concluded, the verifier outputs a bit indicating acceptance or rejection.*

The vector $\boldsymbol{\lambda}(X) = (\lambda_1(X), \dots, \lambda_m(X))$ defines an encoding of vectors as polynomials: vector $\boldsymbol{v}$ is mapped to the polynomial $\boldsymbol{v}^\top \boldsymbol{\lambda}(X) = \sum_{i=1}^m v_i \lambda_i(X)$. When using a VSS scheme for constructing an argument of membership in linear spaces as in the next section, we choose a characterization of inner product that is compatible with the lagrangian polynomials. Thus, in this work, $\lambda_i(X)$ is defined as the ith Lagrangian polynomial associated to some multiplicative subgroup $\mathbb{H}$ of $\mathbb{F}$, this is why we have chosen to denote the encoding by $\boldsymbol{\lambda}(X)$. It makes sense to consider also VSS arguments for other polynomial encodings, e.g. the monomial basis or Laurent polynomials. In fact, the VSS scheme in the amortized setting described in Section 5.2 is an abstraction of the helped mode of Sonic, that was presented for the encoding with Laurent polynomials.

We require from a VSS scheme to satisfy the following security definitions:

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

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

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

Soundness. A verifiable subspace sampling argument $(\mathcal{I}_{\mathsf{VSS}}, \mathcal{P}_{\mathsf{VSS}}, \mathcal{V}_{\mathsf{VSS}})$ is ϵ-sound if for all $\mathbf{M}$, any polynomial time prover $\mathcal{P}^*_{\mathsf{VSS}} = (\mathcal{P}^*_{\mathsf{VSS}}, \mathcal{P}^*_{\mathsf{VSS}})$, and any $D(X)$ such that $(\mathsf{cns}, D(X)) \in \mathsf{R}_{\mathbf{M}}$:

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

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

Definition 10. *A VSS argument is ϵ-elusive kernel*[2] *if*

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

In practice, for most schemes, $\boldsymbol{s}$ is a vector of monomials or Lagrangian basis polynomials evaluated at some point $x = \mathsf{cns}$, and this property is an immediate application of Schwartz-Zippel lemma.

4.1 Linear Arguments from Verifiable Subspace Sampling

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

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

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

A standard way to prove that some vector $\boldsymbol{y}$ is in $\mathbf{W}^\perp$ is to let the verifier sample a *sufficiently random* vector $\boldsymbol{d}$ in the row space of matrix $\mathbf{W}$, and prove $\boldsymbol{y} \cdot \boldsymbol{d} = 0$. Naturally, the vector $\boldsymbol{y}$ must be declared before $\boldsymbol{d}$ is chosen. We follow this strategy to construct a PHP for $\mathcal{R}_{\mathsf{LA}}$, except that the vector $\boldsymbol{d}$ is sampled by the prover itself on input the coins of the verifier through a VSS argument.

As we have seen in Section 2.1, it is natural in our application to proving R1CS to consider matrices in blocks. Thus, in this section we prove membership in $\mathbf{W}^\perp$ where the matrix is written in k blocks of columns, that is, $\mathbf{W} = (\mathbf{W}_1, \dots, \mathbf{W}_k)$. The vectors $\boldsymbol{y}, \boldsymbol{d} \in \mathbb{F}^{km}$ are also written in blocks as $\boldsymbol{y}^\top = (\boldsymbol{y}_1^\top, \dots, \boldsymbol{y}_k^\top)$ and $\boldsymbol{d}^\top = (\boldsymbol{d}_1^\top, \dots, \boldsymbol{d}_k^\top)$.

[2] The name is inspired by the property of t-elusiveness of [14]

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

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

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

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

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

$$\boldsymbol{Y}(X) \cdot \boldsymbol{D}(X) = XR_t(X) + t(X)H_t(X). \tag{4}$$

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

$$\boldsymbol{Y}(X) \cdot \boldsymbol{D}(X) = XR_t(X) + t(X)H_t(X).$$

Fig. 1. Argument for proving membership in $\mathbf{W}^\perp$, parameterized by the polynomial encoding $\mathcal{E}_Y : W_Y \to \mathbb{F}[X]^k$, and the set $W_Y \subset \mathbb{F}^{km}$.

Theorem 3. *When instantiated using a VSS scheme with perfect completeness, and the encoding $\mathcal{E}_Y : W_Y \to \mathbb{F}[X]^k$ satisfies that, if $\mathcal{E}_Y(\boldsymbol{y}) = \boldsymbol{Y}(X)$, then $Y_j(\mathsf{h}_j) = y_{ij}$, the PHP of Fig. 1 has perfect completeness.*

Proof. By definition, $\boldsymbol{D}(X) = (\boldsymbol{s}^\top \mathbf{W}_1 \boldsymbol{\lambda}(X), \dots, \boldsymbol{s}^\top \mathbf{W}_k \boldsymbol{\lambda}(X))$, for $\boldsymbol{s} = \mathsf{Samp}(\mathsf{cns})$. Note that this is because the k instances of the VSS scheme are run in parallel and the same coins are used to sample each of the $\boldsymbol{d}_i$. Thus, $\boldsymbol{D}(X)$ is the polynomial encoding of $\boldsymbol{d} = (\boldsymbol{s}^\top \mathbf{W}_1, \dots, \boldsymbol{s}^\top \mathbf{W}_k) = \boldsymbol{s}^\top \mathbf{W}$. Therefore, if $\boldsymbol{y}$ is in $\mathbf{W}^\perp$, $\boldsymbol{d} \cdot \boldsymbol{y} = \boldsymbol{s}^\top \mathbf{W} \boldsymbol{y} = \mathbf{0}$. By the characterization of inner product, as explained in Section 3, this implies that polynomials $H_t(X), R_t(X)$ satisfying the verification equation exist. □

Theorem 4. *Let VSS be ϵ-sound and ϵ'-Elusive Kernel, $\mathcal{E}_Y : W_Y \to \mathbb{F}[X]^k$ an encoding such that if $\mathcal{E}_Y(\boldsymbol{y}) = \boldsymbol{Y}(X)$, $Y_j(\mathsf{h}_j) = y_{ij}$. Then, for any polynomial time adversary $\mathcal{A}$ against the soundness of PHP of Fig. 1:*

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

Proof. Let $\boldsymbol{Y}^*(X) = (Y_1^*(X), \dots, (Y_k^*(X))$ be the output of a cheating $\mathcal{P}_{\mathsf{LA}}^*$ and $\boldsymbol{y}^*$ the vector such that $\boldsymbol{Y}^*(X) = ((\boldsymbol{y}_1^*)^\top \boldsymbol{\lambda}(X), \dots, (\boldsymbol{y}_k^*)^\top \boldsymbol{\lambda}(X))$.

As a direct consequence of Theorem 2, $\boldsymbol{Y}^*(X) \cdot \boldsymbol{D}(X) = XR_t(X) + t(X)H_t(X)$ only if $\boldsymbol{y}^* \cdot \boldsymbol{d} = 0$, where $\boldsymbol{d}$ is the unique vector $\boldsymbol{d}$ such that $\boldsymbol{D}(X) = (\boldsymbol{d}_1^\top \boldsymbol{\lambda}(X), \dots, \boldsymbol{d}_k^\top \boldsymbol{\lambda}(X))$.

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

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

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

Extension to other polynomial encodings. As mentioned, the construction is specific to the polynomial encoding defined by Lagrangian polynomials. However, the only place where this plays a role is in the check of equation (4). Now, if the polynomial encoding $\boldsymbol{\lambda}(X)^\top$ associated to the VSS for $\mathbf{W}$ was set to be for instance the monomial basis, i.e. $\boldsymbol{\lambda}(X)^\top = (1, X, \dots, X^m)$, the argument can be easily modified to still work. It suffices to choose the "reverse" polynomial encoding for $\boldsymbol{y}$, that is define $\boldsymbol{Y}(X) = (\boldsymbol{y}_1^\top \tilde{\boldsymbol{\lambda}}(X), \dots, \boldsymbol{y}_k^\top \tilde{\boldsymbol{\lambda}}(X))$, where $\tilde{\boldsymbol{\lambda}}(X)^\top = (X^m, \dots, X, 1)$, and require the prover to find $R_t(X), H_t(X)$, with $R_t(X)$ of degree at most $m-2$ such that:

$$\boldsymbol{Y}(X) \cdot \boldsymbol{D}(X) = R_t(X) + X^m H_t(X). \tag{5}$$

Indeed, observe that this check guarantees that $\boldsymbol{Y}(X) \cdot \boldsymbol{D}(X)$ does not have any term of degree exactly $m-1$, and the term of degree $m-1$ is exactly $\sum_{i=1}^m \boldsymbol{y}_i \cdot \boldsymbol{d}_i = \boldsymbol{y} \cdot \boldsymbol{d}$.

4.2 R1CS-lite from Linear Arguments

In this section we give a PHP for R1CS-lite by combining our linear arguments with other well known techniques.

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

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

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

for $t_l(X) = \prod_{i=1}^{\ell} (X - \mathsf{h}_i)$. It outputs $(A'(X), B'(X))$.

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

Fig. 2. PHP for $\mathcal{R}_{\mathsf{lite}}$ from PHP for $\mathcal{R}_{\mathsf{LA}}$. The PHP for $\mathcal{R}_{\mathsf{LA}}$ should be instantiated for $W_Y = \{(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{a} \circ \boldsymbol{b}) : \boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}^m\}$, $\mathcal{E}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{a} \circ \boldsymbol{b}) = (\boldsymbol{a}^\top \boldsymbol{\lambda}(X), \boldsymbol{b}^\top \boldsymbol{\lambda}(X), (\boldsymbol{a}^\top \boldsymbol{\lambda}(X))(\boldsymbol{b}^\top \boldsymbol{\lambda}(X)))$.

Theorem 5. *When instantiated with a complete and sound linear argument, the PHP of Fig. 2 satisfies completeness and soundness.*

Proof. Completeness follows directly from the definition of $A'(X), B'(X), A(X), B(X)$ and completeness of the linear argument. Soundness holds if the linear argument is sound as well, because $\mathcal{V}_{\mathsf{lite}}$ accepts if $\mathcal{V}_{\mathsf{LA}}$ accepts, meaning $\mathbf{W}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{a} \circ \boldsymbol{b})^\top = 0$ and $\mathcal{R}_{\mathsf{lite}}$ holds. □

4.3 Combining VSS schemes

Since a VSS outputs a linear combination of the rows of input matrix $\mathbf{M}$, different instances of a VSS scheme can be easily combined with linear operations. More precisely, given a matrix $\mathbf{M}$ that can be written as $\begin{pmatrix}\mathbf{M}_1\\ \mathbf{M}_2\end{pmatrix}$, we can use a different VSS arguments for each $\mathbf{M}_i$. If these instances output, respectively, polynomials $D_1(X), D_2(X)$, then we can define a VSS argument for $\mathbf{M}$ that in the sampling phase outputs $D_1(X) + D_2(X)$, and in the proving phase it sends and proves the correctness of each polynomial individually (if the verification of these arguments is compatible, some elements might be saved by batching their proofs).

Since all current constructions of VSS arguments have limitations in terms of the types of matrices they apply to, this opens the door to decomposing the matrix of constraints into different blocks that admit efficient VSS arguments. In other words, when considering the limitations of current VSS schemes, combinations of different schemes should also be taken into account.

A different alternative combines instances of the same VSS argument with the same coins, and proves correctness in a single step by batching these instances. More precisely, if $\mathbf{M} = \begin{pmatrix}\mathbf{M}_1\\ \mathbf{M}_2\end{pmatrix}$, and a certain VSS scheme samples $s = \mathsf{Smp}(\mathsf{cns})$ and defines $D_1(X) = \boldsymbol{s}^\top\mathbf{M}_1\boldsymbol{\lambda}(X)$ and $D_2(X) = \boldsymbol{s}^\top\mathbf{M}_2\boldsymbol{\lambda}(X)$, then $D_1(X) + zD_2(X)$ is a polynomial encoding of some vector in row span of $\mathbf{M}$, more precisely, $D_1(X)+zD_2(X) = (\boldsymbol{s}^\top, z\boldsymbol{s}^\top)\mathbf{M}\boldsymbol{\lambda}(X)$. If the correctness of $D_1(X), D_2(X)$ is checked separately, and $\mathbf{M}_1, \mathbf{M}_2$ are both admissible matrices for this VSS scheme, this combination can be done generically, and results in a new VSS with the sampling function $\mathsf{Smp}(\mathsf{cns}, z) = (\boldsymbol{s}^\top, z\boldsymbol{s}^\top)$ a different sampling function.

To save communication complexity, however, it would be interesting to let the prover just send $D_1(X)+zD_2(X)$, and check the correctness together. However, this cannot be proven in general. It is necessary that: a) the polynomials computed by the indexer of the VSS for $\mathbf{M}_1, \mathbf{M}_2$ can be combined upon receiving the challenge z to the VSS indexer polynomials of $\mathbf{M}_1 + z\mathbf{M}_2$, b) that $\mathbf{M}_1 + z\mathbf{M}_2$ is an admissible matrix for this VSS. For instance, if $\mathbf{M}_1, \mathbf{M}_2$ has K non-zero entries each, and the admissible matrices of a VSS instance must have at most K non-zero entries, then $\mathbf{M}_1 + z\mathbf{M}_2$ is not generally an admissible matrix. We will be using this optimization for our final PHP for sparse matrices, and we will see there that these conditions are met in this case.

5 VSS Instantiations

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

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

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

In Sonic, $\boldsymbol{\alpha}(Y), \boldsymbol{\beta}(X)$ are vectors of Laurent polynomials. In Marlin, Lunar and in this work, we set $\boldsymbol{\alpha}(Y) = \boldsymbol{\beta}(X) = \boldsymbol{\lambda}(X)$. For the proving phase, the common strategy is to follow the general template introduced in Sonic: the verifier samples a challenge $y \in \mathbb{F}$, checks that $D(y)$ is equal to σ, and $\sigma = P(y,x)$ (through what is called a signature of correct computation, as in [15]). This proves that $D(X) = P(X,x)$.

In Section 5.1 we introduce the implicit VSS scheme in Lunar [3], that follows Marlin [5] but works for R1CS-lite instead of R1CS. Note that when we instantiate the PHP of figure 2 with this VSS argument, the result is the PHP in [3]. In Section 5.2 we present the amortized VSS argument of Sonic [13], applied to a polynomial $P(X,Y)$ as above, with $\boldsymbol{\alpha}(Y) = \boldsymbol{\beta}(X) = \boldsymbol{\lambda}(X)$ to fit with our inner product argument. Both constructions are presented for one block of matrix $\mathbf{W}$, i.e., for a matrix $\mathbf{M} \in \mathbb{F}^{m\times m}$.

5.1 The VSS scheme of Marlin and Lunar

We consider two disjoint sets of roots of unity, $\mathbb{H}, \mathbb{K}$ of degree m and K, respectively. For $\mathbb{H}$ we use the notation defined in Section 3. The elements of $\mathbb{K}$ are assumed to have some canonical order, and we use k_ℓ for the ℓth element in $\mathbb{K}$, $\mu_\ell(X)$ for the ℓth Lagrangian interpolation polynomial associated to $\mathbb{K}$, and $u(X)$ for the vanishing polynomial. Assuming the non-zero entries are ordered, this matrix can be represented, as proposed in Marlin, by three functions $\mathsf{v} : \mathbb{K} \to \mathbb{F}$, $\mathsf{r} : \mathbb{K} \to [m]$, $\mathsf{c} : \mathbb{K} \to [m]$ such that $P(X,Y) = \sum_{\ell=1}^{K} \mathsf{v}(\mathsf{k}_\ell)\lambda_{\mathsf{r}(\mathsf{k}_\ell)}(Y)\lambda_{\mathsf{c}(\mathsf{k}_\ell)}(X)$, where the ℓth non-zero entry is $\mathsf{v}(\mathsf{k}_\ell) = m_{\mathsf{r}(\mathsf{k}_\ell),\mathsf{c}(\mathsf{k}_\ell)}$. If the matrix has less than K non-zero entries $\mathsf{v}(\mathsf{k}_\ell) = 0$, for $\ell = |\mathbf{M}| + 1, \ldots, K$, and $\mathsf{r}(\mathsf{k}_\ell), \mathsf{c}(\mathsf{k}_\ell)$ are defined arbitrarily.

Offline Phase: Outputs $\mathcal{W}_{\mathsf{VSS}} = \{\mathsf{c}(X), \mathsf{r}(X), \mathsf{cr}(X), \mathsf{c}'(X), \mathsf{r}'(X), \mathsf{cr}'(X), \mathsf{vcr}(X)\}$, where:

$$\mathsf{cr}(X) = \sum_{\ell=1}^{K} \mathsf{c}(\mathsf{k}_\ell)\mathsf{r}(\mathsf{k}_\ell)\mu_\ell(X), \qquad \mathsf{c}'(X) = X\mathsf{c}(X), \qquad \mathsf{r}'(X) = X\mathsf{r}(X), \qquad \mathsf{cr}'(X) = X\mathsf{cr}(X),$$

$$\mathsf{vcr}(X) = \sum_{\ell=1}^{K} \mathsf{v}(\mathsf{k}_\ell)\mathsf{cr}(\mathsf{k}_\ell)\mu_\ell(X)$$

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

$\mathsf{ProveSampling}$: $\mathcal{V}_{\mathsf{VSS}}$ sends $y \leftarrow \mathbb{F}$. $\mathcal{P}_{\mathsf{VSS}}$ calculates $\sigma = D(y)$ and outputs $R_u(X), H_u(X)$ such that:

$$\Big(\frac{\sigma}{K} + XR_u(X)\Big) n^2\big(x - \mathsf{r}(X)\big)\big(y - \mathsf{c}(X)\big) = \mathsf{vcr}(X)t(x)t(y) + H_u(X)u(X)$$

Decision Phase: Accept if and only if (1) $\deg(R_u) \leq K - 2$, (2) $D(y) = \sigma$, and (3)

$$\frac{\sigma}{K}n^2\big(xy + \mathsf{cr}(X) - x\mathsf{c}(X) - y\mathsf{r}(X)\big) + R_u(X)n^2\big(xyX + \mathsf{cr}'(X) - x\mathsf{c}'(X) - y\mathsf{r}(X)\big) - \mathsf{vcr}(X)t(x)t(y) - H_u(X)u(X) = 0$$

Fig. 3. VSS argument in [3].

5.2 Amortized VSS by Sonic

In this section we present a VSS argument that works only in the *amortized* setting as considered in Sonic ([13]). The construction is basically the protocol in the named work, but for a bivariate polynomial in the Lagrangian basis rather than the basis of monomials.

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

Still, in the $\mathsf{ProveSampling}$ algorithm, the verifier has oracle access to a set of polynomials $\mathcal{D} = \{D_1(X), \ldots, D_t(X)\}$ where each $D_s(X)$ is the output of a different execution of $\mathsf{Sampling}$ with verifier's challenge x_s. Following the original definition, the verifier also has oracle access to the polynomials outputted by $\mathcal{P}_{\mathsf{VSS}}$ (instantiated by what in Sonic is called a helper) in $\mathsf{ProveSampling}$.

6 Future Directions

The aim of this proposal is to start considering modular constructions of universal and updatable zkSNARKs and isolate the essence of these arguments, which in all the cases is satisfiability of linear constraints. In this direction, we

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

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

Decision Phase: Calculate $\tilde{P}(X) = P(u_1, X)$ and accept if and only if $\tilde{D}(X) = \tilde{P}(X)$ and, for every $\{x_s\}_{s=1}^{t}$, $\tilde{D}(x_s) = D_s(u_1)$.

Fig. 4. Amortized VSS scheme from [13].

hope the framework presented above will help in the understanding and comparison of existing and new techniques, in order to facilitate improvements.

Plonk [8] breaks down satisfiability of linear constraint in two satisfiability problems: additive gates and self permutation of the witness. The first one can be proven by the most trivial VSS, where the matrices representing the constraints are formed by multiples of the canonical vectors. In the next step, we aim at formulating Plonk in this language as well.

We also think it is worth discussing as part of the standardization effort if it would make sense to identify a set of algebraic relations for which proof systems should be designed. As an example, Bootle et al. [2] constructed an inner product argument in the DLOG setting in the URS model. In this work we build an inner product argument in the SRS model, which essentially is the lincheck argument of Aurora. Fixing the terminology for referring to the most common algebraic relations could also facilitate the comparison between different works, specially for languages with a strong arithmetic component such as R1CS.

References

1. E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent succinct arguments for R1CS. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Heidelberg, May 2019. 2, 4, 5
2. J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357. Springer, Heidelberg, May 2016. 2, 12
3. M. Campanelli, A. Faonio, D. Fiore, A. Querol, and H. Rodríguez. Lunar: a toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. Cryptology ePrint Archive, Report 2020/1069, 2020. `https://eprint.iacr.org/2020/1069`. 1, 2, 3, 5, 10, 11
4. M. Campanelli, D. Fiore, and A. Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In L. Cavallaro, J. Kinder, X. Wang, and J. Katz, editors, *ACM CCS 2019*, pages 2075–2092. ACM Press, Nov. 2019. 1
5. A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020. 1, 2, 3, 5, 10
6. V. Daza, C. Ràfols, and A. Zacharakis. Updateable inner product argument with logarithmic verifier and applications. In A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 527–557. Springer, Heidelberg, May 2020. 1
7. A. Gabizon. AuroraLight: Improved prover efficiency and SRS size in a sonic-like system. Cryptology ePrint Archive, Report 2019/601, 2019. `https://eprint.iacr.org/2019/601`. 1
8. A. Gabizon, Z. J. Williamson, and O. Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. `https://eprint.iacr.org/2019/953`. 1, 3, 12
9. R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct NIZKs without PCPs. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013. 1, 6

10. J. Groth. Linear algebra with sub-linear zero-knowledge arguments. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 192–208. Springer, Heidelberg, Aug. 2009. 2
11. J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, and I. Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, Aug. 2018. 1
12. C. S. Jutla and A. Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 1–20. Springer, Heidelberg, Dec. 2013. 2
13. M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In L. Cavallaro, J. Kinder, X. Wang, and J. Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, Nov. 2019. 1, 10, 11, 12
14. P. Morillo, C. Ràfols, and J. L. Villar. The kernel matrix Diffie-Hellman assumption. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 729–758. Springer, Heidelberg, Dec. 2016. 7
15. C. Papamanthou, E. Shi, and R. Tamassia. Signatures of correct computation. In A. Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 222–242. Springer, Heidelberg, Mar. 2013. 10
16. S. Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. Cryptology ePrint Archive, Report 2019/550, 2019. `https://eprint.iacr.org/2019/550`. 1