Counting Vampires: From Univariate Sumcheck to Updatable ZK-SNARK*

Tuesday 29th March, 2022, 18:46

Helger Lipmaa[1], Janno Siim[1], and Michał Zając[2]

[1] Simula UiB, Bergen, Norway
[2] Nethermind, London, UK

Abstract. We propose a univariate sumcheck argument $\mathfrak{Count}$ of essentially optimal communication efficiency of one group element. While the previously most efficient univariate sumcheck argument of Aurora is based on polynomial commitments, $\mathfrak{Count}$ is based on inner-product commitments. We use $\mathfrak{Count}$ to construct a new pairing-based updatable and universal zk-SNARK $\mathfrak{Vampire}$ with the shortest known argument length (five group elements and two integers) for NP. In addition, $\mathfrak{Vampire}$ uses the aggregated polynomial commitment scheme of Boneh *et al.* Differently from the previous (efficient) work, both $\mathfrak{Count}$ and $\mathfrak{Vampire}$ have an updatable SRS that consists of *non-consequent* monomials.

Keywords: Aggregatable polynomial commitment, inner-product commitment, univariate sumcheck, updatable and universal zk-SNARK

1 Introduction

Zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs) [Gro10,Lip12,GGPR13,PHGR13] are zero-knowledge argument systems for NP with succinct argument length and efficient verification. In many applications, one can describe the desired NP language instance as an instance $\mathcal{R}$ of the rank-1 constraint system (R1CS) [GGPR13], and the task of the verifier is to check that $\mathcal{R}$ is satisfied on the partially-public input. Zk-SNARKs are immensely popular due to applications in verifiable computation and cryptocurrencies [BCG+14].

Non-interactive zero-knowledge (NIZK) arguments, and thus also zk-SNARKs, are impossible in the plain model. To overcome this, one gives all parties access to a trusted common reference string (CRS). The most efficient zk-SNARKs have a relation-specific structured CRS (SRS). That is, they assume that there exists a trusted third party who, given the description of $\mathcal{R}$ as an input, generates an SRS $\mathsf{srs}_{\mathcal{R}}$. The most efficient zk-SNARK by Groth [Gro16] for R1CS with relation-specific SRS has an argument that consists of only three group elements. This value is close to the lower bound of two group elements [Gro16].

* Most of the work has been done when Janno Siim was employed by the University of Tartu and Michał Zając was employed by Clearmatics Technologies.

A significant practical downside of such "non-universal" SNARKs is that one has to construct and use a new SRS for every instance of the constraint system. It has spurred a large amount of effort to design universal zk-SNARKs, i.e., zk-SNARKs with an SRS that only depends on an upper bound on $\mathcal{R}$'s size. In addition, it is vitally important to decrease the amount of trust in the trusted third party. To this end, one strives to design *updatable and universal zk-SNARKs* [GKM+18,MBKM19], where the universal SRS is updated sequentially by several parties such that the soundness holds if at least one of the updaters is honest. For brevity, from now on, by "updatable" we will mean "updatable and universal".

Plonk [GWC19] and Marlin [CHM+20] are the first genuinely efficient universal zk-SNARKs. Marlin (and most of the subsequent updatable zk-SNARKs) works for sparse R1CS instances, where the underlying matrices contain a linear (instead of quadratic) number of non-zero elements. Chiesa *et al.* [CHM+20] first defines an information-theoretic model, algebraic holographic proof (AHP). An AHP is an interactive protocol, where at each step, the prover forwards oracles to some polynomials, and the verifier sends to the prover random field elements. In the end, the verifier queries the polynomial oracles at some chosen points and performs some low-degree tests. Polynomial oracles are usually implemented using polynomial commitments [KZG10]. After that, [CHM+20] proposes a new AHP for sparse R1CS, and then compiles it to a zk-SNARK named Marlin.

The construction of Marlin relies crucially on a univariate sumcheck. A sumcheck argument aims to prove that the given polynomial sums to the given value over the given domain. The first sumcheck arguments [LFKN90] were for multivariate polynomials but small domains. Ben-Sasson *et al.* [BCR+19] proposed a univariate sumcheck argument for large domains[3] and used it to construct a new zk-SNARK Aurora. Suppose the domain is a multiplicative subgroup of the given finite field. In that case, Aurora's sumcheck argument requires the prover to forward two different polynomial oracles and use a low-degree test on one of the polynomials.

Lunar [CFF+21] improves on Marlin in several aspects. First, they define PHPs, a generalized version of AHPs. In particular, they note that instead of opening all polynomial commitments, one can often operate directly on the polynomial commitments, thus obtaining better efficiency. Second, they define a simpler version of R1CS called R1CSLite, with one of the three characterizing matrices being the identity matrix. Moreover, they provide a more fine-grained analysis of the zero-knowledge property and implement several additional optimizations.

Finally, Basilisk [RZ21a] gains additional efficiency by using a different technique to obtain zero-knowledge and constructing a "free" low-degree test. In addition, [RZ21a] constructs even more efficient zk-SNARKs for somewhat more limited constraint systems. Both Lunar and Basilisk introduce new theoretical

[3] In fact, they constructed two different univariate sumchecks, for affine subspaces and multiplicative subgroups. As in Marlin, Lunar, and Basilisk, we refer to the latter sumcheck since it is more efficient.

Table 1. Argument length comparison of some known updatable zk-SNARKs.

Scheme	Argument length		Arithmetization
	Elements	Bits	
Updatable, universal zk-SNARKs			
Sonic [MBKM19]	$20\|\mathbb{G}_1\| + 16\|\mathbb{F}\|$	11776	[BCC+16] constraints
Marlin [CHM+20]	$13\|\mathbb{G}_1\| + 8\|\mathbb{F}\|$	7040	R1CS, sparse matrices
Plonk [GWC19]	$7\|\mathbb{G}_1\| + 7\|\mathbb{F}\|$	4480	Plonk constraints
LunarLite [CFF+21]	$10\|\mathbb{G}_1\| + 2\|\mathbb{F}\|$	4352	R1CSLite, sparse matrices
Basilisk [RZ21a]	$10\|\mathbb{G}_1\| + 3\|\mathbb{F}\|$	4608	R1CSLite, sparse matrices
Basilisk [RZ21a]	$8\|\mathbb{G}_1\| + 4\|\mathbb{F}\|$	4096	Plonk constraints
Basilisk (full version, [RZ21b])	$6\|\mathbb{G}_1\| + 2\|\mathbb{F}\|$	2816	Weighted R1CS with bounded fan-out
Vampire (this work)	$5\|\mathbb{G}_1\| + 2\|\mathbb{F}\|$	2432	R1CSLite, sparse matrices
Non-universal zk-SNARKs (relation-specific SRS)			
Groth16 [Gro16]	$2\|\mathbb{G}_1\| + 1\|\mathbb{G}_2\|$	1536	R1CS

frameworks; e.g., Basilisk introduces checkable sumcheck sampling (CSS) arguments as a separate primitive. For simplicity (of reading), we opted not to use such frameworks in the context of the current paper.

In Table 1, we overview the argument lengths of most efficient updatable zk-SNARKs. Here, $|X|$ denotes the representation length of an element from X in bits, given the BLS12-381 curve [Bow17], with $|\mathbb{G}_1| = 384$, $|\mathbb{G}_2| = 768$, and $|\mathbb{F}| = 256$. Thus, even the most efficient updatable zk-SNARK has an approximately two times longer argument than the non-universal zk-SNARK of [Gro16].

Moreover, Groth16 works for QAP [GGPR13] (i.e., full R1CS), while the most efficient variant of Basilisk works for instances of R1CSLite where the relation-defining matrices are limited to have a small constant number of elements per row (this corresponds to arithmetic circuits of bounded fan-out).

1.1 Our Contributions

The current paper has four related contributions of independent interest:

(1) The use of SRSs that consist of *non-consequent* monomials. Such a setting does not run against the impossibility result of [GKM+18] yet allows us to construct more efficient updatable arguments. In particular, the inefficient zk-SNARK of [GKM+18] used a non-consequent monomial SRS, while all efficient updatable zk-SNARKs seem to use consequent monomial SRSs.

(2) The combined use of polynomial commitments and inner-product commitments in the sumcheck and updatable zk-SNARK design. The use of polynomial commitment schemes in zk-SNARKs has dramatically increased their popularity, and we hope the same will happen with inner-product commit-

ments. In particular, ILV inner-product commitments [ILV11] use a SRS made of non-consequent monomials.[4]

(3) A new updatable univariate sumcheck argument $\mathfrak{Count}$ that uses inner-product commitments to achieve optimal computation complexity of a single group element. Since sumchecks are used in many different zk-SNARKs (and elsewhere, [BCS21]), we believe $\mathfrak{Count}$ will have wider interest.

(4) A new updatable and universal zk-SNARK $\mathfrak{Vampire}$ for sparse R1CSLite with the smallest argument length among all known updatable zk-SNARKs for NP-complete languages. (See Table 1.) $\mathfrak{Vampire}$ uses $\mathfrak{Count}$ and thus SRSs of non-consequent monomials.

1.2 Our Techniques

Non-Consequent Monomial SRSs. Groth *et al.* [GKM+18] proved that the SRS of an updatable zk-SNARK cannot contain non-monomial polynomials. Moreover, the SRS's correctness must be verifiable. For example, if the SRS contains[5] $[1, \sigma, \sigma^3, \sigma^4]_1 \in \mathbb{G}_1^4$, it must also contain $[\sigma, \sigma^2]_2 \in \mathbb{G}_2^2$, so that one can verify the consistency of the SRS elements by using pairing operations. We observe that $[\sigma^2]_1$ does not have to belong to the SRS, and thus, an updatable SRS may contain holes. Similarly, the SRS can have multivariate monomials. On the other hand, most of the known updatable zk-SNARKs ([GKM+18] being an exception, but their zk-SNARK is inefficient) use SRSs that consist of consequent univariate monomials only, i.e., are of the shape $([\sigma^i]_1, [\sigma^i]_2)_{i=0}^m$ for some m.

One reason why efficient updatable zk-SNARKs use a consequent monomial SRS is their reliance on polynomial commitment schemes like KZG [KZG10] that have such SRSs. While other polynomial commitment schemes are known, up to our knowledge, no efficient one relies on non-consequent monomials SRSs. In particular, AHP [CHM+20] and PHP [CFF+21] model polynomial commitments as polynomial oracles and allow the parties to perform operations (e.g., queries to committed oracles and low-degree tests) related explicitly to such oracles. Low-degree tests model consequent monomial SRSs: a committed polynomial is a degree-$\leq m$ polynomial iff it belongs to the span of X^i for $i \leq m$.

It is known how to use non-consequent monomial SRSs to efficiently construct protocols like broadcast encryption [BGW05] and inner-product commitments [ILV11]. We use non-consequent monomial SRSs in the context of sumchecks and updatable zk-SNARKs. For simplicity, we will not define an information-theoretic model. We only mention two possible approaches that both have their limitations. First, the pairing-based setting can be modeled as linear interactive proofs (LIPs, [BCI+13]) or non-interactive LIPs (NILPs, [Gro16]). However, either model has to be tweaked to our setting: namely, we allow the generation of updatable SRS for multi-round protocols, with the restrictions

[4] Inner-product commitments and arguments are commonly used in the zk-SNARK design. However, the way we use them is markedly different from the prior work.

[5] We rely on the pairing-based setting and use the by now standard additive bracket notation, see Section 2 for more details.

natural in such a setting (e.g., one can efficiently "span test" that a committed element is in the span of the SRS). Such a model is tailor-fit to pairings and might not be suitable in other algebraic settings. Second, one can generalize PHPs [CFF+21] by adding an abstract model of inner-product commitment schemes and allowing for span tests. Such a model is independent of the algebraic setting but would restrict one a priori to a limited number of cryptographic tools (polynomial and inner-product commitment schemes), with a need to redefine the model when more tools are discovered to be helpful.

We have chosen to remain agnostic on this issue by defining new arguments without an intermediate information-theoretic model.

New Univariate Sumcheck Argument $\mathfrak{Count}$. Let $\mathbb{F}$ be a finite field and let $\mathbb{H} \subset \mathbb{F}$ be a fixed multiplicative subgroup $\mathbb{H}$. In a *univariate sumcheck argument* (for multiplicative subgroups), the prover convinces the verifier that the committed polynomial $f(X) \in \mathbb{F}[X]$ sums to the given integer value v_M over $\mathbb{H}$.

Let $n_h := |\mathbb{H}|$, $\mathcal{Z}_{\mathbb{H}}(X) := \prod_{\chi \in \mathbb{H}}(X - \chi)$ be the vanishing polynomial of $\mathbb{H}$, and $f \in \mathbb{F}_{\leq n_h - 1}[X]$ be any polynomial with $\deg f \leq n_h - 1$. Aurora's sumcheck [BCR+19] relies on the simple fact that $\sum_{\chi \in \mathbb{H}} f(\chi) = n_h f(0)$. Thus, for any $f \in \mathbb{F}[X]$ of arbitrarily large degree, $\sum_{\chi \in \mathbb{H}} f(\chi) = v_f$ iff there exist polynomials $R, Q \in \mathbb{F}[X]$, such that (1) $\deg R \leq n_h - 2$, and (2) $f(X) = XR(X) + Q(X)\mathcal{Z}_{\mathbb{H}}(X)$. In a cryptographic implementation of Aurora's sumcheck argument, the prover uses the KZG polynomial commitment scheme [KZG10] to commit to R and Q; this means the communication of two group elements. In addition, the prover uses a low-degree test to convince the verifier that (1) holds.

We construct a new sumcheck argument $\mathfrak{Count}$ based on the ILV inner-product commitment [ILV11]. Let us first recall (non-randomized) ILV. In ILV, the non-consequent monomial SRS contains $([(\sigma^i)_{i=0:i\neq n+1}^{2n}]_1, [(\sigma^i)_{i=0}^{n}]_2)$, where σ is a trapdoor and n is a large integer. The prover commits to a vector $\boldsymbol{\mu} \in \mathbb{Z}_p^n$ as $[\mu(\sigma)]_1 \leftarrow \sum_{j=1}^{n} \mu_j [\sigma^j]_1$. When the verifier outputs a vector $\boldsymbol{\nu} \in \mathbb{Z}_p^n$, the prover returns the inner product $v \leftarrow \boldsymbol{\mu}^\top \boldsymbol{\nu}$ together with a short argument (a single group element $[\mathsf{op}]_1$) that v is correctly computed. ILV's security relies on the fact that $[\sigma^n]_1$ is not in the SRS.

We present an alternative extension of the equality $\sum_{\chi \in \mathbb{H}} f(\chi) = n_h f(0)$, for small-degree f, to the case when $d = \deg f$ is arbitrarily large. Namely, we prove that if $f(X) = \sum_{i=0}^{d} f_i X^i \in \mathbb{F}_{\leq d}[X]$, then $\sum_{\chi \in \mathbb{H}} f(\chi) = n_h \cdot (\sum_{i=0}^{\lfloor d/n_h \rfloor} f_{n_h i})$. (See Lemma 1.) Alternatively, $\sum_{\chi \in \mathbb{H}} f(\chi) = v_f$ iff $\boldsymbol{f}^\top \boldsymbol{s} = v_f$, where $\boldsymbol{f} = (f_i)$ and $\boldsymbol{s}$ is a Boolean vector that has ones in positions $n_h i$ for $i \leq \lfloor n/n_h \rfloor$.

In the new univariate sumcheck argument $\mathfrak{Count}$ for $f \in \mathbb{F}_{\leq d}[X]$, the prover first ILV-commits to $\boldsymbol{f}$ and then ILV-opens the commitment to $\boldsymbol{f}^\top \boldsymbol{s}$. Thus, the prover has to output one ILV commitment (one group element) instead of two polynomial commitments (two group elements). Moreover, there is no need for a low-degree test, making $\mathfrak{Count}$ even more efficient. In addition, in our case $\boldsymbol{s}$ has a small constant number of non-zero elements; thus, the prover's computation is linear in both field operations and group operations. An explicit cost of this technique is that the SRS becomes larger: if the SRS, without $\mathfrak{Count}$, contains

$[(\sigma^i)_{i=0}^{d}]_1$ (where d is some constant, resulting from the techniques used in the rest of the zk-SNARK), it now has to contain also $[(\sigma^i)_{i=d+2}^{2d}]_1$ and $[(\sigma^i)_{i=0}^{d}]_2$.

Since sumchecks have ubiquitous applications [BCS21], $\mathfrak{Count}$ is of independent interest. In particular, sumcheck is used in both updatable zk-SNARKs and transparent zk-SNARKs. As an important application, we will design a new updatable zk-SNARK. We leave it an interesting open question to apply $\mathfrak{Count}$ in transparent zk-SNARKs.

New Updatable zk-SNARK. We use $\mathfrak{Count}$ to design a new pairing-based updatable zk-SNARK $\mathfrak{Vampire}$ for the R1CSLite [CFF+21] constraint system, given the R1CS matrices are sparse as in [CHM+20,RZ21a,CFF+21]. The argument length of $\mathfrak{Vampire}$ is five elements of $\mathbb{G}_1$ and two elements of $\mathbb{F}$, which is less than in any known updatable zk-SNARK. It is almost twice smaller than in the previously best updatable zk-SNARK (LunarLite) for the same arithmetization. While Basilisk [RZ21a] (as improved in their full version, [RZ21b]) has just slightly larger communication than $\mathfrak{Vampire}$, it works for a version of R1CSLite with additional restrictions on the underlying matrices; the version of Basilisk for the same arithmetization handled by $\mathfrak{Vampire}$ is less communication-efficient than LunarLite. (It might be because this version of Basilisk was not optimized for communication efficiency.)

Let us now describe $\mathfrak{Vampire}$. Let m be the number of constraints. Following Lunar and Basilisk, we use the R1CSLite constraint system, where an instance consists of two parameter matrices $\mathbf{L}$ and $\mathbf{R}$ (the left and right inputs to all constraints) instead of three in the case of R1CS. Following Marlin, Lunar, and Basilisk, we use the setting of sparse matrices, where $\mathbf{L}$ and $\mathbf{R}$ have together at most $|\mathbb{K}| = \Theta(m)$ non-zero entries. Here, $\mathbb{K}$ is a multiplicative subgroup of $\mathbb{F}$.

$\mathfrak{Vampire}$ is based on the underlying ideas of Marlin (e.g., we use the same arithmetization of sparse matrices), but it uses optimizations of both Lunar [CFF+21] and Basilisk [RZ21a]. These optimizations (together with an apparently novel combination of the full witness to a single commitment) result in the argument length of 7 elements of $\mathbb{G}_1$ and 2 integers, which is already better than any prior updatable zk-SNARK for any NP-complete constraint system except Basilisk's version for bounded fan-out matrices.

$\mathfrak{Count}$ helps to remove one more group element from the argument of $\mathfrak{Vampire}$. This step in $\mathfrak{Vampire}$ is not trivial: the sumcheck argument requires that the polynomial f is committed to, which is not the case in $\mathfrak{Vampire}$. We solve this issue using a batching technique similar to Lunar and Basilisk, asking the prover to open two polynomial commitments. The second committed polynomial is a linear combination of other polynomial commitments with coefficients known to the prover and the verifier after opening the first polynomial.

Our second innovation is the use of polynomial commitment aggregation at different points from [BDFG20]. Intuitively, we commit to a single polynomial that encodes both the left and right inputs of all constraints; this allows us to save one more group element. When combining the result with the batching technique of the previous paragraph, we need to open two polynomials at different points.

For this, we use a technique of Boneh *et al.* [BDFG20]. However, our batching is not randomized since the two opening points are different.

More precisely, the prover starts $\mathfrak{Vampire}$ by committing to $\tilde{z}$, a polynomial related to the witness-encoding polynomial z. Committing to $\tilde{z}$ helps one check efficiently that the prover used the correct public input. The verifier replies with a random field element α. We reformulate the check that $\tilde{z}$ satisfies the R1CSLite instance as a univariate sumcheck argument for $\sum_{y\in\mathbb{H}} \psi_\alpha(y) = 0$, for a well-chosen polynomial ψ_α. We then run $\mathfrak{Count}$, letting the prover send an ILV-opening $[\psi_{\mathsf{ipc}}(\sigma)]_1$ to the verifier. The verifier replies with another random field element β. The prover's final message consists of two field elements and three group elements. These elements are needed to batch-open two polynomial commitments; it also involves a complicated but by now standard step of proving the correctness of the arithmetization of a sparse matrix. We will omit further details here and refer to Section 4 for more details.

We prove that $\mathfrak{Vampire}$ is knowledge-sound in the Algebraic Group Model (AGM) [FKL18]. We construct a reduction to the Power Discrete Logarithm (PDL) assumption, where the adversary gets as an input $([(\sigma^i)_{i=0}^{d_1}]_1, [(\sigma^i)_{i=1}^{d_2}]_2)$ and has to compute σ. Here, d_1 and d_2 are parameters related to the SRS size. Although the proof is technically challenging, the high-level approach we use is standard, [FKL18,BFL20,KMSV21]. In AGM, when the adversary outputs a group element $[a]_\iota$ in $\mathbb{G}_\iota$, one can extract coefficients $x_1, \ldots, x_t$, such that $[a]_\iota = \sum_{i=1}^t x_i [f_i(\sigma)]_\iota$, where $[f_i(\sigma)]_\iota$ are SRS elements and σ is a secret trapdoor. Essentially, we extract a polynomial $a(X) = \sum_{i=1}^t x_i f_i(X)$ such that $a = a(\sigma)$. The SNARK verification equation can also be expressed as a polynomial $\mathcal{V}(X)$ (where coefficients depend on the coefficients extracted from the argument) such that $\mathcal{V}(\sigma) = 0$ when the verifier accepts. Now, the proof proceeds in two branches. If $\mathcal{V}(X) = 0$, we reconstruct the witness from the coefficients extracted from the prover. However, if $\mathcal{V}(X) \neq 0$, it is possible to embed a PDL challenge into the SRS, and the reduction algorithm can efficiently find roots $\mathcal{V}$, one of which is the discrete logarithm σ. The actual proof is more complicated since we have two trapdoors (the challenge needs to be embedded in both of them such that trapdoors still look uniformly random and independent) and more than one verification equation.

We prove that $\mathfrak{Vampire}$ is perfectly zero-knowledge by constructing a simulator that utilizes the knowledge of trapdoor to work around the soundness of $\mathfrak{Count}$ and makes the sumcheck argument acceptable for any, even all-zero witness. For the simulated argument to be indistinguishable from a real one, we add random terms to polynomial $\tilde{z}(X)$ which, in the case of real argument, encodes witness at its coefficients, and, in the case of a simulated argument, encodes a (mostly) zero vector. This assures that even an unbounded adversary who knows the instance and witness cannot tell a commitment to $\tilde{z}(X)$ from a real argument from a commitment to a simulated $\tilde{z}(X)$.

We prove that $\mathfrak{Vampire}$ is Sub-ZK (i.e., zero-knowledge even if the SRS generation is compromised, [BFS16,ABLZ17,Fuc18,ALSZ21]) under the BDH-KE knowledge assumption [ABLZ17]. We first define Sub-ZK for interactive pro-

tocols. [ALSZ21] proved that a perfectly zero-knowledge argument system is statistically Sub-ZK if (1) there exists a PPT SRS-verifying algorithm SrsVer that certifies the correctness of the SRS (given only publicly available data); and (2) for every subverter $\mathcal{Z}$, that produces the SRS, there exists an extractor $\mathsf{Ext}_{\mathcal{Z}}$ that outputs the corresponding trapdoor. We construct SrsVer and $\mathsf{Ext}_{\mathcal{Z}}$ (given the BDH-KE extractor). Next, we conclude that given the trapdoor extracted by the extractor and SRS verified by SrsVer, the simulator produces an argument indistinguishable from the real one.

On Efficiency. We study how much the argument length (i.e., the communication complexity) can be reduced in updatable and universal SNARKs while only allowing minimal relaxations in other efficiency parameters. We achieve the shortest argument so far, and in particular, among zk-SNARKs for the sparse R1CS(Lite) constraint system, the argument size is significantly smaller than the previous work. The SRS size of our zk-SNARK is at most a small constant factor larger than in the previous work, which we believe is a reasonable compromise as the SRS needs to be delivered only once. $\mathfrak{Vampire}$ is especially advantageous when the R1CSLite instance is not super sparse. As a function of the number of non-zero elements in R1CSLite matrices only, $\mathfrak{Vampire}$ has the best prover's computation and the same SRS length and SRS generation time as any previous updatable zk-SNARK. See Appendix A for a thorough efficiency comparison.

2 Preliminaries

Let $\mathbb{F}$ be a finite field of order p, and let $\mathbb{F}_{\le d}[X] \subset \mathbb{F}[X]$ be the set of degree $\le d$ polynomials. (We always have $\mathbb{F} = \mathbb{Z}_p$.) Define the set of (d, d_{gap})-*punctured* univariate polynomials over $\mathbb{F}$ as $\mathsf{PolyPunc}_{\mathbb{F}}(d, d_{\mathsf{gap}}, X) := \{f(X) = \sum_{i=0}^{d_{\mathsf{gap}}+d} f_i X^i \in \mathbb{F}_{\le d_{\mathsf{gap}}+d}[X] : f_{d_{\mathsf{gap}}} = 0\}$. Let $\boldsymbol{x} \circ \boldsymbol{y}$ be the elementwise product of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, $\forall i.(x \circ y)_i = x_i y_i$. Let $\mathbf{I}_n \in \mathbb{F}^{n\times n}$ be the n-dimensional identity matrix. We denote matrix and vector elements by using square brackets as in $\mathbf{A}[i, j]$ and $\boldsymbol{a}[i]$.

Interpolation. Let ω be the n_h-th primitive root of unity in $\mathbb{F}$ and let $\mathbb{H} = \{\omega^j : 0 \le j < n_h\}$ be a multiplicative subgroup of $\mathbb{F}$. Then,

- The *vanishing polynomial* $\mathcal{Z}_{\mathbb{H}}(Y) := \prod_{i=1}^{n_h}(Y - \omega^{i-1}) = Y^{n_h} - 1$ is the unique degree n_h monic polynomial, such that $\mathcal{Z}_{\mathbb{H}}(\omega^{i-1}) = 0$ for all $i \in [1, n_h]$.
- For $i \in [1, n_h]$, $\ell_i^{\mathbb{H}}(Y)$ is the *ith Lagrange polynomial*, i.e., the unique degree $n_h - 1$ polynomial, such that $\ell_i^{\mathbb{H}}(\omega^{i-1}) = 1$ and $\ell_i^{\mathbb{H}}(\omega^{j-1}) = 0$ for $i \ne j$. It is well known that $\ell_i^{\mathbb{H}}(Y) = \mathcal{Z}_{\mathbb{H}}(Y)/(\mathcal{Z}'_{\mathbb{H}}(\omega^{i-1}) \cdot (Y - \omega^{i-1})) = \mathcal{Z}_{\mathbb{H}}(Y)\omega^{i-1}/(n_h(Y - \omega^{i-1}))$ when $Y \ne \omega^{i-1}$. (Here, $\mathcal{Z}'_{\mathbb{H}}(X) = d\mathcal{Z}_{\mathbb{H}}(X)/dX$.)
- $L_X(Y) := \mathcal{Z}_{\mathbb{H}}(Y)X/(n_h(Y - X)) \in \mathbb{F}(X, Y)$ (a lifted Lagrange polynomial), with $L_{\omega^{i-1}}(Y) = \ell_i^{\mathbb{H}}(Y)$ for $i \in [1, n_h]$.

For $f \in \mathbb{F}[X]$, let $\widehat{f}^{\mathbb{H}}(X) := \sum_{i=1}^{n_h} f(\omega^{i-1})\ell_i^{\mathbb{H}}(X)$ be its low-degree extension. To simplify notation, we often omit the accent $\widehat{\cdot}$ and the superscript $\mathbb{H}$.

R1CSLite. R1CSLite [CFF+21,RZ21a] is a variant of the well-known Rank 1 Constraint System [GGPR13,CHM+20]. An R1CSLite instance $\mathcal{I}_{\mathsf{r1cslite}} =$

$(\mathbb{F}, m, m_0, \mathbf{L}, \mathbf{R})$ consists of a field $\mathbb{F}$, instance size m, input size m_0, and matrices $\mathbf{L}, \mathbf{R} \in \mathbb{F}^{m\times m}$. An R1CSLite instance is *sparse* if $\mathbf{L}$ and $\mathbf{R}$ have $n_k = O(m)$ non-zero elements.

$\mathcal{I}_{\mathsf{r1cslite}} = (\mathbb{F}, m, m_0, \mathbf{L}, \mathbf{R})$ defines the following relation $\mathcal{R} = \mathcal{R}_{\mathcal{I}_{\mathsf{r1cslite}}}$:

$$\mathcal{R} := \left\{ \begin{array}{l} (\mathbb{x}, \mathbb{w}) \colon \mathbb{x} = (z_1, \ldots, z_{m_0})^\top \ \wedge \ \mathbb{w} = \left(\begin{smallmatrix} \boldsymbol{z}_\mathsf{a} \\ \boldsymbol{z}_\mathsf{b} \end{smallmatrix}\right) \ \wedge \ \boldsymbol{z}_\mathsf{a}, \boldsymbol{z}_\mathsf{b} \in \mathbb{F}^{m-m_0-1} \ \wedge \\ \boldsymbol{z}_\mathsf{l} = \left(\begin{smallmatrix} 1 \\ \mathbb{x} \\ \boldsymbol{z}_\mathsf{a} \end{smallmatrix}\right) \ \wedge \ \boldsymbol{z}_\mathsf{r} = \left(\begin{smallmatrix} \mathbf{1}_{m_0+1} \\ \boldsymbol{z}_\mathsf{b} \end{smallmatrix}\right) \ \wedge \ \boldsymbol{z}_\mathsf{l} = \mathbf{L}(\boldsymbol{z}_\mathsf{l} \circ \boldsymbol{z}_\mathsf{r}) \ \wedge \ \boldsymbol{z}_\mathsf{r} = \mathbf{R}(\boldsymbol{z}_\mathsf{l} \circ \boldsymbol{z}_\mathsf{r}) \end{array} \right\} .$$

Equivalently, $\mathbf{W}\boldsymbol{z}^* = \mathbf{0}$, where

$$\mathbf{W} = \left(\begin{smallmatrix} \mathbf{I}_m & \mathbf{0} & -\mathbf{L} \\ \mathbf{0} & \mathbf{I}_m & -\mathbf{R} \end{smallmatrix}\right) \in \mathbb{F}^{2m\times 3m} \quad , \quad \boldsymbol{z}^* = \left(\begin{smallmatrix} \boldsymbol{z}_\mathsf{l} \\ \boldsymbol{z}_\mathsf{r} \\ \boldsymbol{z}=\boldsymbol{z}_\mathsf{l}\circ\boldsymbol{z}_\mathsf{r} \end{smallmatrix}\right) . \tag{1}$$

Basic Cryptography. We denote the security parameter by λ. For any algorithm $\mathcal{A}$, $r \leftarrow\!\$\, \mathsf{RND}(\mathcal{A})$ samples random coins of sufficient length for $\mathcal{A}$. By $y \leftarrow \mathcal{A}(x; r)$, we denote that $\mathcal{A}$ outputs y on input x and random coins r. PPT means probabilistic polynomial time.

Pairings. A bilinear group generator $\mathsf{Pgen}(1^\lambda)$ returns $\mathsf{p} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, [1]_1, [1]_2)$, where p is a prime such that $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are three additive cyclic groups of order p, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate efficiently computable bilinear pairing, and $[1]_\iota$ is a generator of $\mathbb{G}_\iota$ for $\iota \in \{1, 2, T\}$ with $[1]_T = \hat{e}([1]_1, [1]_2)$. In the context of this work, $\mathbb{F} = \mathbb{Z}_p$ must always have two large multiplicative subgroups $\mathbb{H}$ and $\mathbb{K}$. Thus, we assume implicitly that $|\mathbb{H}|, |\mathbb{K}| \mid (p-1)$. We require the bilinear pairing to be Type-3, that is, not to have an efficient isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. In practice, one uses a fixed pairing-friendly curve like BLS-381; in this case, also $\mathbb{K}$ and $\mathbb{H}$ have a fixed order.

We use the by now standard additive bracket notation, by writing $[a]_\iota$ to denote $a[1]_\iota$ for $\iota \in \{1, 2, T\}$. We denote $\hat{e}([x]_1, [y]_2)$ by $[x]_1 \bullet [y]_2$. Thus, $[x]_1 \bullet [y]_2 = [xy]_T$. We freely use the bracket notation together with matrix notation; for example, if $\mathbf{A} \cdot \mathbf{B} = \mathbf{C}$ then $[\mathbf{A}]_1 \bullet [\mathbf{B}]_2 = [\mathbf{C}]_T$.

Polynomial Commitment Schemes. In a polynomial commitment scheme [KZG10], the prover commits to a polynomial $f \in \mathbb{F}_{\le d}[X]$ and later opens it to $f(\beta)$ for $\beta \in \mathbb{F}$ chosen by the verifier. The (non-randomized) KZG [KZG10] polynomial commitment scheme consists of the following algorithms:

Setup: Given 1^λ, return $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$.

Commitment key generation: Given the system parameter p and an upper-bound d on the polynomial degrees, compute the trapdoor $\mathsf{tk} = \sigma \leftarrow\!\$\, \mathbb{Z}_p^*$ and the commitment key $\mathsf{ck} \leftarrow (\mathsf{p}, [(\sigma^i)_{i=0}^d]_1, [1, \sigma]_2)$. Return $(\mathsf{ck}, \mathsf{tk})$.

Commitment: Given a commitment key ck and a polynomial $f \in \mathbb{F}_{\le d}[X]$, return the commitment $[f(\sigma)]_1 \leftarrow \sum_{j=0}^d f_j [\sigma^j]_1$.

Opening: Given a commitment key ck, a commitment $[f(\sigma)]_1$, an evaluation point $\beta \in \mathbb{F}$, and a polynomial $f \in \mathbb{F}_{\le d}[X]$, set $v \leftarrow f(\beta)$ and $f_{\mathsf{pc}}(X) \leftarrow (f(X) - v)/(X - \beta)$. Let the evaluation proof be $[f_{\mathsf{pc}}(\sigma)]_1 \leftarrow \sum_{j=0}^{d-1} (f_{\mathsf{pc}})_j [\sigma^j]_1$. Return $(v, [f_{\mathsf{pc}}(\sigma)]_1)$.

Verification: Given a commitment key ck, a commitment $[f(\sigma)]_1$, an evaluation point β, a purported evaluation $v = f(\beta)$, and an evaluation proof $[f_{\mathsf{pc}}(\sigma)]_1$, check $[f(\sigma) - v]_1 \bullet [1]_2 = [f_{\mathsf{pc}}(\sigma)]_1 \bullet [\sigma - \beta]_2$.

Its security is based on the fact that $(X - \beta) \mid (f(X) - v) \Leftrightarrow f(\beta) = v$.

Inner-Product Commitment Schemes. In an inner-product commitment scheme [LY10,ILV11], the prover commits to a vector $\boldsymbol{\mu} \in \mathbb{F}^n$ and later opens it to the inner product $\boldsymbol{\mu}^\top \boldsymbol{\nu}$ for $\boldsymbol{\nu} \in \mathbb{F}^n$ chosen by the verifier. The (non-randomized) ILV [ILV11] inner-product commitment scheme consists of the following algorithms:

Setup: Given 1^λ, return $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$.

Commitment key generation: Given system parameters p and a vector dimension n, compute the trapdoor $\mathsf{tk} = \sigma \leftarrow\!\!\$\, \mathbb{Z}_p^*$ and the commitment key $\mathsf{ck} \leftarrow ([(\sigma^i)_{i=0:i\neq n+1}^{2n}]_1, [(\sigma^i)_{i=0}^{n}]_2)$. Return $(\mathsf{ck}, \mathsf{tk})$.

Commitment: Given a commitment key ck and a vector $\boldsymbol{\mu} \in \mathbb{F}^n$, compute the coefficients of $\mu(X) \leftarrow \sum_{j=1}^n \mu_j X^j \in \mathbb{F}_{\leq n}[X]$; $[\mu(\sigma)]_1 = \sum_{j=1}^n \mu_j [\sigma^j]_1$. Return the commitment $[\mu(\sigma)]_1$.

Opening: Given a commitment key ck, a commitment $[\mu(\sigma)]_1$, the original vector $\boldsymbol{\mu}$, and a vector $\boldsymbol{\nu}$, let $v \leftarrow \boldsymbol{\mu}^\top \boldsymbol{\nu}$. Set $\nu^*(X) \leftarrow \sum_{j=1}^n \nu_j X^{n+1-j} \in \mathbb{F}_{\leq n}[X]$, and $\mu_{\mathsf{ipc}}(X) \leftarrow \mu(X)\nu^*(X) - vX^{n+1} \in \mathsf{PolyPunc}_{\mathbb{F}}(n-1, n+1, X)$. Let the evaluation proof be $[\mu_{\mathsf{ipc}}(\sigma)]_1 \leftarrow \sum_{i=1, i\neq n+1}^{2n} \mu_{\mathsf{ipc}}[\sigma^i]_1$. Return $(v, [\mu_{\mathsf{ipc}}(\sigma)]_1)$.

Verification: Given a commitment key ck, a commitment $[\mu(\sigma)]_1$, a vector $\boldsymbol{\nu}$, a purported value $v = \boldsymbol{\mu}^\top \boldsymbol{\nu}$, and an evaluation proof $[\mu_{\mathsf{ipc}}(\sigma)]_1$, check $[\mu_{\mathsf{ipc}}(\sigma)]_1 \bullet [1]_2 = [\mu(\sigma)]_1 \bullet \sum_{j=1}^n \nu_j [\sigma^{n+1-j}]_2 - v[\sigma^n]_1 \bullet [\sigma]_2$.

The security of ILV relies on the fact that the coefficient of X^{n+1} in $\mu_{\mathsf{ipc}}(X)$ is $\boldsymbol{\mu}^\top \boldsymbol{\nu} - v$, which is zero iff v is correctly computed. In our application, the vector $\boldsymbol{\nu}$ is known in advance and public. In this case, the verifier only has to compute two pairings and no exponentiations.

Succinct Zero-Knowledge Arguments. The following definition is closely based on [CFF+21]. Groth *et al.* [GKM+18] introduced the notion of (preprocessing) zk-SNARKs with specializable universal structured reference string (SRS). This notion formalizes the idea that key generation for $\mathcal{R} \in \mathcal{UR}$ can be seen as the sequential combination of two steps: first, a probabilistic algorithm that generates an SRS for the universal relation $\mathcal{UR}$ and second, a deterministic algorithm that specializes this universal SRS into one for a specific $\mathcal{R}$.

We consider relation families $(\mathsf{Pgen}, \{\mathcal{UR}_{\mathsf{p},N}\}_{\mathsf{p}\in\mathrm{range}(\mathsf{Pgen}),N\in\mathbb{N}})$ parametrized by $\mathsf{p} \in \mathsf{Pgen}(1^\lambda)$ and a size bound $N \in \mathsf{poly}(\lambda)$.[6] A *succinct zero-knowledge argument* $\Pi = (\mathsf{Pgen}, \mathsf{KGen}, \mathsf{Derive}, \mathsf{P}, \mathsf{V})$ *with specializable universal SRS for a relation family* $(\mathsf{Pgen}, \{\mathcal{UR}_{\mathsf{p},N}\}_{\mathsf{p}\in\{0,1\}^*,N\in\mathbb{N}})$ works as follows.

Setup: Given 1^λ, return $\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda)$.

Universal SRS Generation: a probabilistic algorithm $\mathsf{KGen}(\mathsf{p}, N) \to (\mathsf{srs}, \mathsf{td})$ that takes as input public parameters p, an upper bound N on the relation

[6] In the case of $\mathfrak{Count}$ and $\mathfrak{Vampire}$, we actually have several size bounds. The definitions generalize naturally.

size, and outputs $\mathsf{srs} = (\mathsf{ek}, \mathsf{vk})$ together with a trapdoor. We assume implicitly that srs, and other elements like ek, and vk contain p.

SRS Specialization: a deterministic algorithm $\mathsf{Derive}(\mathsf{srs}, \mathcal{R}) \to (\mathsf{ek}_\mathcal{R}, \mathsf{vk}_\mathcal{R})$ that takes as input a universal SRS srs and a relation $\mathcal{R} \in \mathcal{UR}_{\mathsf{p},N}$, and outputs specialized SRS $\mathsf{srs}_\mathcal{R} := (\mathsf{ek}_\mathcal{R}, \mathsf{vk}_\mathcal{R})$.

Prover/Verifier: a pair of interactive algorithms $\langle \mathsf{P}(\mathsf{ek}_\mathcal{R}, \mathbb{x}, \mathbb{w}), \mathsf{V}(\mathsf{vk}_\mathcal{R}, \mathbb{x})\rangle \to b$, where P takes a proving key $\mathsf{ek}_\mathcal{R}$ for a relation $\mathcal{R}$, a statement $\mathbb{x}$, and a witness $\mathbb{w}$ such that $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, and V takes a verification key for a relation $\mathcal{R}$, a statement $\mathbb{x}$, and either accepts ($b = 1$) or rejects ($b = 0$) the proof.

Π must satisfy the following four requirements.

Completeness. For all $\mathsf{p} \in \mathrm{range}(\mathsf{Pgen})$, $N \in \mathbb{N}$, $\mathcal{R} \in \mathcal{UR}_{\mathsf{p},N}$, and $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$,

$$\Pr\left[\langle \mathsf{P}(\mathsf{ek}_\mathcal{R}, \mathbb{x}, \mathbb{w}), \mathsf{V}(\mathsf{vk}_\mathcal{R}, \mathbb{x})\rangle = 1 \;\middle|\; \begin{array}{c}(\mathsf{srs}, \mathsf{td}) \leftarrow \mathsf{KGen}(\mathsf{p}, N);\\ (\mathsf{ek}_\mathcal{R}, \mathsf{vk}_\mathcal{R}) \leftarrow \mathsf{Derive}(\mathsf{srs}, \mathcal{R})\end{array}\right] = 1 \ .$$

Succinctness. Π is *succinct* if the running time of V is $\mathsf{poly}(\lambda + |\mathbb{x}| + \log|\mathbb{w}|)$ and the communication size is $\mathsf{poly}(\lambda + \log|\mathbb{w}|)$.

Knowledge-Soundness. Π has *knowledge-soundness* for an auxiliary input distribution $\mathcal{C}$, if for every non-uniform PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a non-uniform PPT extractor $\mathsf{Ext}_\mathcal{A}$, such that

$$\Pr\left[\begin{array}{c}\langle \mathcal{A}_2(st; r), \mathsf{V}(\mathsf{vk}_\mathcal{R}, \mathbb{x})\rangle = 1\\ \wedge\, \neg\mathcal{R}(\mathbb{x}, \mathbb{w})\end{array} \;\middle|\; \begin{array}{c}\mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); (\mathsf{srs}, \mathsf{td}) \leftarrow \mathsf{KGen}(\mathsf{p}, N);\\ \mathsf{aux} \leftarrow \mathcal{C}(\mathsf{srs}); r \leftarrow\$\, \mathsf{RND}(\mathcal{A});\\ (\mathcal{R}, \mathbb{x}, st) \leftarrow \mathcal{A}_1(\mathsf{srs}, \mathsf{aux}; r);\\ \mathbb{w} \leftarrow \mathsf{Ext}_\mathcal{A}(\mathsf{srs}, \mathsf{aux}; r);\\ (\mathsf{ek}_\mathcal{R}, \mathsf{vk}_\mathcal{R}) \leftarrow \mathsf{Derive}(\mathsf{srs}, \mathcal{R})\end{array}\right]$$

is $\mathsf{negl}(\lambda)$. Π is *knowledge-sound* if there exists benign $\mathcal{C}$ such that Π is knowledge-sound for $\mathcal{C}$.

Zero-Knowledge. Π is (statistical) *zero-knowledge* if there exists a PPT simulator Sim, s.t. for all unbound $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, all $\mathsf{p} \in \mathrm{range}(\mathsf{Pgen})$, all $N \in \mathsf{poly}(\lambda)$,

$$\Pr\left[\begin{array}{c}\langle \mathsf{P}(\mathsf{ek}_\mathcal{R}, \mathbb{x}, \mathbb{w}), \mathcal{A}_2(st)\rangle = 1 \wedge\\ \mathcal{R}(\mathbb{x}, \mathbb{w}) \wedge \mathcal{R} \in \mathcal{UR}_{\mathsf{p},N}\end{array} \;\middle|\; \begin{array}{c}(\mathsf{srs}, \mathsf{td}) \leftarrow \mathsf{KGen}(\mathsf{p}, N);\\ (\mathcal{R}, \mathbb{x}, \mathbb{w}, st) \leftarrow \mathcal{A}_1(\mathsf{srs});\\ (\mathsf{ek}_\mathcal{R}, \mathsf{vk}_\mathcal{R}) \leftarrow \mathsf{Derive}(\mathsf{srs}, \mathcal{R})\end{array}\right] \approx_s$$
$$\Pr\left[\begin{array}{c}\langle \mathsf{Sim}(\mathsf{srs}, \mathsf{td}, \mathcal{R}, \mathbb{x}), \mathcal{A}_2(st)\rangle = 1 \wedge\\ \mathcal{R}(\mathbb{x}, \mathbb{w}) \wedge \mathcal{R} \in \mathcal{UR}_{\mathsf{p},N}\end{array} \;\middle|\; \begin{array}{c}(\mathsf{srs}, \mathsf{td}) \leftarrow \mathsf{KGen}(\mathsf{p}, N);\\ (\mathcal{R}, \mathbb{x}, \mathbb{w}, st) \leftarrow \mathcal{A}_1(\mathsf{srs});\\ (\mathsf{ek}_\mathcal{R}, \mathsf{vk}_\mathcal{R}) \leftarrow \mathsf{Derive}(\mathsf{srs}, \mathcal{R})\end{array}\right] \ .$$

Here, $\approx_s$ denotes the statistical distance as a function of λ. Π is *perfect zero-knowledge* if the above probabilities are equal.

Π is *subversion zero-knowledge* (Sub-ZK, [BFS16]), if it is zero-knowledge even in the case the SRS is maliciously generated. For perfect zero-knowledge arguments, Sub-ZK follows from the usual zero-knowledge (with trusted SRS), the SRS verifiability (there exists a PPT algorithm that checks that the SRS belongs to range(KGen)), and a SNARK-specific knowledge assumption, [ABLZ17,ALSZ21]. We will provide the formal definition in Section 5.2.

Π is *updatable* [GKM+18], if the SRS can be sequentially updated by many updaters, such that knowledge-soundness holds if either the original SRS creator

or one of the updaters is honest. [GKM+18] showed that an updatable SRS cannot contain non-monomial polynomial evaluations. Moreover, an updatable SRS must be verifiable in the same sense as in the case of Sub-ZK.

Since $\mathfrak{Vampire}$ is public-coin and has a constant number of rounds, we can apply the Fiat-Shamir heuristic [FS87] to obtain a zk-SNARK.

Sumcheck Arguments. In a sumcheck argument [LFKN90] over $\mathbb{F}$, the prover convinces the verifier that for $\mathbb{H} \subseteq \mathbb{F}$, $f \in \mathbb{F}[X_1, \ldots, X_c]$, and $v_f \in \mathbb{F}$, it holds that $\sum_{(x_1,\ldots,x_c)\in\mathbb{H}^c} f(x_1, \ldots, x_c) = v_f$. Multivariate sumcheck has many applications in both complexity theory and cryptography, [BCS21], with usually relatively small $|\mathbb{H}|$ but large c. In the context of efficient updatable zk-SNARKs, one is often interested in *univariate sumcheck*, where $c = 1$ but $|\mathbb{H}|$ is large. Univariate sumcheck arguments are most efficient when $\mathbb{H}$ is either an affine subspace or a multiplicative subgroup [BCR+19]; see [RZ21a] for a generalization to arbitrary $\mathbb{H}$.

The univariate sumcheck relation for multiplicative subgroups is the set of all pairs $\mathcal{R}_{\mathsf{sum}} := ((\mathbb{F}, d, \mathbb{H}, v_f), f)$, where $\mathbb{F}$ is a finite field, d is a positive integer, $\mathbb{H}$ is a multiplicative subgroup of $\mathbb{F}$, $v_f \in \mathbb{F}$, $f \in \mathbb{F}_{\leq d}[X]$, and $\sum_{\chi\in\mathbb{H}} f(\chi) = v_f$.

Aurora's Sumcheck. As a part of the zk-SNARK Aurora, Ben-Sasson *et al.* [BCR+19] proposed an efficient univariate sumcheck ("Aurora's sumcheck") for multiplicative subgroups. Since the new univariate sumcheck relies on similar techniques, we next recall Aurora's sumcheck.

As before, let $\mathbb{H} = \langle\omega\rangle = \{\omega^i : i \in [0, n_h - 1]\}$ be a cyclic multiplicative subgroup of order $n_h = |\mathbb{H}|$. Fact 1 underlies Aurora's sumcheck. (See, e.g., Fact 9.1, [Tha21] for the proof.)

Fact 1 *Let $f \in \mathbb{F}[X]$ with $\deg f \leq n_h - 1$. Then $\sum_{\chi\in\mathbb{H}} f(\chi) = n_h f(0)$.*

In the case of a large-degree f, Aurora's sumcheck uses the following result that follows from Fact 1 and polynomial division (namely, that $f(X) = R(X) + Q(X)\mathcal{Z}_{\mathbb{H}}(X)$ for $R \in \mathbb{F}_{\leq n_h-1}[X]$).

Fact 2 (Core Lemma of Aurora's Sumcheck) *Let $f \in \mathbb{F}[X]$ with $d = \deg f$. Then, $\sum_{\chi\in\mathbb{H}} f(\chi) = v_f$ iff there exist $R \in \mathbb{F}_{\leq n_h-2}[X]$ and $Q \in \mathbb{F}_{\leq d-n_h}[X]$, such that $f(X) = v_f/n_h + R(X)X + Q(X)\mathcal{Z}_{\mathbb{H}}(X)$.*

Ben-Sasson *et al.* [BCR+19] used Fact 2 to construct a sumcheck argument for proving that $\sum_{\chi\in\mathbb{H}} f(\chi) = v_f$. In Aurora's sumcheck argument, the prover sends to the verifier polynomial commitments to f, R, and Q. Assume that $d = \deg f = \mathsf{poly}(\lambda)$ while $p = 2^{\Theta(\lambda)}$. The verifier accepts if (1) R has a low degree $\leq n_h - 2$ and (2) $f(X) = v_f/n_h + R(X)X + Q(X)\mathcal{Z}_{\mathbb{H}}(X)$.

On top of two polynomial commitments (two group elements), one has to implement a low-degree test to check that the degree of R is at most $n_h - 2$. As the low-degree test, Aurora used an interactive oracle proof for testing proximity to the Reed–Solomon code [BBHR18], resulting in additional costs. Recently, the full version of Basilisk [RZ21b] implemented a low-degree test in a partially costless way (namely, without added argument size or verifier's computation);

however, one may need to add a large number of elements to the SRS of the resulting zk-SNARK for their low-degree test to succeed.

Assumptions. Let $d_1(\lambda), d_2(\lambda) \in \mathsf{poly}(\lambda)$. Then, Pgen is (d_1, d_2)-*PDL (Power Discrete Logarithm) secure* if for any non-uniform PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{pdl}}_{d_1,d_2,\mathsf{Pgen},\mathcal{A}}(\lambda) :=$

$$\Pr\left[\mathcal{A}\left(\mathsf{p}, [(x^i)_{i=0}^{d_1}]_1, [(x^i)_{i=0}^{d_2}]_2\right) = x \,\middle|\, \mathsf{p} \leftarrow \mathsf{Pgen}(1^\lambda); x \leftarrow\!\$\, \mathbb{F}^*\right] = \mathsf{negl}(\lambda) \ .$$

Algebraic Group Model (AGM). AGM is an idealized model [FKL18] for security proofs. In the AGM, adversaries are restricted to be *algebraic* in the following sense: if $\mathcal{A}$ inputs some group elements and outputs a group element, it can provide an algebraic representation of the latter in terms of the former.

A PPT algorithm $\mathcal{A}$ is *algebraic (in* p*)* if there exists an efficient extractor $\mathsf{Ext}_{\mathcal{A}}$, such that for any PPT-sampleable distribution $\mathcal{D}$, $\mathsf{Adv}^{\mathrm{agm}}_{\mathsf{p},\mathcal{D},\mathcal{A},\mathsf{Ext}_{\mathcal{A}}}(\lambda) :=$

$$\Pr\left[\begin{array}{l} \boldsymbol{y}_1 \neq \boldsymbol{\Gamma}_1 \boldsymbol{x}_1 \vee \\ \boldsymbol{y}_2 \neq \boldsymbol{\Gamma}_2 \boldsymbol{x}_2 \end{array} \middle| \begin{array}{c} \mathbb{x} = ([\boldsymbol{x}_1]_1, [\boldsymbol{x}_2]_2) \leftarrow\!\$\, \mathcal{D}; r \leftarrow\!\$\, \mathsf{RND}(\mathcal{A}); \\ ([\boldsymbol{y}_1]_1, [\boldsymbol{y}_2]_2) \leftarrow\!\$\, \mathcal{A}(\mathbb{x}; r); (\boldsymbol{\Gamma}_1, \boldsymbol{\Gamma}_2) \leftarrow \mathsf{Ext}_{\mathcal{A}}(\mathbb{x}; r) \end{array} \right] = \mathsf{negl}(\lambda) \ .$$

3 New Univariate Sumcheck Argument

In this section, we propose $\mathfrak{Count}$, a new sumcheck argument that has improved online efficiency (including the argument size) but a larger SRS size than Aurora's univariate sumcheck. We first prove the following generalization of Fact 1, an alternative to Fact 2 in the case f has degree larger than $n_h - 1$.

Lemma 1. *Let* $f(X) = \sum_{i=0}^{d} f_i X^i$ *for* $d \geq 0$. *Then,*

$$\textstyle\sum_{\chi \in \mathbb{H}} f(\chi) = n_h \cdot \sum_{i=0}^{\lfloor d/n_h \rfloor} f_{n_h i} \ .$$

Proof. Write $f(X) = R(X) + Q(X)\mathcal{Z}_{\mathbb{H}}(X)$ for $\deg R \leq n_h - 1$. Based on Fact 1, $\sum_{\chi \in \mathbb{H}} f(\chi) = \sum_{\chi \in \mathbb{H}} R(\chi) = n_h R(0)$. Since $X^{n_h} \equiv 1 \pmod{\mathcal{Z}_{\mathbb{H}}(X)}$, $f(X) = \sum_{i=0}^{d} f_i X^i \equiv \sum_{j=0}^{n_h-1} (\sum_{i=0:n_h|(i-j)}^{d} f_i) X^j \pmod{\mathcal{Z}_{\mathbb{H}}(X)}$. Since $f(X) \equiv R(X) \pmod{\mathcal{Z}_{\mathbb{H}}(X)}$, $R(0) = \sum_{i=0}^{\lfloor d/n_h \rfloor} f_{n_h i}$. Thus, $\sum_{\chi \in \mathbb{H}} f(\chi) = n_h \cdot \sum_{i=0}^{\lfloor d/n_h \rfloor} f_{n_h i}$. □

$\mathfrak{Count}$ is based on the following result.

Lemma 2 (Core Lemma of $\mathfrak{Count}$). *Let* $n_h > 1$, $d_{\mathsf{gap}}, d > 0$ *with* $d_{\mathsf{gap}} \geq n_h \cdot \lfloor d/n_h \rfloor$, *and* $f \in \mathsf{PolyPunc}_{\mathbb{F}}(d, d_{\mathsf{gap}}, X)$. *Let* $\mathbb{H}$ *be an order-*n_h *multiplicative subgroup of* $\mathbb{F}$. *Define*

$$S(X) := \textstyle\sum_{i=0}^{\lfloor d/n_h \rfloor} X^{d_{\mathsf{gap}} - n_h i} \in \mathbb{F}_{\leq d_{\mathsf{gap}}}[X] \ .$$

Then, $\sum_{\chi \in \mathbb{H}} f(\chi) = v_f$ *and* $\deg f \leq d$ *iff there exists* $f_{\mathsf{ipc}} \in \mathsf{PolyPunc}_{\mathbb{F}}(d, d_{\mathsf{gap}}, X)$, *such that*

$$f(X)S(X) - f_{\mathsf{ipc}}(X) = \tfrac{v_f}{n_h} \cdot X^{d_{\mathsf{gap}}} \ . \tag{2}$$

$\mathsf{Pgen}(\mathsf{p})$: generate p as usually. We implicitly assume $n_h \mid (p-1)$.

$\mathsf{KGen}(\mathsf{p}, n_h, d, d_{\mathsf{gap}})$: $\mathcal{S}_1 \leftarrow \{(X^i)_{i=0:i\neq d_{\mathsf{gap}}}^{d_{\mathsf{gap}}+d}\}$; $\mathcal{S}_2 \leftarrow \{1, X, (X^{d_{\mathsf{gap}}-n_h i})_{i=0}^{\lfloor d/n_h \rfloor}\}$;
$\sigma \leftarrow\$ \mathbb{F}^*$; $\mathsf{td} \leftarrow \sigma$; $\mathsf{srs} \leftarrow (\mathsf{p}, n_h, d, d_{\mathsf{gap}}, [g(\sigma) : g \in \mathcal{S}_1]_1, [g(\sigma) : g \in \mathcal{S}_2]_2)$

$\mathsf{Derive}(\mathsf{srs})$: $S(X) \leftarrow \sum_{i=0}^{\lfloor d/n_h \rfloor} X^{d_{\mathsf{gap}}-n_h i} \in \mathbb{F}_{\leq d_{\mathsf{gap}}}[X]$; $\mathsf{ek}_{\mathcal{R}} \leftarrow \mathsf{srs}$;
$\mathsf{vk}_{\mathcal{R}} \leftarrow (\mathsf{srs}, [S(\sigma)]_2, [\sigma^{d_{\mathsf{gap}}}]_T)$; return $(\mathsf{ek}_{\mathcal{R}}, \mathsf{vk}_{\mathcal{R}})$;

$\mathsf{P}(\mathsf{ek}_{\mathcal{R}}, \mathbb{x}, \mathbb{w} = f)$ /* $\mathbb{x} = ([f(\sigma)]_1, v_f)$ */ $\mathsf{V}(\mathsf{vk}_{\mathcal{R}}, \mathbb{x})$

Online phase

$S(X) \leftarrow \sum_{i=0}^{\lfloor d/n_h \rfloor} X^{d_{\mathsf{gap}}-n_h i} \in \mathbb{F}_{\leq d_{\mathsf{gap}}}[X]$; $f_{\mathsf{ipc}}(X) \leftarrow f(X)S(X) - v_f/n_h \cdot X^{d_{\mathsf{gap}}}$

$[f_{\mathsf{ipc}}(\sigma)]_1 \longrightarrow$

Check $[f(\sigma)]_1 \bullet [S(\sigma)]_2 - [f_{\mathsf{ipc}}(\sigma)]_1 \bullet [1]_2 = v_f/n_h \cdot [\sigma^{d_{\mathsf{gap}}}]_T$

Fig. 1. The new univariate sumcheck zk-SNARK $\mathfrak{Count}$ for $\sum_{\chi\in\mathbb{H}} f(\chi) = v_f$.

Here, d_{gap} is a parameter fixed by the master protocol (in our case, $\mathfrak{Vampire}$) that uses $\mathfrak{Count}$ as a subroutine.

Proof. Clearly, we need $d_{\mathsf{gap}} \geq n_h \cdot \lfloor d/n_h \rfloor$ for S to be a polynomial.

($\Rightarrow$) Define $f_{\mathsf{ipc}}(X) := f(X)S(X) - v_f/n_h \cdot X^{d_{\mathsf{gap}}}$. We must only show that $f_{\mathsf{ipc}} \in \mathsf{PolyPunc}_{\mathbb{F}}(d, d_{\mathsf{gap}}, X)$. Since $\deg f \leq d$ and $\deg S = d_{\mathsf{gap}}$, we have $\deg f_{\mathsf{ipc}} \leq d_{\mathsf{gap}} + d$. Since $f(X)S(X) = (\sum_{i=0}^{d} f_i X^i)(\sum_{i=0}^{\lfloor d/n_h \rfloor} X^{d_{\mathsf{gap}}-n_h i})$, the coefficient of $X^{d_{\mathsf{gap}}}$ in $f(X)S(X)$ is $\sum_{i=0}^{\lfloor d/n_h \rfloor} f_{n_h i}$. By Lemma 1, $\sum_{i=0}^{\lfloor d/n_h \rfloor} f_{n_h i} = v_f/n_h$. Thus, the coefficient of $X^{d_{\mathsf{gap}}}$ in f_{ipc} is 0 and $f_{\mathsf{ipc}} \in \mathsf{PolyPunc}_{\mathbb{F}}(d, d_{\mathsf{gap}}, X)$.

($\Leftarrow$) Suppose Eq. (2) holds for $f_{\mathsf{ipc}} \in \mathsf{PolyPunc}_{\mathbb{F}}(d, d_{\mathsf{gap}}, X)$. Since $\deg S = d_{\mathsf{gap}}$ and $\deg f_{\mathsf{ipc}} \leq d_{\mathsf{gap}} + d$, we have $\deg f \leq d$. As in ($\Rightarrow$), the coefficient of $X^{d_{\mathsf{gap}}}$ in $f(X)S(X)$ is $\sum_{i=0}^{\lfloor d/n_h \rfloor} f_{n_h i}$, which is equal to $(\sum_{\chi\in\mathbb{H}} f(\chi))/n_h$ due to Lemma 1. Since f_{ipc} is missing the monomial $X^{d_{\mathsf{gap}}}$, we get that $v_f = \sum_{\chi\in\mathbb{H}} f(\chi)$. □

It is important that f_{ipc} has degree $\leq d_{\mathsf{gap}} + d$. Thus, one cannot add elements $[\sigma^i]_1$ for $i > d_{\mathsf{gap}} + d$ to the SRS of a master argument that uses $\mathfrak{Count}$.

Description of $\mathfrak{Count}$. In $\mathfrak{Count}$, the common input is $([f(\sigma)]_1, v_f)$. The prover sends to the verifier a polynomial commitment to $[f_{\mathsf{ipc}}(\sigma)]_1$, and the verifier accepts that $\sum_{\chi\in\mathbb{H}} f(\chi) = v_f$ iff a naturally modified version of Eq. (2) holds on committed polynomials. See Fig. 1 for the full argument.

Since we only use $\mathfrak{Count}$ as a sub-argument of $\mathfrak{Vampire}$, we do not formally have to prove that it is knowledge-sound or zero-knowledge. Nevertheless, we provide proof sketches for the sake of completeness.

Lemma 3. *The new sumcheck zk-SNARK* $\mathfrak{Count}$ *in Eq.* (2) *is complete and perfectly zero-knowledge. Additionally, the probability that any algebraic* $\mathcal{A}$ *can break knowledge soundness is bounded by* $\mathsf{Adv}^{\mathrm{pdl}}_{d_1,d_2,\mathsf{Pgen},\mathcal{B}}(\lambda)$ *where* $\mathcal{B}$ *is some PPT adversary,* $d_1 = d_{\mathsf{gap}} + d$, *and* $d_2 = d_{\mathsf{gap}}$.

Proof. Completeness follows from Lemma 2.

We sketch a knowledge-soundness proof in the AGM [FKL18]. Since $\mathcal{A}$ is algebraic, we get $f(X), f_{\mathsf{ipc}}(X)$ are in the span of X^i for $i \in \mathcal{S}_1$, i.e., $f, f_{\mathsf{ipc}} \in \mathsf{PolyPunc}_{\mathbb{F}}(d, d_{\mathsf{gap}}, X)$. If Eq. (2) holds, then by Lemma 2, the prover is honest. Otherwise, we have a known non-zero polynomial $\mathcal{V}(X) := f(X)S(X) - f_{\mathsf{ipc}}(X) - v_f/n_h \cdot X^{d_{\mathsf{gap}}}$ (its coefficients are known since the adversary is algebraic), such that (since the verifier accepts) σ is a root of $\mathcal{V}$. Thus, we can construct a (d_1, d_2)-PDL adversary $\mathcal{B}$ that gets $(\mathsf{p}, [(\sigma^i)_{i=0}^{d_1}]_1, [(\sigma^i)_{i=0}^{d_2}]_2)$ as an input. $\mathcal{B}$ constructs srs from the challenge input, and runs $\mathcal{A}$ and its extractor $\mathsf{Ext}_{\mathcal{A}}$ to obtain $\mathcal{V}(X)$. Whenever, $\mathcal{V}(X) \neq 0$, $\mathcal{B}$ can find the root σ and break the PDL assumption.

We now construct a simulator Sim that on input $(\mathsf{srs}, \mathsf{td}, ([f(\sigma)]_1, v_f))$ outputs an argument indistinguishable from a real argument. The simulator just computes $[f_{\mathsf{ipc}}(\sigma)]_1$, such that the verification equation holds. That is, $[f_{\mathsf{ipc}}(\sigma)]_1 \leftarrow S(\sigma)[f(\sigma)]_1 - v_f/n_h \cdot \sigma^{d_{\mathsf{gap}}}[1]_1$. Since in the real argument, $[f_{\mathsf{ipc}}(\sigma)]_1$ is computed the same way, the simulated and real argument are indistinguishable. □

SRS Verifiability. As noted in Section 2, for both Sub-ZK and updatability, it is required that the SRS is verifiable, i.e., that there exists a PPT algorithm that checks that the SRS belongs to the span of KGen. One can verify $\mathfrak{Count}$'s SRS by checking that $[\sigma]_1 \bullet [1]_2 = [1]_1 \bullet [\sigma]_2$, $[\sigma^i]_1 \bullet [1]_2 = [\sigma^{i-1}]_1 \bullet [\sigma]_2$ for $i \in [1, d_{\mathsf{gap}} + d] \setminus \{d_{\mathsf{gap}}, d_{\mathsf{gap}} + 1\}$, $[\sigma^{d_{\mathsf{gap}}+1}]_1 \bullet [1]_2 = [\sigma]_1 \bullet [\sigma^{d_{\mathsf{gap}}}]_2$, $[\sigma^{d_{\mathsf{gap}}-1}]_1 \bullet [\sigma]_2 = [1]_1 \bullet [\sigma^{d_{\mathsf{gap}}}]_2$, and $[\sigma^{n_h i}]_1 \bullet [\sigma^{d_{\mathsf{gap}}-n_h i}]_2 = [1]_1 \bullet [\sigma^{d_{\mathsf{gap}}}]_2$ for $i \in [1, \lfloor d/n_h \rfloor]$. Since, in addition, $\mathfrak{Count}$'s SRS consists of monomials only, $\mathfrak{Count}$ is updatable.

Efficiency. In $\mathfrak{Count}$, the prover outputs a single group element instead of two in Aurora's univariate sumcheck argument. The latter also requires one to implement a low-degree test, while there is no need for a low-degree test in $\mathfrak{Count}$.

Another important aspect of $\mathfrak{Count}$ is the prover's computation. In Aurora's univariate sumcheck, the prover computes polynomials R and Q, such that $f(X) = v_f/n_h + XR(X) + Q(X)\mathcal{Z}_{\mathbb{H}}(X)$; this can be done in quasilinear number of $\mathbb{F}$ operations. On the other hand, since in $\mathfrak{Vampire}$, S only has a small number of non-zero coefficients, the prover of $\mathfrak{Count}$ only executes a linear number of $\mathbb{F}$ operations, Both univariate sumchecks however require the prover to use a linear number of $\mathbb{G}_1$ operations. Note that linear-time *multivariate* sumchecks are well-known, [Tha13] and important in applications.

We emphasize that d_{gap} needs to satisfy $d_{\mathsf{gap}} \geq n_h \cdot \lfloor d/n_h \rfloor$, but it can be bigger. Iin $\mathfrak{Vampire}$, $d_{\mathsf{gap}} = d$.

As a drawback, the SRS contains more elements than in Aurora's sumcheck. This is a consequence of using the ILV inner-product commitment scheme.

4 $\mathfrak{Vampire}$: New Updatable Zk-SNARK

In this section, we will use $\mathfrak{Count}$ to construct an efficient zk-SNARK $\mathfrak{Vampire}$, with non-consequent monomial SRS, for the sparse R1CSLite constraint system. At a very high level, we use the general approach as Marlin [CHM+20], taking into account optimizations of Lunar [CFF+21] and Basilisk [RZ21a]. On top of already aggressive optimization, we use three novel techniques.

First, Marlin uses Aurora's univariate sumcheck twice. We replace the latter with $\mathfrak{Count}$ in one of the instantiations (in the second instantiation, Aurora's univariate sumcheck is actually more efficient). Second, we use a variant of the aggregated polynomial commitment scheme of [BDFG20] to batch together the opening of two different polynomials at different points. While [BDFG20] proposed only a randomized batch-opening protocol, we observe that in our case, it can be deterministic. Third, we use a single commitment to commit to left and right inputs of each constraint. Each techniques shaves one group element from the communication. As the end result, $\mathfrak{Vampire}$ is the most communication-efficient known updatable zk-SNARK $\mathfrak{Vampire}$ for any NP-complete constraint system. (See Table 1 for an efficiency comparison.)

4.1 Formulating R1CSLite as Sumcheck

Let $\mathbb{F} = \mathbb{Z}_p$. As in [CHM+20,RZ21a,CFF+21], let $\mathbb{H} = \langle \omega \rangle$ and $\mathbb{K}$ be two multiplicative subgroups of $\mathbb{F}$. We use $\mathbb{H}$ to index the rows (and columns) and $\mathbb{K}$ to index the non-zero elements of specific matrices. From now on, we assume that $\mathcal{I}_{\mathsf{r1cslite}} = (\mathbb{F}, \mathbb{H}, \mathbb{K}, m, m_0, \mathbf{L}, \mathbf{R})$ includes the description of $\mathbb{H}$ and $\mathbb{K}$.

We want to demonstrate the satisfiability of an R1CSLite instance $\mathcal{I}_{\mathsf{r1cslite}}$. Recall from Eq. (1) that for this we need to show that $\mathbf{W} \cdot \boldsymbol{z}^* = \mathbf{0}$, where $\mathbf{W} = (\mathbf{I}_{2m} \| -\mathbf{M})$, $\mathbf{M} = \left(\begin{smallmatrix} \mathbf{L} \\ \mathbf{R} \end{smallmatrix}\right)$, and $\boldsymbol{z}^* = (\boldsymbol{z}_{\mathbf{l}}{}^\top \| \boldsymbol{z}_{\mathbf{r}}{}^\top \| (\boldsymbol{z}_{\mathbf{l}} \circ \boldsymbol{z}_{\mathbf{r}})^\top)^\top$, where $\boldsymbol{z}_{\mathbf{l}}$ and $\boldsymbol{z}_{\mathbf{r}}$ are the vectors of all left and right inputs of all R1CSLite constraints.

Zero-Knowledge. To obtain zero-knowledge, we use a technique motivated by [RZ21b]. Let $|\mathbb{H}| = n_h := 2m + b$, for a randomizing parameter $b \in \mathbb{N}$ (to be fixed to $b = 4$ in Theorem 2) that helps us to achieve zero knowledge. We add new random elements to $\boldsymbol{z}^*$ and zero elements to $\mathbf{W}$; the latter are needed not to disturb the knowledge-soundness proof. More precisely, for $\boldsymbol{r}_z \leftarrow\!\!\$\, \mathbb{F}^b$, let

$$\boldsymbol{z}_{\mathbf{l}} := \begin{pmatrix} 1 \\ \mathbb{x} \\ \boldsymbol{z}_{\mathsf{a}} \end{pmatrix} \in \mathbb{F}^m \ , \ \boldsymbol{z}_{\mathbf{r}} := \begin{pmatrix} \mathbf{1}_{m_0+1} \\ \boldsymbol{z}_{\mathsf{b}} \end{pmatrix} \in \mathbb{F}^m \ , \text{ and } \boldsymbol{z} := \begin{pmatrix} \boldsymbol{z}_{\mathbf{l}} \\ \boldsymbol{z}_{\mathbf{r}} \\ \boldsymbol{r}_z \end{pmatrix} \ .$$

Let $\mathbf{I}^b := \left(\begin{smallmatrix} \mathbf{I}_m & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{smallmatrix}\right)$ and $\mathbf{M}^b := \left(\begin{smallmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right)$ be $n_h \times n_h$ matrices. Let $\boldsymbol{z}' := \left(\begin{smallmatrix} \boldsymbol{z}_{\mathbf{r}} \\ \mathbf{0}_{n_h - m} \end{smallmatrix}\right)$. Our goal is to show

$$\mathbf{W}^b \cdot \left(\begin{smallmatrix} \boldsymbol{z} \\ \boldsymbol{z} \circ \boldsymbol{z}' \end{smallmatrix}\right) = \mathbf{0} \ , \tag{3}$$

where $\mathbf{W}^b = \left(\mathbf{I}^b \| -\mathbf{M}^b\right)$. Clearly, Eq. (3) is equivalent to $\mathbf{W} \cdot \boldsymbol{z}^* = \mathbf{0}$.

Next, Eq. (3) holds iff $\mathbf{I}^b \boldsymbol{z} - \mathbf{M}^b (\boldsymbol{z} \circ \boldsymbol{z}') = \mathbf{0}$, i.e.,

$$\forall x \in \mathbb{H}. P[x] := \textstyle\sum_{y \in \mathbb{H}} \left(\mathbf{I}^b[x,y] - \mathbf{M}^b[x,y] z'[y]\right) z[y] = 0 \ .$$

Language of Polynomials. Next, we replace vectors with their low-degree encodings, with say

$$z(Y) := \textstyle\sum_{\chi \in \mathbb{H}} z[\chi] L^{\mathbb{H}}_{\chi}(Y) \in \mathbb{F}_{\le n_h - 1}[Y] \ .$$

Let $\Lambda^b_{\mathbb{H}}(X,Y)$ and M^b be polynomials, fixed later, that interpolate the matrices $\mathbf{I}^b$ and $\mathbf{M}^b$. That is, $\Lambda^b_{\mathbb{H}}(x,y) = \mathbf{I}^b[x,y]$ and $M^b(x,y) = \mathbf{M}^b[x,y]$ for

$x, y \in \mathbb{H}$. Thus, $\mathbf{I}^b[x,y]z[y] = \Lambda^b_{\mathbb{H}}(x,y)z(y)$ for any $x, y \in \mathbb{H}$. Moreover, $\mathbf{M}^b[x,y]z'[y]z[y] = M^b(X,Y)z(Y\omega^m)z(Y)$. Really, $z(y\omega^m) = z[y\omega^m] = z'[y]$ for $y \in \{\omega^0, \ldots, \omega^{m-1}\}$. Notably, for $x \in \mathbb{H}$ and $y \in \{\omega^m, \ldots, \omega^{n_h-1}\}$, the value of $z[y\omega^m]$ does not matter since it is multiplied by $M^b(x,y) = 0$.

Thus, Eq. (3) is equivalent to $\forall x \in \mathbb{H}.P(x) = 0$, where

$$\psi(X,Y) := \left(\Lambda^b_{\mathbb{H}}(X,Y) - M^b(X,Y)z(Y\omega^m)\right) z(Y) \ , \tag{4}$$

$$P(X) := \textstyle\sum_{y\in\mathbb{H}} \psi(X,y) \ . \tag{5}$$

To simplify this further, $\Lambda^b_{\mathbb{H}}(X,Y)$ and $M^b(X,Y)$ have to satisfy additional conditions that we define in the rest of this subsection.

Interpolating $\mathbf{I}^b$. Following [CFF+21], we interpolate $\mathbf{I}$ with the function

$$\Lambda_{\mathbb{H}}(X,Y) := \tfrac{Z_{\mathbb{H}}(X)Y - Z_{\mathbb{H}}(Y)X}{n_h(X-Y)} \ . \tag{6}$$

$\Lambda_{\mathbb{H}}$ satisfies the following properties: (1) $\Lambda_{\mathbb{H}}(x,y)$ is PPT computable, (2) $\Lambda_{\mathbb{H}}$ is a polynomial, (3) $\Lambda_{\mathbb{H}}$ is symmetric, $\Lambda_{\mathbb{H}}(X,Y) = \Lambda_{\mathbb{H}}(Y,X)$, (4) $\Lambda_{\mathbb{H}}(x,y)$ interpolates $\mathbf{I}$ over $\mathbb{H}^2$, i.e., $\forall x, y \in \mathbb{H}.\Lambda_{\mathbb{H}}(x,y) = \mathbf{I}[x,y]$. (5) $\Lambda_{\mathbb{H}}(x,y) = L^{\mathbb{H}}_x(y)$ for any $x \in \mathbb{H}, y \in \mathbb{F}$. Thus, $\{\Lambda_{\mathbb{H}}(x,Y)\}_{x\in\mathbb{H}}$ is a basis of $\mathbb{F}_{\le|\mathbb{H}|-1}[Y]$.

It is natural to define the interpolating polynomial of $\mathbf{I}^b$ as

$$\Lambda^b_{\mathbb{H}}(X,Y) := \Lambda_{\mathbb{H}}(X,Y) - \textstyle\sum_{i=1}^{b} \ell^{\mathbb{H}}_{n_h-b+i}(X)\ell^{\mathbb{H}}_{n_h-b+i}(Y) \ .$$

Clearly, if b is small, then $\Lambda^b_{\mathbb{H}}(X,Y)$ is efficiently computable. Moreover, $\Lambda^b_{\mathbb{H}}(X,Y)$ is symmetric since $\Lambda_{\mathbb{H}}(X,Y)$ is symmetric.

Interpolating $\mathbf{M}^b$. We use the sparse encoding of $\mathbf{M}^b$ from [CHM+20] that keeps track of the matrix's non-zero entries. Let $\mathcal{NZ} := \left\{(i,j) \in \mathbb{H}\times\mathbb{H} : \mathbf{M}^b[i,j] \neq 0\right\}$ be the set of indices where $\mathbf{M}^b$ is non-zero. Let $\mathbb{K}$ be the minimal-size multiplicative subgroup of $\mathbb{F}$ such that $n_k := |\mathbb{K}| \ge |\mathcal{NZ}|$.[7]

We encode $\mathbf{M}^b$ by using polynomials row and col to keep track of the indices of its non-zero entries while using a polynomial val for the values of these entries. That is, $\forall\kappa \in \mathbb{K}$, $\mathsf{row}(\kappa) \in \mathbb{H}$ is the row index of the κth element of $\mathcal{NZ}$, $\mathsf{col}(\kappa) \in \mathbb{H}$ is the column index of the κth element of $\mathcal{NZ}$, and $\mathsf{val}(\kappa) = \mathbf{M}^b[\mathsf{row}(\kappa), \mathsf{col}(\kappa)] \in \mathbb{F}$ is the corresponding matrix entry. Let

$$\mathsf{row}(Z) := \textstyle\sum_{\kappa\in\mathbb{K}} \mathsf{row}(\kappa) L^{\mathbb{K}}_{\kappa}(Z) \in \mathbb{F}_{\le n_k-1}[Z]$$

be the low-degree extension of the vector $(\mathsf{row}(\kappa))_{\kappa\in\mathbb{K}}$. Let $\mathsf{col}(Z)$ and $\mathsf{val}(Z)$ be the low-degree extensions of $(\mathsf{col}(\kappa))_{\kappa\in\mathbb{K}}$ and $(\mathsf{val}(\kappa))_{\kappa\in\mathbb{K}}$. Let $\mathsf{zcv}(Z)$, $\mathsf{rcv}(Z)$, $\mathsf{zrow}(Z)$, $\mathsf{zcol}(Z)$, $\mathsf{rc}(Z)$, and $\mathsf{zrc}(Z)$ be the low-degree encodings of $Z\mathsf{col}(Z)\mathsf{val}(Z)$, $\mathsf{row}(Z)\mathsf{col}(Z)\mathsf{val}(Z)$, $Z\mathsf{row}(Z)$, $Z\mathsf{col}(Z)$, $\mathsf{row}(Z)\mathsf{col}(Z)$, and $Z\mathsf{row}(Z)\mathsf{col}(Z)$. For example,

$$\mathsf{rcv}(Z) := \textstyle\sum_{\kappa\in\mathbb{K}} \mathsf{row}(\kappa)\mathsf{col}(\kappa)\mathsf{val}(\kappa) L^{\mathbb{K}}_{\kappa}(Z) \in \mathbb{F}_{\le n_k-1}[Z] \ .$$

[7] $\mathbb{H}$ and $\mathbb{K}$ can be arbitrary subsets of $\mathbb{F}$, but the most efficient algorithms are known when they are multiplicative subgroups. One can assume $\mathbb{K} = \mathbb{H}$ by adding all-zero rows and columns to the matrix, but we generally do not need that $\mathbb{K} = \mathbb{H}$. Keeping $|\mathbb{K}|$ and $|\mathbb{H}|$ flexible allows us to achieve different trade-offs.

We define $M^b \in \mathbb{F}[X,Y]$, so that

$$\forall x,y \in \mathbb{H}.M^b(x,y) = \textstyle\sum_{\kappa\in\mathbb{K}} \mathsf{val}(\kappa)\Lambda_{\mathbb{H}}(\mathsf{row}(\kappa),x)\Lambda_{\mathbb{H}}(\mathsf{col}(\kappa),y) \ .$$

From the definition of $\Lambda_{\mathbb{H}}$, $\Lambda_{\mathbb{H}}(\mathsf{row}(\kappa),x) = (\mathcal{Z}_{\mathbb{H}}(\mathsf{row}(\kappa))x - \mathcal{Z}_{\mathbb{H}}(x)\mathsf{row}(\kappa))/(n_h(\mathsf{row}(\kappa) - x))$. Since $\mathcal{Z}_{\mathbb{H}}(\mathsf{row}(\kappa)) = 0$, $\Lambda_{\mathbb{H}}(\mathsf{row}(\kappa),x) = \mathcal{Z}_{\mathbb{H}}(x)\mathsf{row}(\kappa)/(n_h(x - \mathsf{row}(\kappa)))$. Similarly, $\Lambda_{\mathbb{H}}(\mathsf{col}(\kappa),y) = \mathcal{Z}_{\mathbb{H}}(y)\mathsf{col}(\kappa)/(n_h(y - \mathsf{col}(\kappa)))$. Thus,

$$\begin{aligned}\forall x,y \in \mathbb{H}.M^b(x,y) &= \textstyle\sum_{\kappa\in\mathbb{K}} \mathsf{val}(\kappa) \cdot \frac{\mathcal{Z}_{\mathbb{H}}(x)\mathsf{row}(\kappa)}{n_h(x-\mathsf{row}(\kappa))} \cdot \frac{\mathcal{Z}_{\mathbb{H}}(y)\mathsf{col}(\kappa)}{n_h(y-\mathsf{col}(\kappa))} \\ &= \tfrac{\mathcal{Z}_{\mathbb{H}}(x)\mathcal{Z}_{\mathbb{H}}(y)}{n_h^2} \textstyle\sum_{\kappa\in\mathbb{K}} \frac{\mathsf{rcv}(\kappa)}{(x-\mathsf{row}(\kappa))(y-\mathsf{col}(\kappa))} \\ &\overset{(*)}{=} \tfrac{\mathcal{Z}_{\mathbb{H}}(x)\mathcal{Z}_{\mathbb{H}}(y)}{n_h^2} \textstyle\sum_{\kappa\in\mathbb{K}} \frac{\mathsf{rcv}(\kappa)}{xy-x\mathsf{col}(\kappa)-y\mathsf{row}(\kappa)+\mathsf{rc}(\kappa)} \ ,\end{aligned}$$

where say (*) follows from $\forall\kappa \in \mathbb{K}.\mathsf{rc}(\kappa) = \mathsf{col}(\kappa)\mathsf{row}(\kappa)$. Thus, we define

$$M^b(X,Y) := \tfrac{\mathcal{Z}_{\mathbb{H}}(X)\mathcal{Z}_{\mathbb{H}}(Y)}{n_h^2} \textstyle\sum_{\kappa\in\mathbb{K}} \frac{\mathsf{rcv}(\kappa)}{XY-X\mathsf{col}(\kappa)-Y\mathsf{row}(\kappa)+\mathsf{rc}(\kappa)} \ . \tag{7}$$

Since $\deg_X M^b(X,Y) \le |\mathbb{H}| - 1$, $\forall y \in \mathbb{H}.M^b(X,y) = \sum_{\chi\in\mathbb{H}} M^b(\chi,y)\Lambda_{\mathbb{H}}(\chi,X)$.

Getting to Sumcheck. Next, we show that, under mild conditions on interpolating matrices satisfied by the above encodings, $\forall x \in \mathbb{H}.P(x) = 0$ (and thus also Eq. (3)) is equivalent to $\sum_{y\in\mathbb{H}} \psi(X,y) = 0$.

Lemma 4. *Assume* $\deg_X \Lambda_{\mathbb{H}}(X,Y), \deg_X M^b(X,Y) \le |\mathbb{H}| - 1$. *Then,* $\forall x \in \mathbb{H}.P(x) = 0$ *iff* $\sum_{y\in\mathbb{H}} \psi(X,y) = 0$.

Proof. ($\Rightarrow$) Assume $\forall x \in \mathbb{H}.P(x) = 0$. Then,

$$\begin{aligned}\textstyle\sum_{y\in\mathbb{H}} \psi(X,y) &\overset{4}{=} \textstyle\sum_{y\in\mathbb{H}} (\Lambda^b_{\mathbb{H}}(X,y) - M^b(X,y)z(y\omega^m))z(y) \\ &\overset{(*)}{=} \textstyle\sum_{x\in\mathbb{H}}\sum_{y\in\mathbb{H}} (\Lambda^b_{\mathbb{H}}(x,y) - M^b(x,y)z(y\omega^m))z(y)L_x(X) \\ &\overset{4}{=} \textstyle\sum_{x\in\mathbb{H}}\sum_{y\in\mathbb{H}} \psi(x,y)L_x(X) \overset{5}{=} \sum_{x\in\mathbb{H}} P(x)L_x(X) \overset{(**)}{=} 0 \ ,\end{aligned}$$

where (*) follows from the low degree of $\Lambda^b_{\mathbb{H}}(X,Y)$ and $M^b(X,Y)$, and (**) follows from $\forall x \in \mathbb{H}.P(x) = 0$.
($\Leftarrow$) Let $\sum_{y\in\mathbb{H}} \psi(X,y) = 0$. By Eq. (5), $\forall x \in \mathbb{H}.P(x) = \sum_{y\in\mathbb{H}} \psi(x,y) = 0$. □

To enable efficient verification that the public input was correctly computed, the prover transmits $[\tilde{z}(\sigma)]_1$, for the polynomial $\tilde{z}(Y)$ defined as follows. Let

$$\begin{aligned}\mathcal{Z}_{\mathsf{inp}}(Y) &:= \textstyle\prod_{i=1}^{m_0+1}(Y-\omega^{i-1})(Y-\omega^{m+i-1}) \in \mathbb{F}_{\le 2(m_0+1)}[Y] \ , \\ \mathsf{inp}(Y) &:= \ell^{\mathbb{H}}_1(Y) + \textstyle\sum_{i=1}^{m_0} \mathbb{x}_i \ell^{\mathbb{H}}_{i+1}(Y) + \sum_{i=1}^{m_0+1} \ell^{\mathbb{H}}_{m+i}(Y) \in \mathbb{F}_{\le n_h-1}[Y] \ , \\ \tilde{z}(Y) &:= \textstyle\sum_{i=1}^{m-m_0-1} \boldsymbol{z}_{\mathsf{a}}[i] \frac{\ell^{\mathbb{H}}_{m_0+1+i}(Y)}{\mathcal{Z}_{\mathsf{inp}}(Y)} + \sum_{i=1}^{m-m_0-1} \boldsymbol{z}_{\mathsf{b}}[i] \frac{\ell^{\mathbb{H}}_{m+m_0+1+i}(Y)}{\mathcal{Z}_{\mathsf{inp}}(Y)} + \\ &\quad \textstyle\sum_{i=1}^{b} \boldsymbol{r}_z[i] \frac{\ell^{\mathbb{H}}_{2m+i}(Y)}{\mathcal{Z}_{\mathsf{inp}}(Y)} \ .\end{aligned} \tag{8}$$

Since $\ell_i^{\mathbb{H}}(Y) = \prod_{j\neq i}(Y-\omega^{j-i})/(\omega^{i-i}-\omega^{j-i})$, $\tilde{z}(Y) \in \mathbb{F}_{\leq n_h-2m_0-3}[Y]$. Thus, $\mathcal{Z}_{\mathsf{inp}}(Y)\tilde{z}(Y) = \sum_{i=1}^{m-m_0-1} \boldsymbol{z}_{\mathsf{a}}[i]\ell^{\mathbb{H}}_{m_0+1+i}(Y) + \sum_{i=1}^{m-m_0-1} \boldsymbol{z}_{\mathsf{b}}[i]\ell^{\mathbb{H}}_{m+m_0+1+i}(Y) + \sum_{i=1}^{b} \boldsymbol{r}_z[i]\ell^{\mathbb{H}}_{2m+i}(Y)$ interpolates $(\mathbf{0}^\top_{m_0+1}\|\boldsymbol{z}_{\mathsf{a}}^\top\|\mathbf{0}^\top_{m_0+1}\|\boldsymbol{z}_{\mathsf{b}}^\top\|\boldsymbol{r}_z^\top)^\top$. Moreover,

$$z(Y) = \mathcal{Z}_{\mathsf{inp}}(Y)\tilde{z}(Y) + \mathsf{inp}(Y) \in \mathbb{F}_{\leq n_h-1}[Y] \ . \tag{9}$$

Thus, the existence of $\tilde{z}(Y)$, such that Eq. (9) holds, guarantees that $z(Y)$ interpolates $(1\|\mathbb{x}^\top\|\boldsymbol{z}_{\mathsf{a}}^\top\|\mathbf{1}^\top_{m_0+1}\|\boldsymbol{z}_{\mathsf{b}}^\top\|\boldsymbol{r}_z^\top)^\top$ for some $\boldsymbol{z}_{\mathbf{a}}$, $\boldsymbol{z}_{\mathbf{b}}$, and $\boldsymbol{r}_z$.

4.2 From Sumcheck to $\mathfrak{Vampire}$

According to the preceding discussion, one can handle R1CSLite by having a zk-SNARK that proves that $\sum_{y\in\mathbb{H}}\psi(X,y) = 0$. We will do construct the latter in the current subsection. We replace X with a random α chosen by the verifier, obtaining the polynomial $\psi_\alpha(Y) := \psi(\alpha, Y)$. We use $\mathfrak{Count}$ to show that $\sum_{y\in\mathbb{H}}\psi_\alpha(y) = 0$. For this, as in Section 3, the prover computes a polynomial ψ_{ipc} and the verifier checks $\varphi(Y) := \psi_\alpha(Y)S(Y) - \psi_{\mathsf{ipc}}(Y)$ is a zero polynomial. The latter can be checked by KZG-opening all involved polynomials (e.g., $\tilde{z}(Y)$; see Eq. (4)), but this is inefficient. Instead, the prover KZG-opens $\tilde{z}(Y)$ at $Y = \beta\omega^m$ and $\Phi(Y)$, $M^b(\alpha, Y)$ at $Y = \beta$, where (1) Φ is a polynomial defined so that $\Phi(\beta) = \varphi(\beta) = 0$, and (2) one can verify efficiently the correctness, given $\tilde{z}(\beta\omega^m)$ and $v_M = M^b(\alpha,\beta)$. This requires one to open a polynomial related to the ILV-opening of $\psi_\alpha(Y)$. We aggregate KZG-openings by using the technique of [BDFG20]. Finally, we use another sumcheck to check the correctness of v_M; this step is complicated, but it follows closely Marlin [CHM+20] and Lunar [CFF+21].

To simplify some formulas, we assume always $n_h > 3$. This is w.l.o.g., since $n_h = 2m + b$, $m \geq 1$, and $b \geq 2$.

Details. Let $\alpha \in \mathbb{F}\setminus\mathbb{H}$ be a random value chosen by the verifier. (We will explain later why $\alpha \notin \mathbb{H}$.) To test that $\sum_{y\in\mathbb{H}}\psi(X,y) = 0$, we define

$$\psi_\alpha(Y) := \psi(\alpha, Y) \in \mathbb{F}_{\leq d}[Y] \ .$$

From Eqs. (4) and (9), we get

$$\psi_\alpha(Y) = \big(\Lambda^b_{\mathbb{H}}(\alpha, Y) - M^b(\alpha, Y)z(Y\omega^m)\big)\cdot(\mathcal{Z}_{\mathsf{inp}}(Y)\tilde{z}(Y) + \mathsf{inp}(Y)) \ .$$

Clearly,

$$d := \deg\psi_\alpha = 3(n_h - 1) \ . \tag{10}$$

We use $\mathfrak{Count}$ to prove that $\sum_{y\in\mathbb{H}}\psi_\alpha(y) = 0$. As in Lemma 2, we define

$$\begin{aligned} S(Y) &:= \textstyle\sum_{i=0}^{\lfloor d/n_h\rfloor} Y^{d_{\mathsf{gap}}-n_h i} \in \mathbb{F}_{\leq d_{\mathsf{gap}}}[Y] \ , \\ \psi_{\mathsf{ipc}}(Y) &:= \psi_\alpha(Y)S(Y) \in \mathsf{PolyPunc}_{\mathbb{F}}(d, d_{\mathsf{gap}}, Y) \ . \end{aligned} \tag{11}$$

Here, $d_{\mathsf{gap}} \in \mathbb{N}$ is an integer, such that $S(Y)$ and $\psi_{\mathsf{ipc}}(Y)$ are polynomials, i.e., $d_{\mathsf{gap}} \geq n_h \cdot \lfloor d/n_h \rfloor$. Thus, we need $d_{\mathsf{gap}} \geq n_h \cdot \lfloor d/n_h \rfloor = n_h \cdot \lfloor 3(n_h-1)/n_h \rfloor = 2n_h$. (This holds for $n_h \geq 3$.) Taking into account later considerations, we set

$$d_{\mathsf{gap}} := 3(n_h - 1) \ . \tag{12}$$

Clearly, $d_{\mathsf{gap}} \geq 2n_h$ if $n_h \geq 3$. Hence, $S(Y) = Y^{d_{\mathsf{gap}}} + Y^{d_{\mathsf{gap}}-n_h} - Y^{d_{\mathsf{gap}}-2n_h}$.

According to Lemma 2, we need to check that the coefficient of $Y^{d_{\mathsf{gap}}}$ in $\psi_\alpha(Y)S(Y)$ is 0. We do it by checking that

(i) $\psi_{\mathsf{ipc}}(Y) \in \mathsf{PolyPunc}_{\mathbb{F}}(d, d_{\mathsf{gap}}, Y)$, and
(ii) $\psi_{\mathsf{ipc}}(Y)$ is the correct ILV-opening polynomial, i.e.,

$$\begin{aligned}\varphi(Y) :=& \psi_\alpha(Y)S(Y) - \psi_{\mathsf{ipc}}(Y) \\ =& \left(\Lambda^b_{\mathbb{H}}(\alpha, Y) - M^b(\alpha, Y) z(Y\omega^m)\right) \left(\mathcal{Z}_{\mathsf{inp}}(Y)\tilde{z}(Y) + \mathsf{inp}(Y)\right) \cdot S(Y) - \psi_{\mathsf{ipc}}(Y)\end{aligned}$$

is a zero polynomial.

The prover sends to the verifier commitments to $\tilde{z}(Y)$ and $\psi_{\mathsf{ipc}}(Y)$. Checking i is free in the pairing-based setting. To check ii, we verify that $\varphi(\beta) = 0$, where $\beta \in \mathbb{F} \setminus \mathbb{H}$ is a random point chosen by the verifier. (We will explain later why $\beta \notin \mathbb{H}$; in fact, β is even more restricted.) More precisely, we verify that $\varphi(\beta) = 0$, where $M^b(\alpha, \beta)$ is substituted by a value v_M computed by the prover. We first describe how to check that $\varphi(\beta) = 0$, assuming v_M is correct. After that, we use another sumcheck instantiation to prove that v_M is correctly computed.

First: checking $\varphi(\beta) = 0$. A straightforward check that $\varphi(\beta) = 0$ requires, on top of sending v_M, the prover to KZG-open $\tilde{z}(Y)$ both at $Y = \beta$ and $Y = \beta\omega^m$ and $\psi_{\mathsf{ipc}}(Y)$ at $Y = \beta$. (The verifier can efficiently evaluate other polynomials like $\Lambda^b_{\mathbb{H}}(X, Y)$, $\mathcal{Z}_{\mathsf{inp}}(Y)$, and $S(Y)$ at $(X, Y) = (\alpha, \beta)$ herself.) To improve on efficiency, we *implicitly* KZG-commit to the polynomial Φ, where

$$\begin{aligned}\Psi(Y) :=& \left(\Lambda^b_{\mathbb{H}}(\alpha, \beta) - v_M \cdot z(\beta\omega^m)\right) \left(\mathcal{Z}_{\mathsf{inp}}(\beta)\tilde{z}(Y) + \mathsf{inp}(\beta)\right) \in \mathbb{F}_{\leq n_h - 2m_0 - 3}[Y] \ , \\ \Phi(Y) :=& (\Psi(Y)S(Y) - \psi_{\mathsf{ipc}}(Y))/S(Y) = \Psi(Y) - \psi_\alpha(Y) \in \mathbb{F}_{\leq d}[Y] \ .\end{aligned}$$

Here, $\Psi(Y)$ is obtained from $\psi_\alpha(Y)$ by replacing all but one occurrences of Y with β, and Φ is a (low-degree) polynomial that satisfies $\Phi(\beta) = \varphi(\beta) = 0$.

We open KZG-commitments to $\tilde{z}(Y)$ at $Y = \beta\omega^m$ (needed to compute $z(\beta\omega^m)$) and $\Phi(Y)$ at $Y = \beta$. Here, one can open and verify a commitment to Φ since we have KZG-commitments to $\tilde{z}$ and ψ_{ipc}, KZG is homomorphic, and the verifier knows all other information present in Φ like $\mathsf{inp}(\beta)$ and v_M. More precisely, the prover batch-opens the two KZG-commitments by computing the KZG-opening polynomials

$$\begin{aligned}\tilde{z}_{\mathsf{pc}}(Y) :=& \tfrac{\tilde{z}(Y) - \tilde{z}(\beta\omega^m)}{Y - \beta\omega^m} \in \mathbb{F}_{\leq n_h - 2m_0 - 4}[Y] \ , \\ \Phi_{\mathsf{pc}}(Y) :=& \tfrac{\Phi(Y) - \Phi(\beta)}{Y - \beta} = \tfrac{\Psi(Y) - \psi_\alpha(Y)}{Y - \beta} \in \mathbb{F}_{\leq d-1}[Y] \ .\end{aligned}$$

The prover batches the openings as $[B_{\mathsf{pc}}(\sigma)]_1 \leftarrow [\tilde{z}_{\mathsf{pc}}(\sigma) + \Phi_{\mathsf{pc}}(\sigma)]_1$. Notably, since the two polynomial openings are at different locations β and $\beta\omega^m$, one

does not have to randomize this check. (We motivate it in Section 5.1.) The latter is a general fact, not mentioned in [BDFG20] and is thus an independent contribution.

The prover also sends $v_z \leftarrow \tilde{z}(\beta\omega^m) \in \mathbb{F}$. Since $\Phi(\beta) = 0$, $\Phi(\beta)$ is not transferred. The verifier checks that $[\tilde{z}(\sigma) - v_z]_1 \bullet [\sigma - \beta]_2 + [\Phi(\sigma)]_1 \bullet [\sigma - \beta\omega^m]_2 = [B_{\mathsf{pc}}(\sigma)]_1 \bullet [(\sigma-\beta)(\sigma-\beta\omega^m)]_2$, where $[\Phi(\sigma)]_1[\Psi(\sigma)]_1 - [\psi_\alpha(\sigma)]_1$. Since the verifier does not know $[\psi_\alpha(\sigma)]_1$ but knows $[\psi_{\mathsf{ipc}}(\sigma)]_1 = [\psi_\alpha(\sigma)S(\sigma)]_1$, we multiply each term of the verification equation by $S(\sigma)$, obtaining the check (♯♯) in Fig. 2.

Thus, instead of KZG-opening three polynomials, we do a single KZG-commitment and batch-opening. It reduces the communication from three $\mathbb{G}_1$ elements and three field elements (three openings and $(\tilde{z}(\beta), \tilde{z}(\beta\omega^m), \psi_{\mathsf{ipc}}(\beta))$) to one $\mathbb{G}_1$ element and one field element ($[B_{\mathsf{pc}}(\sigma)]_1$ and v_z).

Note that the prover has to compute $[\Phi_{\mathsf{pc}}(\sigma)]_1 \leftarrow [(\Phi(\sigma) - \psi_\alpha(\sigma))/(\sigma - \beta)]_1$, where $\Phi_{\mathsf{pc}} \in \mathbb{F}_{\leq d-1}[Y]$ and σ is a trapdoor. For $\mathfrak{Count}$ to be secure, the SRS cannot contain $[\sigma^{d_{\mathsf{gap}}}]_1$. Hence, we need to assume $d \leq d_{\mathsf{gap}}$. This motivates the choice of $d_{\mathsf{gap}} = 3(n_h - 1)$ in Eq. (10).

Second (correctness of v_M). Here, we use a technique from [CHM+20,CFF+21]. Recall that M^b satisfies Eq. (7). Moreover, $\deg_X M^b(X,Y), \deg_Y M^b(X,Y) \leq n_h - 1$. Thus, $M^b(\alpha,\beta) = \sum_{\kappa\in\mathbb{K}} T(\kappa) \in \mathbb{F}[X,Y]$, where

$$\begin{aligned} \mathsf{num}(Z) &:= \tfrac{\mathcal{Z}_{\mathbb{H}}(\alpha)\mathcal{Z}_{\mathbb{H}}(\beta)}{n_h^2} \cdot \mathsf{rcv}(Z) \in \mathbb{F}_{\leq n_k-1}[Z] \ , \\ \mathsf{den}(Z) &:= \alpha\beta - \alpha\cdot\mathsf{col}(Z) - \beta\cdot\mathsf{row}(Z) + \mathsf{rc}(Z) \in \mathbb{F}_{\leq n_k-1}[Z] \ , \\ T(Z) &:= \tfrac{\mathsf{num}(Z)}{\mathsf{den}(Z)} \in \mathbb{F}(Z) \ . \end{aligned} \tag{13}$$

Here, we need $\mathsf{den}(\kappa) = (\alpha - \mathsf{row}(\kappa))(\beta - \mathsf{col}(\kappa)) \neq 0$ for any $\kappa \in \mathbb{K}$. This explains the choice of $\alpha, \beta \notin \mathbb{H}$.

We use a sumcheck to check that $v_M = M^b(\alpha,\beta)$. We use Aurora's sumcheck since here, it is otherwise as efficient as $\mathfrak{Count}$ but results in a shorter SRS. Let $\widehat{T}(Z) := \sum_{\kappa\in\mathbb{K}} T(\kappa) L_\kappa^{\mathbb{K}}(Z) \in \mathbb{F}_{\leq n_k-1}[Z]$. Clearly, $\mathsf{num}(Z) - \widehat{T}(Z)\mathsf{den}(Z) \equiv 0 \pmod{\mathcal{Z}_{\mathbb{K}}(Z)}$. Since $\sum_{\kappa\in\mathbb{K}} \widehat{T}(\kappa) = v_M$, by Aurora's sumcheck, $\widehat{T}(Z) = ZR(Z) + v_M/n_k$ for some $R \in \mathbb{F}_{\leq n_k-2}[Z]$. Thus, $\mathsf{num}(Z) - (ZR(Z) + v_M/n_k)\mathsf{den}(Z) \equiv 0 \pmod{\mathcal{Z}_{\mathbb{K}}(Z)}$. Since this equality has to hold only when $Z \in \mathbb{K}$, we modify it as follows. For some $Q \in \mathbb{F}_{\leq n_k-3}[Z]$, let

$$\mathsf{num}(Z) - R(Z)\cdot\mathsf{zden}(Z) - v_M/n_k \cdot \mathsf{den}(Z) = Q(Z)\mathcal{Z}_{\mathbb{K}}(Z) \ , \tag{14}$$

where $\mathsf{zden}(Z) := \alpha\beta Z - \alpha\mathsf{zcol}(Z) - \beta\mathsf{zrow}(Z) + \mathsf{zrc}(Z) \in \mathbb{F}_{\leq n_k-1}[Z]$. Thus, $\mathsf{zden}(Z) = Z\mathsf{den}(Z)$ for $Z \in \mathbb{K}$. This rewriting minimizes the degree of polynomials (e.g., $\mathsf{zcol}(Z) \in \mathbb{F}_{\leq n_k-1}[Z]$ while $Z\mathsf{col}(Z) \in \mathbb{F}_{\leq n_k}[Z]$).

The prover commits to R and Q. The verifier can check that Eq. (14) holds on KZG-commitments without opening them, checking that $[1]_1 \bullet ([\mathsf{num}(\sigma)]_2 - v_M/n_k \cdot [\mathsf{den}(\sigma)]_2) = [R(\sigma)]_1 \bullet [\mathsf{zden}(\sigma)]_2 + [Q(\sigma)]_1 \bullet [\mathcal{Z}_{\mathbb{K}}(\sigma)]_2$. When we add to $\mathsf{srs}_{\mathcal{R}}$ elements like $[\mathsf{rcv}(\sigma), \mathsf{col}(\sigma)]_2$, the verifier can compute the elements of $\mathbb{G}_2$ in the last equation since he knows α and β. Thus, polynomials like $[\mathsf{rcv}(\sigma)]_2$

need to be in $\mathsf{srs}_{\mathcal{R}}$ while monomials, needed for the V to be able to compute $\mathsf{srs}_{\mathcal{R}}$, need to be in srs. This explains the definition of $\mathsf{srs}_{\mathcal{R}}$ in Fig. 2.

Finally, one needs to check that $\deg R \leq n_k - 2$. To perform this low-degree test without increasing the argument size, we use a second trapdoor $\tau \leftarrow\!\$\, \mathbb{F}^*$. We add $[(\sigma^i\tau)_{i=0}^{n_k-2}]_2$ to the SRS and transfer $[R(\sigma)\tau, Q(\sigma)\tau]_1$ instead of $[R(\sigma), Q(\sigma)]_1$. This also modifies the verification equations. The idea behind it is that if the SRS contains $[(\sigma^i)_{i\in\mathcal{S}}, (\sigma^i\tau)_{i\in\mathcal{S}'}]_1$, then a verification $[a]_1 \bullet [1]_2 = [b]_1 \bullet [\tau]_2$ guarantees in the AGM that $a \in \operatorname{span}(\sigma^i\tau)_{i\in\mathcal{S}'}$.

4.3 Description of $\mathfrak{Vampire}$

In Fig. 2, we describe interactive $\mathfrak{Vampire}$, the new succinct interactive zero-knowledge argument with a specializable universal SRS. Since this argument is public-coin and has a constant number of rounds, we can apply the Fiat-Shamir heuristic ([FS87], we omit the details) to obtain the zk-SNARK $\mathfrak{Vampire}$.

We sample the challenge β from the set

$$C_\beta := \{\beta \in \mathbb{F} \mid \beta \notin (\mathbb{H} \cup \{0, \sigma, \sigma/\omega^m\}) \wedge S(\beta) \neq 0 \wedge S(\beta\omega^m) \neq 0\} \ .$$

We need that $\beta \notin \{\sigma, \sigma/\omega^m\}$ since otherwise the prover cannot open polynomial commitments. One can efficiently verify that $\beta \notin \{\sigma, \sigma/\omega^m\}$, given $[\sigma]_1$ from the SRS. In addition, in the knowledge-soundness proof we need that $S(Y)$, $(Y-\beta)$, and $(Y - \beta\omega^m)$ are coprime. Hence, we need that $S(\beta) \neq 0$, $S(\beta\omega^m) \neq 0$, and $\beta \neq 0$. As mentioned previously, $\alpha, \beta \notin \mathbb{H}$ since otherwise $\mathsf{den}(\kappa) = 0$ for any $\kappa \in \mathbb{K}$. Note that if n_h and d_{gap} are much smaller than $|\mathbb{F}|$ (which is typically the case), then $\beta \leftarrow\!\$\, \mathbb{F}$ is contained in C_β with an overwhelming probability. Hence, in practice β can be sampled from $\mathbb{F}$, resulting in only a negligible security risk.

$\mathfrak{Vampire}$'s updatability follows from the facts that $\mathcal{S}_1$ and $\mathcal{S}_2$ consist of monomials and moreover, one can verify the correctness of a SRS efficiently. We will prove the latter in Theorem 3.

4.4 Efficiency

P sends to V five $\mathbb{G}_1$ elements and two field elements. When using the Fiat-Shamir heuristic, the verifier's output elements can be replaced with the output of the random oracle. Thus, the total communication of $\mathfrak{Vampire}$ is $5|\mathbb{G}_1| + 2|\mathbb{F}|$ bits. Assuming that an element of $\mathbb{G}_1$ is 384 bits and and element of $\mathbb{F}$ is 256 bits, the total communication is 2432 bits or 304 bytes.

We primarily optimized the argument size of $\mathfrak{Vampire}$, even if it resulted in worsening other parameters. Still, each parameter, like prover's computation or SRS length, is within a small constant factor of the corresponding parameter in the case of the most efficient known zk-SNARKs. $\mathfrak{Vampire}$ is especially advantageous when $n_k \gg m$. As a function of n_k only, $\mathfrak{Vampire}$ has the best prover's computation and the same SRS length and SRS generation time as any previous updatable zk-SNARK. See Appendix A for a thorough efficiency comparison.

$\mathsf{Pgen}(1^\lambda)$: generate p as usually, assuming that $n_h, n_k \mid (p-1)$

$\mathsf{KGen}(\mathsf{p}, n_h, n_k)$: $\mathcal{S}_1 = \{(X^i)_{i=0:i\neq d_{\mathsf{gap}}}^{d_{\mathsf{gap}}+d}, (X^i X_\tau)_{i=0}^{n_k-2}\}$; $\mathcal{S}_2 = \{(X^i)_{i=0}^{n_k}, ((X^{d_{\mathsf{gap}}+j-n_h i})_{j=0}^2)_{i=0}^{\lfloor d/n_h \rfloor}\}$;
$\sigma, \tau \leftarrow\$ \mathbb{F}^*$; $\mathsf{td} \leftarrow (\sigma, \tau)$; $\mathsf{srs} \leftarrow (\mathsf{p}, n_h, n_k, [g(\sigma,\tau) : g \in \mathcal{S}_1]_1, [g(\sigma,\tau) : g \in \mathcal{S}_2]_2)$

$\mathsf{Derive}(\mathsf{srs}, \mathcal{I}_{\mathsf{r1cslite}})$: $\mathsf{ek}_\mathcal{R} \leftarrow (\mathsf{p}, \mathcal{I}_{\mathsf{r1cslite}}, [g(\sigma,\tau) : g \in \mathcal{S}_1]_1)$;
$\mathsf{srs}_\mathcal{R} \leftarrow [\mathsf{rcv}(\sigma), \mathsf{col}(\sigma), \mathsf{row}(\sigma), \mathsf{rc}(\sigma), \mathsf{zcol}(\sigma), \mathsf{zrow}(\sigma), \mathsf{zrc}(\sigma), S(\sigma), \sigma S(\sigma), \sigma^2 S(\sigma), \mathcal{Z}_\mathbb{K}(\sigma)]_2$
$\mathsf{vk}_\mathcal{R} \leftarrow (\mathsf{p}, \mathcal{I}_{\mathsf{r1cslite}}, [1,\tau]_1, [1,\sigma]_2, \mathsf{srs}_\mathcal{R})$

$\mathsf{P}(\mathsf{ek}_\mathcal{R}, \mathbb{x}, \mathbb{w})$ $\qquad$ $\mathsf{V}(\mathsf{vk}_\mathcal{R}, \mathbb{x})$

Init

Both do: $\mathcal{Z}_{\mathsf{inp}}(Y) \leftarrow \prod_{i=1}^{m_0+1}(Y - \omega^{i-1})(Y - \omega^{m+i-1})$; $\boldsymbol{r}_z \leftarrow\$ \mathbb{F}^b$;

Both do: $\mathsf{inp}(Y) \leftarrow \ell_1^\mathbb{H}(Y) + \sum_{i=1}^{m_0} \mathbb{x}_i \ell_{i+1}^\mathbb{H}(Y) + \sum_{i=1}^{m_0+1} \ell_{m+i}^\mathbb{H}(Y) \in \mathbb{F}_{\leq n_h-1}[Y]$

$\boldsymbol{r}_z \leftarrow\$ \mathbb{F}^b$; $\tilde{z}(Y) \leftarrow \sum_{i=1}^{m-m_0-1} \boldsymbol{z}_\mathsf{a}[i] \frac{\ell_{m_0+1+i}^\mathbb{H}(Y)}{\mathcal{Z}_{\mathsf{inp}}(Y)} + \sum_{i=1}^{m-m_0-1} \boldsymbol{z}_\mathsf{b}[i] \frac{\ell_{m+m_0+1+i}^\mathbb{H}(Y)}{\mathcal{Z}_{\mathsf{inp}}(Y)} + \sum_{i=1}^{b} \boldsymbol{r}_z[i] \frac{\ell_{2m+i}^\mathbb{H}(Y)}{\mathcal{Z}_{\mathsf{inp}}(Y)}$

$z(Y) \leftarrow \mathcal{Z}_{\mathsf{inp}}(Y)\tilde{z}(Y) + \mathsf{inp}(Y)$; // $\tilde{z}(Y) \in \mathbb{F}_{\leq n_h-2m_0-3}[Y]$; $z(Y) \in \mathbb{F}_{\leq n_h-1}[Y]$

$[\tilde{z}(\sigma)]_1 \rightarrow$

$\leftarrow \alpha$ $\qquad$ $\alpha \leftarrow\$ \mathbb{F} \setminus \mathbb{H}$

Abort if $\alpha \notin \mathbb{F} \setminus \mathbb{H}$

$\mathfrak{Count}$: $\sum_{y\in\mathbb{H}} \psi_\alpha(y) = 0$ for $\psi_\alpha(Y) = \left(\Lambda_\mathbb{H}^b(\alpha, Y) - M^b(\alpha, Y) z(Y\omega^m)\right) z(Y)$

$S(Y) \leftarrow \sum_{i=0}^{\lfloor d/n_h \rfloor} Y^{d_{\mathsf{gap}}-n_h i} \in \mathbb{F}_{\leq d_{\mathsf{gap}}}[Y]$; $\psi_{\mathsf{ipc}}(Y) \leftarrow \psi_\alpha(Y) S(Y) \in \mathsf{PolyPunc}_\mathbb{F}(d, d_{\mathsf{gap}}, Y)$

$[\psi_{\mathsf{ipc}}(\sigma)]_1 \rightarrow$

$\leftarrow \beta$ $\qquad$ $\beta \leftarrow\$ C_\beta$

Aurora's sumcheck for $\sum_{\kappa\in\mathbb{K}} (\mathsf{num}(\kappa)/\mathsf{den}(\kappa)) = v_M = M^b(\alpha, \beta)$

Abort if $\beta \notin C_\beta$; $v_M \leftarrow M^b(\alpha,\beta) \in \mathbb{F}$; $v_z \leftarrow \tilde{z}(\beta\omega^m) \in \mathbb{F}$

Compute $R \in \mathbb{F}_{\leq n_k-2}[Z], Q \in \mathbb{F}_{\leq n_k-3}[Z]$, such that
$\mathsf{num}(Z) - R(Z)\mathsf{zden}(Z) - v_M/n_k \cdot \mathsf{den}(Z) = Q(Z)\mathcal{Z}_\mathbb{K}(Z)$

$z(\beta\omega^m) \leftarrow \mathcal{Z}_{\mathsf{inp}}(\beta\omega^m) v_z + \mathsf{inp}(\beta\omega^m)$

$\Psi(Y) \leftarrow \left(\Lambda_\mathbb{H}^b(\alpha,\beta) - v_M \cdot z(\beta\omega^m)\right)(\mathcal{Z}_{\mathsf{inp}}(\beta)\tilde{z}(Y) + \mathsf{inp}(\beta)) \in \mathbb{F}_{\leq n_h-m_0-3}[Y]$

$\tilde{z}_{\mathsf{pc}}(Y) \leftarrow (\tilde{z}(Y) - v_z)/(Y - \beta\omega^m) \in \mathbb{F}_{\leq n_h-2m_0-4}[Y]$

$\Phi_{\mathsf{pc}}(Y) \leftarrow (\Psi(Y) - \psi_\alpha(Y))/(Y-\beta) \in \mathbb{F}_{\leq d-1}[Y]$

$B_{\mathsf{pc}}(Y) \leftarrow \tilde{z}_{\mathsf{pc}}(Y) + \Phi_{\mathsf{pc}}(Y) \in \mathbb{F}_{\leq d-1}[Y]$

$v_z, v_M, [R(\sigma)\tau, Q(\sigma)\tau, B_{\mathsf{pc}}(\sigma)]_1 \rightarrow$

$[\mathsf{num}(\sigma)]_2 \leftarrow \mathcal{Z}_\mathbb{H}(\alpha)\mathcal{Z}_\mathbb{H}(\beta)/n_h^2 \cdot [\mathsf{rcv}(\sigma)]_2$

$[\mathsf{den}(\sigma)]_2 \leftarrow \alpha\beta[1]_2 - \alpha[\mathsf{col}(\sigma)]_2 - \beta[\mathsf{row}(\sigma)]_2 + [\mathsf{rc}(\sigma)]_2$

$[\mathsf{zden}(\sigma)]_2 \leftarrow \alpha\beta[\sigma]_2 - \alpha[\mathsf{zcol}(\sigma)]_2 - \beta[\mathsf{zrow}(\sigma)]_2 + [\mathsf{zrc}(\sigma)]_2$

($\sharp$) Check $[\tau]_1 \bullet ([\mathsf{num}(\sigma)]_2 - \frac{v_M}{n_k}[\mathsf{den}(\sigma)]_2) = [R(\sigma)\tau]_1 \bullet [\mathsf{zden}(\sigma)]_2 + [Q(\sigma)\tau]_1 \bullet [\mathcal{Z}_\mathbb{K}(\sigma)]_2$

Sumcheck end

$z(\beta\omega^m) \leftarrow \mathcal{Z}_{\mathsf{inp}}(\beta\omega^m) v_z + \mathsf{inp}(\beta\omega^m)$

$[\Psi(\sigma)]_1 \leftarrow \left(\Lambda_\mathbb{H}^b(\alpha,\beta) - v_M \cdot z(\beta\omega^m)\right)(\mathcal{Z}_{\mathsf{inp}}(\beta)[\tilde{z}(\sigma)]_1 + \mathsf{inp}(\beta)[1]_1)$

($\sharp\sharp$) Check $[\tilde{z}(\sigma) - v_z]_1 \bullet [(\sigma-\beta)S(\sigma)]_2 +$
$\left([\Psi(\sigma)]_1 \bullet [(\sigma-\beta\omega^m)S(\sigma)]_2 - [\psi_{\mathsf{ipc}}(\sigma)]_1 \bullet [\sigma - \beta\omega^m]_2\right) = [B_{\mathsf{pc}}(\sigma)]_1 \bullet [(\sigma-\beta)(\sigma-\beta\omega^m)S(\sigma)]_2$

Sumcheck end

Fig. 2. Interactive $\mathfrak{Vampire}$. $\mathcal{I}_{\mathsf{r1cslite}} = (\mathbb{F}, \mathbb{H}, \mathbb{K}, m, m_0, \mathbf{L}, \mathbf{R})$, $\mathbb{w} = \left(\begin{smallmatrix}\boldsymbol{z}_\mathsf{a}\\ \boldsymbol{z}_\mathsf{b}\end{smallmatrix}\right) \in \mathbb{F}^{2(m-m_0-1)}$.

5 Security Proofs

We provide some additional preliminaries, needed to prove $\mathfrak{Vampire}$'s security.

Fact 3 (Schwartz-Zippel Lemma) *Let $f(X_1,\ldots,X_c)\neq 0$ be a total degree-d polynomial over a field $\mathbb{F}$ and let $\mathcal{S}\subseteq\mathbb{F}^c$. Then, $\Pr[\boldsymbol{x}\leftarrow\$\,\mathcal{S}: f(\boldsymbol{x})=0]\leq d/|\mathcal{S}|$.*

Fact 4 (Bauer *et al.* [BFL20]) *Let $\mathcal{V}(X_1,\ldots,X_c)\in\mathbb{F}[X_1,\ldots,X_c]$ be a non-zero polynomial of total degree d. Define $\mathcal{P}(Z)\in(\mathbb{F}[S_1,\ldots,S_c,R_1,\ldots,R_c])[Z]$ as $\mathcal{P}(Z):=\mathcal{V}(S_1Z+R_1,\ldots,S_cZ+R_c)$. Then the coefficient of the leading term in $\mathcal{P}(Z)$ is a polynomial in $\mathbb{F}[S_1,\ldots,S_c]$ of degree d.*

5.1 Knowledge-Soundness Proof

Theorem 1. $\mathfrak{Vampire}$ *is knowledge-sound in the AGM under the PDL assumption. More precisely, an algebraic $\mathcal{A}$ breaks the knowledge-soundnness of* $\mathfrak{Vampire}$ *with probability at most*

$$\mathsf{Adv}^{\mathrm{pdl}}_{d_1,d_2,\mathsf{Pgen},\mathcal{B}}(\lambda)\cdot\tfrac{|\mathbb{F}|^2}{|\mathbb{F}|^2-q}+\tfrac{16n_h+4m_0-12}{|C_\beta|}+\tfrac{n_h-1}{|\mathbb{F}|-n_h}\ , \tag{15}$$

where $\mathcal{B}$ is a PPT adversary, $d_1=\max(d_{\mathsf{gap}}+d,n_k-1)$, $d_2=\max(n_k,d_{\mathsf{gap}}+2)$, and $q=\max(q_1,q_2)$, where $q_1=n_k+d_{\max}$, $q_2=d_{\max}+2+d_{\mathsf{gap}}$, and $d_{\max}=\max(d_{\mathsf{gap}}+d,n_k-1)$.

Proof. Let $\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)$ be an algebraic adversary in the knowledge soundness game and $\mathsf{Ext}_{\mathcal{A}}$ its AGM extractor. In each round of the protocol, $\mathcal{A}$ sends either an integer or a $\mathbb{G}_1$ element. For $\mathbb{G}_1$ element, $\mathsf{Ext}_{\mathcal{A}}$ outputs coefficients of a polynomial where its monomials belong to $\mathcal{S}_1$. Let us denote polynomials that the adversary sends as $\tilde{z}(Y,X_\tau)$, $\psi_{\mathsf{ipc}}(Y,X_\tau)$, $R(Y,X_\tau)$, $Q(Y,X_\tau)$, and $B_{\mathsf{pc}}(Y,X_\tau)$, where each of the polynomials is in the span of $\mathcal{S}_1$. Integers $v_z,v_M\in\mathbb{F}$ that the prover sends are denoted as in the honest protocol description.

The knowledge-soundness extractor Ext is described in Fig. 3. It runs $\mathsf{Ext}_{\mathcal{A}}$ to obtain coefficients of $\tilde{z}(Y,X_\tau)$ and then evaluates $\tilde{z}(Y,0)$ at points of $Y\in\mathbb{H}$ which correspond to $\boldsymbol{z}_{\mathsf{a}}$ and $\boldsymbol{z}_{\mathsf{b}}$ in the honest argument and returns those vectors. The rest of the proof shows that the value outputted by Ext is a valid witness for $\mathbb{x}$ with an overwhelming probability.

We have two checks, with each check guaranteeing that $\mathcal{V}_i(\sigma,\tau)=0$, where

$$\begin{aligned}\mathcal{V}_1(Y,X_\tau):=&(\mathsf{num}(Y)-\tfrac{v_M}{n_k}\mathsf{den}(Y))X_\tau-R(Y,X_\tau)\mathsf{zden}(Y)-Q(Y,X_\tau)\mathcal{Z}_{\mathbb{K}}(Y)\ ,\\ \mathcal{V}_2(Y,X_\tau):=&(\tilde{z}(Y,X_\tau)-v_z)(Y-\beta)S(Y)+\Psi(Y,X_\tau)(Y-\beta\omega^m)S(Y)-\\ &\psi_{\mathsf{ipc}}(Y,X_\tau)(Y-\beta\omega^m)-B_{\mathsf{pc}}(Y,X_\tau)(Y-\beta)(Y-\beta\omega^m)S(Y)\ ,\end{aligned}$$

where $\Psi(Y,X_\tau)=\left(\Lambda^b_{\mathbb{H}}(\alpha,\beta)-v_M\cdot z(\beta\omega^m)\right)(\mathcal{Z}_{\mathsf{inp}}(\beta)\tilde{z}(Y,X_\tau)+\mathsf{inp}(\beta))$ and $z(\beta\omega^m)=\mathcal{Z}_{\mathsf{inp}}(\beta\omega^m)v_z+\mathsf{inp}(\beta\omega^m)$.

We can express $\mathcal{A}$'s winning probability as $\Pr[\mathcal{A}\text{ wins}]\leq\Pr[\mathcal{A}\text{ wins}\mid\mathcal{V}_1(Y,X_\tau)=0\wedge\mathcal{V}_2(Y,X_\tau)=0]+\Pr[\mathcal{A}\text{ wins}\mid\mathcal{V}_1(Y,X_\tau)\neq 0\vee\mathcal{V}_2(Y,X_\tau)\neq 0]$. We will analyze both probabilities in separate lemmas.

$\mathsf{Ext}(\mathsf{srs}, \mathsf{aux}; r)$

$(\tilde{z}(Y, X_\tau), \dots) \leftarrow \mathsf{Ext}_{\mathcal{A}}(\mathsf{srs}, \mathsf{aux}; r)$;
$\boldsymbol{z}_{\mathsf{a}} \leftarrow \left(\tilde{z}(\omega^{m_0+1}, 0) \cdot \mathcal{Z}_{\mathsf{inp}}(\omega^{m_0+1}), \dots, \tilde{z}(\omega^{m-1}, 0) \cdot \mathcal{Z}_{\mathsf{inp}}(\omega^{m-1})\right)^\top$;
$\boldsymbol{z}_{\mathsf{b}} \leftarrow \left(\tilde{z}(\omega^{m+m_0+1}, 0) \cdot \mathcal{Z}_{\mathsf{inp}}(\omega^{m+m_0+1}), \dots, \tilde{z}(\omega^{2m-1}, 0) \cdot \mathcal{Z}_{\mathsf{inp}}(\omega^{2m-1})\right)^\top$;
return $\mathbb{w} = \left(\begin{smallmatrix} \boldsymbol{z}_{\mathsf{a}} \\ \boldsymbol{z}_{\mathsf{b}} \end{smallmatrix}\right)$;

Fig. 3. The knowledge-soundness extractor Ext for $\mathfrak{Vampire}$ zk-SNARK where $\mathcal{A}$ is an algebraic adversary and $\mathsf{Ext}_{\mathcal{A}}$ its extractor.

Lemma 5. *For an algebraic $\mathcal{A}$ and $\mathcal{V}_1(Y, X_\tau), \mathcal{V}_2(Y, X_\tau)$ as defined above,* $\Pr[\mathcal{A} \text{ wins} \mid \mathcal{V}_1(Y, X_\tau) = 0 \wedge \mathcal{V}_2(Y, X_\tau) = 0] \leq (16n_h + 4m_0 - 12)/|C_\beta| + (n_h - 1)/(|\mathbb{F}| - n_h)$.

Proof. We analyze both equations in the following.

Eq. $\mathcal{V}_1(Y, X_\tau) = 0$. We express $R(Y, X_\tau) = X_\tau R'(Y) + R''(Y)$ and $Q(Y, X_\tau) = X_\tau Q'(Y) + Q''(Y)$. In particular, $R'(Y)$ and $Q'(Y)$ have at most degree $n_k - 2$ since the only X_τ-dependent monomials $\mathcal{S}_1$ are $(Y^i X_\tau)_{i=0}^{n_k-2}$. Thus, $\mathcal{V}_1(Y, X_\tau) = \mathcal{V}_1'(Y)X_\tau - R''(Y)\mathsf{zden}(Y) - Q''(Y)\mathcal{Z}_{\mathbb{K}}(Y)$, where $\mathcal{V}_1'(Y) := \mathsf{num}(Y) - \frac{v_M}{n_k}\mathsf{den}(Y) - R'(Y)\mathsf{zden}(Y) - Q'(Y)\mathcal{Z}_{\mathbb{K}}(Y)$.

From $\mathcal{V}_1(Y, X_\tau) = 0$, we get that $\mathcal{V}_1'(Y) = 0$. Thus, $\forall y \in \mathbb{K}.(\mathsf{num}(y) - v_M/n_k \cdot \mathsf{den}(y)) - yR'(y)\mathsf{den}(y) = 0$, that is, $\forall y \in \mathbb{K}.T(y) = \mathsf{num}(y)/\mathsf{den}(y) = yR'(y) + v_M/n_k$. Note that $\mathsf{den}(y) \neq 0$ since $\alpha, \beta \notin \mathbb{H}$. Since $\widehat{T}(Z) := \sum_{y \in \mathbb{K}} T(y) L_y^{\mathbb{K}}(Z)$ has degree $\leq n_k - 1$, we get that $\widehat{T}(Z) = ZR'(Z) + v_M/n_k$. By Aurora's sumcheck,

$$M^b(\alpha, \beta) = \textstyle\sum_{y \in \mathbb{K}} T(y) = \sum_{y \in \mathbb{K}} \widehat{T}(y) = v_M \ . \tag{16}$$

Eq. $\mathcal{V}_2(Y, X_\tau) = 0$. Given that $\beta \in C_\beta$, we know that $Y - \beta$ and $Y - \beta\omega^m$ do not divide $S(Y)$ and also that $\beta \neq \beta\omega^m$ since $\beta \neq 0$. Thus, $Y - \beta$, $Y - \beta\omega^m$, and $S(Y)$ are pair-wise coprime. Thus, $\mathcal{V}_2(Y, X_\tau) = 0$ implies that

$$(Y - \beta\omega^m) \mid (\tilde{z}(Y, X_\tau) - v_z) \ , \tag{17}$$

$$(Y - \beta) \mid (\Psi(Y, X_\tau)S(Y) - \psi_{\mathsf{ipc}}(Y, X_\tau)) \ . \tag{18}$$

Eq. (17) gives that $\tilde{z}(\beta\omega^m, X_\tau) = v_z$. Denoting $\tilde{z}(Y, X_\tau) = \tilde{z}'(Y)X_\tau + \tilde{z}''(Y)$, $\tilde{z}'(\beta\omega^m)X_\tau + \tilde{z}''(\beta\omega^m) = v_z$. Thus, $v_z = \tilde{z}''(\beta\omega^m)$ and $\tilde{z}'(\beta\omega^m) = 0$.

Let us denote $\psi_{\mathsf{ipc}}(Y, X_\tau) = \psi'_{\mathsf{ipc}}(Y)X_\tau + \psi''_{\mathsf{ipc}}(Y)$. Observe that $\psi''_{\mathsf{ipc}}(Y) \in \mathsf{PolyPunc}_{\mathbb{F}}(d, d_{\mathsf{gap}}, Y)$. We can express $\Psi(Y, X_\tau)$ as

$$\begin{aligned}
\Psi(Y, X_\tau) &= \left(\Lambda^b_{\mathbb{H}}(\alpha, \beta) - v_M \cdot z(\beta\omega^m)\right)\left(\mathcal{Z}_{\mathsf{inp}}(\beta)\tilde{z}(Y, X_\tau) + \mathsf{inp}(\beta)\right) \\
&= \left(\Lambda^b_{\mathbb{H}}(\alpha, \beta) - v_M \cdot z(\beta\omega^m)\right)\left(\mathcal{Z}_{\mathsf{inp}}(\beta)(\tilde{z}'(Y)X_\tau + \tilde{z}''(Y)) + \mathsf{inp}(\beta)\right) \\
&= \Psi'(Y)X_\tau + \Psi''(Y) \ ,
\end{aligned}$$

where $\Psi'(Y) := (\Lambda^b_{\mathbb{H}}(\alpha, \beta) - v_M \cdot z(\beta\omega^m))\mathcal{Z}_{\mathsf{inp}}(\beta)\tilde{z}'(Y)$ and $\Psi''(Y) := (\Lambda^b_{\mathbb{H}}(\alpha, \beta) - v_M \cdot z(\beta\omega^m))\left(\mathcal{Z}_{\mathsf{inp}}(\beta)\tilde{z}''(Y) + \mathsf{inp}(\beta)\right)$.

Thus, Eq. (18) implies $\Psi(\beta, X_\tau)S(\beta) - \psi_{\mathsf{ipc}}(\beta, X_\tau) = (\Psi'(\beta)S(\beta) - \psi'_{\mathsf{ipc}}(\beta))X_\tau + \Psi''(\beta)S(\beta) - \psi''_{\mathsf{ipc}}(\beta) = 0$. Hence, $\Psi''(\beta)S(\beta) = \psi''_{\mathsf{ipc}}(\beta)$. Let

$$\psi_\alpha(Y) := \left(\Lambda^b_{\mathbb{H}}(\alpha, Y) - M^b(\alpha, Y) \cdot z(Y\omega^m)\right)\left(\mathcal{Z}_{\mathsf{inp}}(Y)\tilde{z}''(Y) + \mathsf{inp}(Y)\right) \ .$$

Let $\mathcal{V}_3(Y) := \psi_\alpha(Y)S(Y) - \psi''_{\mathsf{ipc}}(Y)$. By Eq. (16), it holds that $\mathcal{V}_3(\beta) = 0$.

Since the polynomials ψ_α and ψ_{ipc} were fixed before the adversary received β, we can apply the Schwartz-Zippel lemma to $\mathcal{V}_3$. Note that (1) $\deg \tilde{z}'' \le d_{\mathsf{gap}} + d$, (2) $\deg \mathsf{inp} \le n_h - 1$, (3) $\deg \mathcal{Z}_{\mathsf{inp}} \le 2(m_0 + 1)$, (4) $\deg z \le d_{\mathsf{gap}} + d + 2(m_0 + 1)$, (5) $\deg_Y \Lambda_{\mathbb{H}}(\alpha, Y) \le n_h - 1$, $\deg_Y M^b(\alpha, Y) \le n_h - 1$, (6) $\deg \psi''_{\mathsf{ipc}} \le d_{\mathsf{gap}} + d$, (7) $\deg \psi_\alpha \le (n_h - 1) + 2\left(d_{\mathsf{gap}} + d + 2(m_0 + 1)\right) = 13n_h + 4m_0 - 9$.

Thus, $\deg \mathcal{V}_3 \le \max(\deg \psi_\alpha + d_{\mathsf{gap}}, \deg \psi''_{\mathsf{ipc}}) \le \max(16n_h + 4m_0 - 12, 6(n_h - 1)) = 16n_h + 4m_0 - 12$. It follows from the Schwartz-Zippel lemma that $\mathcal{V}_3(Y) \neq 0$ and $\mathcal{V}_3(\beta) = 0$ can happen at most with probability $(16n_h + 4m_0 - 12)/|C_\beta|$. Thus, let us assume from this point on that $\mathcal{V}_3(Y) = 0$.

Since $\psi_\alpha(Y)S(Y) = \psi''_{\mathsf{ipc}}(Y)$, $\deg \psi''_{\mathsf{ipc}} \le d_{\mathsf{gap}} + d$ and $\deg S = d_{\mathsf{gap}}$, then $\deg \psi_\alpha \le d$. Since $X^{d_{\mathsf{gap}}} \notin \mathcal{S}_1$, the coefficient of $Y^{d_{\mathsf{gap}}}$ in $\psi''_{\mathsf{ipc}}(Y) = \psi_\alpha(Y)S(Y) = \psi_\alpha(Y)(Y^{d_{\mathsf{gap}}} + Y^{d_{\mathsf{gap}} - n_h} + Y^{d_{\mathsf{gap}} - 2n_h})$ is 0. But this coefficient is $(\psi_\alpha)_0 + (\psi_\alpha)_{n_h} + (\psi_\alpha)_{2n_h}$. Thus, from Lemma 1 it follows that $\sum_{y \in \mathbb{H}} \psi_\alpha(y) = 0$.

Let us express $\psi_\alpha(Y)$ as $\psi(X, Y)$, where X corresponds to α. We established that $\sum_{y \in \mathbb{H}} \psi(\alpha, y) = 0$. For any $y \in \mathbb{H}$, $\deg_X \psi(X, y) = n_h - 1$. If $\sum_{y \in \mathbb{H}} \psi(X, y) \neq 0$, then by the Schwartz-Zippel lemma, $\sum_{y \in \mathbb{H}} \psi(\alpha, y) = 0$ with probability at most $(n_h - 1)/(|\mathbb{F}| - n_h)$. Assume that $\sum_{y \in \mathbb{H}} \psi(X, y) = 0$. By Lemma 4, $\forall x \in \mathbb{H}.P(x) = 0$, where $P(X)$ is as in Eq. (5). In the beginning of Section 4, we established that this equation is equivalent to R1CSLite. Since $z(Y) = \mathcal{Z}_{\mathsf{inp}}(Y)\tilde{z}''(Y) + \mathsf{inp}(Y) = \tilde{z}''(Y)\prod_{i=1}^{m_0+1}(Y - \omega^{i-1})(Y - \omega^{m+i-1}) + \ell_1^{\mathbb{H}}(Y) + \sum_{i=1}^{m_0} \mathtt{x}_i \ell_{i+1}^{\mathbb{H}}(Y) + \sum_{i=1}^{m_0+1} \ell_{m+i}^{\mathbb{H}}(Y)$, then $z(\omega^{i-1})$ for $i \in \{1, \ldots, m_0 + 1\}$ correctly encodes $(1, \mathtt{x}_1, \ldots, \mathtt{x}_{m_0})$. The extractor extracts $z(\omega^{i-1})$ for $i \in \{m_0 + 2, \ldots, m\} \cup \{m + m_0 + 2, \ldots, 2m\}$ which indeed corresponds to the R1CSLite witness. □

Lemma 6. *Let $d_1 = \max(d_{\mathsf{gap}} + d, n_k - 1)$, $d_2 = \max(n_k, d_{\mathsf{gap}} + 2)$. For an algebraic $\mathcal{A}$ and $\mathcal{V}_1(Y, X_\tau), \mathcal{V}_2(Y, X_\tau)$ as above, there exists a PPT $\mathcal{B}$, s.t.* $\Pr[\mathcal{A}$ *wins* $\mid \mathcal{V}_1(Y, X_\tau) \neq 0 \vee \mathcal{V}_2(Y, X_\tau) \neq 0] \le \mathsf{Adv}^{\mathrm{pdl}}_{d_1, d_2, \mathsf{Pgen}, \mathcal{B}}(\lambda) \cdot |\mathbb{F}|^2/(|\mathbb{F}|^2 - q)$.

Proof. This part of the proof is standard and essentially equivalent to [FKL18] AGM proof for Groth16 SNARK. Hence, we only sketch the main idea. We construct an adversary $\mathcal{B}$ that breaks the (d_1, d_2)-PDL assumption whenever $\mathcal{A}$ wins in the knowledge-soundness game and either $\mathcal{V}_1(Y, X_\tau) \neq 0$ or $\mathcal{V}_2(Y, X_\tau) \neq 0$.

The adversary $\mathcal{B}$ gets as an input $(\mathsf{p}; [(x^i)_{i=0}^{d_1}]_1, [(x^i)_{i=0}^{d_2}]_2)$. It samples s_1, s_2, r_1, r_2 and defines $\sigma = s_1 x + r_1$ and $\tau = s_2 x + r_2$. Although $\mathcal{B}$ does not know σ or τ (they depend on the challenge x), $\mathcal{B}$ is able to homomorphically compute elements of the form $\left[\tau\sigma^i\right]_\iota$ (e.g., $[\sigma]_1 = s_1[x]_1 + r_1[1]_1$). The degrees d_1 and d_2 are sufficiently large so that $\mathcal{B}$ can compute srs where σ and τ are the trapdoors. Next, $\mathcal{B}$ runs $\mathcal{A}$ and $\mathsf{Ext}_{\mathcal{A}}$ on this srs to obtain the argument and related argument polynomials. Moreover, $\mathcal{B}$ now knows coefficients of the verification polynomials $\mathcal{V}_1(Y, X_\tau)$ and $\mathcal{V}_2(Y, X_\tau)$.

When $\mathcal{A}$ wins, $\mathcal{V}_1(\sigma,\tau)=0$ and $\mathcal{V}_2(\sigma,\tau)=0$. We define $\mathcal{P}_i(X) := \mathcal{V}_i(S_1X+R_1, S_2X+R_2) \in (\mathbb{F}[S_1,S_2,R_1,R_2])[X]$ for $i\in\{1,2\}$. According to Fact 4, if $\mathcal{V}_i(Y,X_\tau)\neq 0$ has degree q_i, then the coefficient of the maximal degree of $\mathcal{P}_i(X)$ is some polynomial $C(S_1,S_2)\in\mathbb{F}[S_1,S_2]$ of degree q_i. Thus, the coefficient of the leading term of $\mathcal{P}'_i(X) := \mathcal{V}_i(s_1X+r_1, s_2X+r_2)\in\mathbb{Z}_p[X]$ is $C(s_1,s_2)$. Since s_1 and s_2 are information-theoretically hidden from $\mathcal{A}$ (they are masked by r_1 and r_2), by the Schwartz-Zippel lemma, $C(s_1,s_2)=0$ at most with probability $q_i/|\mathbb{F}|^2$. Thus, with an overwhelming probability, $C(s_1,s_2)\neq 0$ and $\mathcal{P}'_i(X)\neq 0$.

Thus, $\mathcal{B}$ can check which out of $\mathcal{V}_i(Y,X_\tau)$, $i\in\{1,2\}$, is non-zero and then find the roots of corresponding $\mathcal{P}'_i(X)$. One of the roots must be σ since $\mathcal{P}'_i(\sigma) = \mathcal{V}_i(s_1\sigma+r_1, s_2\sigma+r_2)=0$. Finally, $\mathcal{B}$ outputs σ.

Let $q := \max(q_1,q_2)$. Clearly, the maximum total degree q_1 of $\mathcal{V}_1$ is at most $n_k+d_{\max}$, where $d_{\max}=\max(d_{\mathsf{gap}}+d, n_k-1)$. The maximum total degree q_2 of $\mathcal{V}_2$ is at most $d_{\max}+2+d_{\mathsf{gap}}$. Finally, $\Pr[\mathcal{A}\text{ wins} \mid \mathcal{V}_1(Y,X_\tau)\neq 0 \vee \mathcal{V}_2(Y,X_\tau)\neq 0]\cdot(1-q/|\mathbb{F}|^2) \le \mathsf{Adv}^{\mathrm{pdl}}_{d_1,d_2,\mathsf{Pgen},\mathcal{B}}(\lambda)$. Thus, $\Pr[\mathcal{A}\text{ wins} \mid \mathcal{V}_1(Y,X_\tau)\neq 0 \vee \mathcal{V}_2(Y,X_\tau)\neq 0] \le \mathsf{Adv}^{\mathrm{pdl}}_{d_1,d_2,\mathsf{Pgen},\mathcal{B}}(\lambda)\cdot|\mathbb{F}|^2/(|\mathbb{F}|^2-q)$. □

It follows from these lemmas that Eq. (15) holds. This proves the claim. □

5.2 Zero-Knowledge Proof

Theorem 2. *Let $b=4$. Then, $\mathfrak{Vampire}$ is perfectly zero-knowledge.*

Proof. We construct a simulator that, given an input $(\mathsf{srs},\mathsf{td},\mathcal{R},\mathbb{x})$, simulates the argument using a witness with $\boldsymbol{z}_\mathsf{a}=\boldsymbol{z}_\mathsf{b}=\mathbf{0}$. We argue that no adversary can distinguish an argument with a zero witness from argument with a real witness.

Let $\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)$ be an unbounded adversary. Suppose that $\mathcal{A}_1(\mathsf{srs})$ outputs $(\mathcal{R},\mathbb{x},\mathbb{w},st)$ such that $(\mathbb{x},\mathbb{w})\in\mathcal{R}$ and $\mathcal{R}\in\mathcal{UR}_{\mathsf{p},n_h}$. We describe the simulator as it interacts with $\mathcal{A}_2(st)$ who plays the role of a malicious verifier.

The simulator Sim proceeds as follows. In the first round, Sim sets $\boldsymbol{z}_\mathsf{a},\boldsymbol{z}_\mathsf{b}\leftarrow\mathbf{0}_{m-m_0-1}$ and outputs the commitment $[\tilde{z}(\sigma)]_1$ computed from it as in the real protocol. Then, it obtains α from $\mathcal{A}_2(st)$ and aborts if $\alpha\notin\mathbb{F}\setminus\mathbb{H}$.

In the second round, Sim computes the polynomial $\psi_{\mathsf{ipc}}(Y)$ as in the honest protocol. In the real argument, $\psi_{\mathsf{ipc}}(Y)$ has a zero coefficient of $Y^{d_{\mathsf{gap}}}$; with an overwhelming probability, this is not a case in the simulated argument. Hence, Sim uses the trapdoor σ to compute the commitment $[\psi_{\mathsf{ipc}}(\sigma)]_1$ as in Eq. (11). Next, Sim obtains β from $\mathcal{A}_2(st)$ and aborts if $\beta\notin\mathbb{F}\setminus(\mathbb{H}\cup\{\sigma,\sigma/\omega^m\})$.

After that, Sim follows the protocol. In particular, Sim computes $[B_{\mathsf{pc}}(\sigma)]_1$ honestly as in Fig. 2, by setting $B_{\mathsf{pc}}(\sigma) \leftarrow (\tilde{z}(\sigma)-v_z)/(\sigma-\beta\omega^m) + \Psi(\sigma)/(\sigma-\beta) - \psi_\alpha(\sigma)/(\sigma-\beta)$; this is possible since $\beta\notin\{\sigma,\sigma/\beta^m\}$. Hence, if there are no aborts, the argument transcript is $[\tilde{z}(\sigma)]_1,\alpha,[\psi_{\mathsf{ipc}}(\sigma)]_1,\beta, v_z, v_M, [R(\sigma)\tau, Q(\sigma)\tau]_1, [B_{\mathsf{pc}}(\sigma)]_1$.

We show that $\mathcal{A}$ cannot tell apart the real and simulated arguments, by showing that each argument element has identical distribution in the honest and simulated case. First, only the polynomials $\tilde{z}$ and z depend on the witness; moreover, z is determined by $\tilde{z}$ and the public input $\mathbb{x}$. Furthermore, $\tilde{z}$ is evaluated

at the points β, $\beta\omega^m$, σ, and $\sigma\omega^m$ (the last evaluation is done inside $\psi_{\mathsf{ipc}}(\sigma)$). Thus, given $b = 4$, $\tilde{z}(\sigma)$ (and $z(\sigma)$) have the same distribution.

Recall from Eq. (11) that $\psi_{\mathsf{ipc}}(Y) = (\Lambda^b_{\mathbb{H}}(\alpha, Y) - M^b(\alpha, Y)z(Y\omega^m))$ $z(Y)S(Y)$. All polynomials except z are public. As observed, $z(Y\omega^m)$ is uniquely determined by $\mathbb{x}$ and $\tilde{z}(Y\omega^m)$. Since $\tilde{z}(\sigma)$ has identical distributions in the honest and simulated arguments and $\psi_{\mathsf{ipc}}(\sigma)$ is deterministically computed from it, $[\psi_{\mathsf{ipc}}(\sigma)]_1$ has the same distribution in the honest and in the simulated argument.

The values $v_M = M^b(\alpha, \beta)$ and $[R(\sigma)\tau, Q(\sigma)\tau]_1$ are independent of the witness and computed honestly by the simulator. Finally, $[B_{\mathsf{pc}}(\sigma)]_1$ is uniquely determined by the verification equation and can be computed from $\tilde{z}(\sigma), \tau, \sigma, \beta$ and $\mathbb{x}$. That is, from elements with identical distributions in the honest and simulated arguments. This proves the claim. □

Subversion Zero Knowledge. Next, we show that $\mathfrak{Vampire}$ is subversion zero-knowledge (Sub-ZK, [BFS16,ABLZ17,Fuc18,ALSZ21]), i.e., $\mathfrak{Vampire}$ stays zero-knowledge even when the SRS generator is compromized. For this, we need to modify the Sub-ZK definition of [ALSZ21] to match interactive argument systems for universal relations. The new definition divides $\mathcal{A}$ into $\mathcal{A}_1$ and $\mathcal{A}_2$; moreover, we allow it to pick the relation which will be proven. Since Derive is deterministic and uses only public input, we assume that the SRS specialization is performed honestly. Hence, any party, when given $\mathsf{ek}_{\mathcal{R}}$ or $\mathsf{vk}_{\mathcal{R}}$ by an untrusted party, can verify their correctness by running $\mathsf{Derive}(\mathsf{srs}, \mathcal{R})$. More precisely, we explicitly assume that the prover, who is supposed to verify correctness of $\mathsf{ek}_{\mathcal{R}}$, has access to the whole SRS.

Sub-ZK. Π is (statistical) *subversion zero-knowledge*, if there exist a PPT SRS verification algorithm SrsVer and a PPT simulator Sim, such that the following holds: for any PPT subverter $\mathcal{Z}$, there exists a PPT $\mathsf{Ext}_{\mathcal{Z}}$, such that for all unbounded $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and $N \in \mathsf{poly}(\lambda)$,

$$\Pr\left[\begin{array}{c|c} \langle \mathsf{P}(\mathsf{ek}_{\mathcal{R}}, \mathbb{x}, \mathbb{w}), \mathcal{A}_2(st)\rangle = 1 & \mathsf{srs} \leftarrow \mathcal{Z}(\mathsf{p}, N); \\ \wedge \mathsf{SrsVer}(\mathsf{srs}) = 1 \wedge \mathcal{R}(\mathbb{x}, \mathbb{w}) & (\mathcal{R}, \mathbb{x}, \mathbb{w}, st) \leftarrow \mathcal{A}_1(\mathsf{srs}); \\ \wedge\, \mathcal{R} \in \mathcal{UR}_{\mathsf{p},N} & (\mathsf{ek}_{\mathcal{R}}, \mathsf{vk}_{\mathcal{R}}) \leftarrow \mathsf{Derive}(\mathsf{srs}, \mathcal{R}) \end{array}\right] \approx_s$$

$$\Pr\left[\begin{array}{c|c} \langle \mathsf{Sim}(\mathsf{srs}, \mathsf{td}, \mathcal{R}, \mathbb{x}), \mathcal{A}_2(st)\rangle = 1 & \mathsf{srs} \leftarrow \mathcal{Z}(\mathsf{p}, N); \mathsf{td} \leftarrow \mathsf{Ext}_{\mathcal{A}}(\mathsf{p}, N); \\ \wedge \mathsf{SrsVer}(\mathsf{srs}) = 1 \wedge \mathcal{R}(\mathbb{x}, \mathbb{w}) & (\mathcal{R}, \mathbb{x}, \mathbb{w}, st) \leftarrow \mathcal{A}_1(\mathsf{srs}); \\ \wedge\, \mathcal{R} \in \mathcal{UR}_{\mathsf{p},N} & (\mathsf{ek}_{\mathcal{R}}, \mathsf{vk}_{\mathcal{R}}) \leftarrow \mathsf{Derive}(\mathsf{srs}, \mathcal{R}) \end{array}\right].$$

We highlighted the changes compared to the definition of zero-knowledge.

We prove that $\mathfrak{Vampire}$ is subversion zero-knowledge under the BDH-KE assumption [ABLZ17,ALSZ21]. BDH-KE states that if an adversary, given p, outputs $([\sigma]_1, [\sigma]_2)$, then one can extract σ. Here, we also construct an algorithm SrsVer that verifies the correctness of srs; SrsVer is also needed for $\mathfrak{Vampire}$ to be updatable.

Theorem 3. $\mathfrak{Vampire}$ *is subversion zero-knowledge under the BDH-KE.*

Proof. As proven in [ALSZ21], a perfectly zero-knowledge argument system is subversion zero-knowledge if (1) there exists a PPT algorithm $\mathsf{SrsVer}(\mathsf{srs})$ that

outputs 1 or 0; in the first case, for a valid $\mathbb{x}$, $\mathsf{Sim}(\mathsf{srs}, \mathsf{td}, \mathcal{R}, \mathbb{x})$ outputs an argument indistinguishable from the real one, and (2) for any PPT adversary $\mathcal{Z}$, there exists a PPT extractor $\mathsf{Ext}_{\mathcal{Z}}$, such that: if $\mathsf{srs} \leftarrow \mathcal{Z}(\mathsf{p}, N; r)$ and $\mathsf{SrsVer}(\mathsf{srs}) = 1$, then $\mathsf{Ext}_{\mathcal{Z}}(\mathsf{p}, N; r)$ outputs the simulation trapdoor σ with overwhelming probability,.

The existence of Sim follows since if SrsVer accepts then the SRS is computed correctly and the extractor provides Sim with the corresponding trapdoor, i.e., Sim behaves as in Theorem 2. Next, we construct SrsVer and $\mathsf{Ext}_{\mathcal{Z}}$.

$\underline{\mathsf{SrsVer}(\mathsf{srs})}$: Recall from Fig. 2 that $\{(X^i)_{i=0:i\neq d_{\mathsf{gap}}}^{d_{\mathsf{gap}}+d}, (X^i X_\tau)_{i=0}^{n_k-2}\}$ and $\mathcal{S}_2 = \{(X^i)_{i=0}^{n_k}, ((X^{d_{\mathsf{gap}}+j-n_h i})_{j=0}^{2})_{i=0}^{\lfloor d/n_h \rfloor}\}$. We use $\underline{[x]_\iota}$ to denote the claimed value (e.g., $\underline{[\sigma^2]_1}$) of an entry in the SRS (e.g., $[\sigma^2]_1$) and $[x]_\iota$ to denote the same value after it has already been verified. We assume in the start that $[1, \sigma, \tau]_1$ and $[1]_2$ are verified. After a check of $\underline{[x]_\iota}$ in an equation where all other variables are already verified, we can think of $[x]_\iota$ to be verified too. For example, the check $[\sigma]_1 \bullet [1]_2 = [1]_1 \bullet \underline{[\sigma]_2}$ convinces us that $\underline{[\sigma]_2} = [\sigma]_2$ is correctly computed.

1. Check $[\sigma]_1 \bullet [1]_2 = [1]_1 \bullet \underline{[\sigma]_2}$.
2. For $i \in [1, d_{\mathsf{gap}} - 1]$: check $[\sigma^{i-1}]_1 \bullet [\sigma]_2 = \underline{[\sigma^i]_1} \bullet [1]_2$.
3. Check $[\sigma^2]_1 \bullet [1]_2 = [1]_1 \bullet \underline{[\sigma^2]_2}$.
4. Check $[\sigma^{d_{\mathsf{gap}}-1}]_1 \bullet [\sigma^2]_2 = \underline{[\sigma^{d_{\mathsf{gap}}+1}]_1} \bullet [1]_2$.
5. For $i \in [d_{\mathsf{gap}} + 2, d_{\mathsf{gap}} + d]$: check $[\sigma^{i-1}]_1 \bullet [\sigma]_2 = \underline{[\sigma^i]_1} \bullet [1]_2$.
6. For $i \in [2, n_k]$: check $[\sigma]_1 \bullet [\sigma^{i-1}]_2 = [1]_1 \bullet \underline{[\sigma^i]_2}$.
7. (if $d_{\mathsf{gap}} > n_k$) Check $[\sigma^{d_{\mathsf{gap}}-1}]_1 \bullet [\sigma]_2 = [1]_1 \bullet \underline{[\sigma^{d_{\mathsf{gap}}}]_2}$.
8. For $i \in [1, n_k - 2]$: check $[\sigma^{i-1}\tau]_1 \bullet [\sigma]_2 = \underline{[\sigma^i \tau]_1} \bullet [1]_2$.
9. For $k \in \{d_{\mathsf{gap}} - n_h, d_{\mathsf{gap}} - 2n_h, d_{\mathsf{gap}} + 1, d_{\mathsf{gap}} + 1 - n_h, d_{\mathsf{gap}} + 1 - 2n_h, d_{\mathsf{gap}} + 2, d_{\mathsf{gap}} + 2 - n_h, d_{\mathsf{gap}} + 2 - 2n_h\}$ (if $k > n_k$): check $[\sigma^k]_1 \bullet [1]_2 = [1]_1 \bullet \underline{[\sigma^k]_2}$.

If all of the above checks pass, then SrsVer outputs 1, otherwise it outputs 0. Clearly, SrsVer is correctly constructed.

$\underline{\mathsf{Ext}_{\mathcal{Z}}(\mathsf{p}, n_h)}$: Let $\mathcal{Z}$ be an adversary, that on input (p, N) outputs srs. Since $\underline{\mathsf{SrsVer}(\mathsf{srs})} = 1$, the SRS has the form specified at Fig. 2. In particular, it contains $([\sigma]_1, [\sigma]_2) = \sigma([1]_1, [1]_2)$. By the BDH-KE assumption, there exists an extractor $\mathsf{Ext}'_{\mathcal{A}}$ that extracts σ from $\mathcal{A}$. The Sub-ZK extractor $\mathsf{Ext}_{\mathcal{A}}$ just returns σ.

Since the SRS has been computed correctly and there exists an extractor that extracts σ, $\mathsf{Sim}(\mathsf{srs}, \mathbb{x})$ outputs an argument indistinguishable from a real one. This proves the claim. □

Acknowledgment. Janno Siim was partially supported by the Estonian Research Council grant (PRG49).

References

ABLZ17. Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, and Michal Zajac. A subversion-resistant SNARK. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2017. doi:10.1007/978-3-319-70700-6_1.

ALSZ21. Behzad Abdolmaleki, Helger Lipmaa, Janno Siim, and Michal Zajac. On subversion-resistant SNARKs. *Journal of Cryptology*, 34(3):17, July 2021. doi:10.1007/s00145-021-09379-y.

BBHR18. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl, July 2018. doi:10.4230/LIPIcs.ICALP.2018.14.

BCC+16. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357. Springer, Heidelberg, May 2016. doi:10.1007/978-3-662-49896-5_12.

BCG+14. Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *IEEE SP 2014*, pages 459–474, Berkeley, CA, USA, May 18–21, 2014. IEEE Computer Society.

BCI+13. Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333. Springer, Heidelberg, March 2013. doi:10.1007/978-3-642-36594-2_18.

BCR+19. Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17653-2_4.

BCS21. Jonathan Bootle, Alessandro Chiesa, and Katerina Sotiraki. Sumcheck arguments and their applications. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 742–773, Virtual Event, August 2021. Springer, Heidelberg. doi:10.1007/978-3-030-84242-0_26.

BDFG20. Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. Efficient polynomial commitment schemes for multiple points and polynomials. Cryptology ePrint Archive, Report 2020/081, 2020. https://eprint.iacr.org/2020/081.

BFL20. Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. A classification of computational assumptions in the algebraic group model. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 121–151. Springer, Heidelberg, August 2020. doi:10.1007/978-3-030-56880-1_5.

BFS16. Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 777–804. Springer, Heidelberg, December 2016. doi:10.1007/978-3-662-53890-6_26.

BGW05. Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 258–275. Springer, Heidelberg, August 2005. doi:10.1007/11535218_16.

Bow17. Sean Bowe. BLS12-381: New zk-SNARK Elliptic Curve Construction. Blog post, https://blog.z.cash/new-snark-curve/, last accessed in July, 2018, March 11, 2017.

CFF+21. Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: a toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. In Tibouchi and Wang [TW21], pages 3–33.

CHM+20. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45721-1_26.

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319-96881-0_2.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987. doi:10.1007/3-540-47721-7_12.

Fuc18. Georg Fuchsbauer. Subversion-zero-knowledge SNARKs. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 315–347. Springer, Heidelberg, March 2018. doi:10.1007/978-3-319-76578-5_11.

GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013. doi:10.1007/978-3-642-38348-9_37.

GKM+18. Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, August 2018. doi:10.1007/978-3-319-96878-0_24.

Gro10. Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2010. doi:10.1007/978-3-642-17373-8_19.

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016. doi:10.1007/978-3-662-49896-5_11.

GWC19. Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. https://eprint.iacr.org/2019/953.

ILV11. Malika Izabachène, Benoît Libert, and Damien Vergnaud. Block-wise P-signatures and non-interactive anonymous credentials with efficient attributes. In Liqun Chen, editor, *13th IMA International Conference on Cryptography and Coding*, volume 7089 of *LNCS*, pages 431–450. Springer, Heidelberg, December 2011.

KMSV21. Markulf Kohlweiss, Mary Maller, Janno Siim, and Mikhail Volkhov. Snarky Ceremonies. In Tibouchi and Wang [TW21], pages 98–127.

KZG10. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Heidelberg, December 2010. doi:10.1007/978-3-642-17373-8_11.

LFKN90. Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. In *31st FOCS*, pages 2–10. IEEE Computer Society Press, October 1990. doi:10.1109/FSCS.1990.89518.

Lip12. Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Heidelberg, March 2012. doi:10.1007/978-3-642-28914-9_10.

LY10. Benoît Libert and Moti Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 499–517. Springer, Heidelberg, February 2010. doi:10.1007/978-3-642-11799-2_30.

MBKM19. Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019. doi:10.1145/3319535.3339817.

PHGR13. Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013. doi:10.1109/SP.2013.47.

RZ21a. Carla Ràfols and Arantxa Zapico. An algebraic framework for universal and updatable SNARKs. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 774–804, Virtual Event, August 2021. Springer, Heidelberg. doi:10.1007/978-3-030-84242-0_27.

RZ21b. Carla Ràfols and Arantxa Zapico. An Algebraic Framework for Universal and Updatable SNARKs. Technical Report 2021/590, IACR, May 5, 2021. Last checked modification from August 19, 2021. URL: https://ia.cr/2021/590.

Tha13. Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 71–89. Springer, Heidelberg, August 2013. doi:10.1007/978-3-642-40084-1_5.

Tha21. Justin Thaler. Proofs, Arguments, and Zero-Knowledge. January 23, 2021. URL: http://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.html.

TW21. Mehdi Tibouchi and Huaxiong Wang, editors. *ASIACRYPT 2021 (3)*, volume 13092 of *LNCS*, Singapore, December 5–9, 2021. Springer, Cham.

A Full Efficiency Comparison

Next, we establish the efficiency of Vampire (see Fig. 2 for an easy reference). In Table 2, we provide an extensive comparison with other recent updatable and

universal zk-SNARKs. Recall that $n_h = |\mathbb{H}|$ is equal to $2m + b = 2m + 4$ and $n_k = |\mathbb{K}|$ is equal to the total number of non-zero entries in $\mathbf{L}$ and $\mathbf{R}$.

Size of srs**.** Let us compute which SRS elements are needed by $\mathfrak{Vampire}$. For the prover's computation to succeed, srs needs to contain the following $\mathbb{G}_1$ elements (we list the group elements output by P, important underlying polynomials, and the needed SRS elements):

Group element	Polynomial	SRS elements
$[\tilde{z}(\sigma)]_1$	$\tilde{z}(Y) \in \mathbb{F}_{\leq n_h - 2m_0 - 3}[Y]$	$[(\sigma^i)_{i=0}^{n_h - 2m_0 - 3}]_1$
$[\psi_{\mathsf{ipc}}(\sigma)]_1$	$\psi_{\mathsf{ipc}}(Y) \in \mathsf{PolyPunc}_{\mathbb{F}}(d, d_{\mathsf{gap}}, Y)$	$[(\sigma^i)_{i=0:i\neq d_{\mathsf{gap}}}^{d_{\mathsf{gap}}+d}]_1$
$[R(\sigma)\tau]_1$	$R(Z) \in \mathbb{F}_{\leq n_k - 2}[Z]$	$[(\sigma^i\tau)_{i=0}^{n_k-2}]_1$
$[Q(\sigma)\tau]_1$	$Q(Z) \in \mathbb{F}_{\leq n_k - 3}[Z]$	$[(\sigma^i\tau)_{i=0}^{n_k-3}]_1$
$[B_{\mathsf{pc}}(\sigma)]_1$	$B_{\mathsf{pc}}(Y) \in \mathbb{F}_{\leq d-1}[Y]$	$[(\sigma^i)_{i=0}^{d-1}]_1$

Recalling the definitions of d and d_{gap} from Eqs. (10) and (12) (in particular, $n_h < d = d_{\mathsf{gap}}$), the prover needs the following elements to be in the SRS: $[(\sigma^i)_{i=0:i\neq d_{\mathsf{gap}}}^{d_{\mathsf{gap}}+d}, (\sigma^i\tau)_{i=0}^{n_k-2}]_1 = [g(\sigma, \tau) : g \in \mathcal{S}_1]_1$.

For the verifier's computation to succeed, the SRS needs to contain the following elements:

- To check ($\sharp$) in Fig. 2, the SRS needs to contain $([\tau]_1, [1, \sigma]_2)$. Moreover, $\mathsf{srs}_{\mathcal{R}}$ needs to contain $[\mathsf{rcv}(\sigma), \mathsf{col}(\sigma), \mathsf{row}(\sigma), \mathsf{rc}(\sigma), \mathsf{zcol}(\sigma), \mathsf{zrow}(\sigma), \mathsf{zrc}(\sigma), \mathcal{Z}_{\mathbb{K}}(\sigma)]_2$. All these elements of $\mathsf{srs}_{\mathcal{R}}$ can be computed from $[(\sigma^i)_{i=0}^{n_k}]_2$.
- To check ($\sharp\sharp$), the SRS needs to contain $([1]_1, [1, \sigma, S(\sigma), \sigma S(\sigma), \sigma^2 S(\sigma)]_2)$, where $[S(\sigma), \sigma S(\sigma), \sigma^2 S(\sigma)]_2$ can be computed in the preprocessing phase from $[((\sigma^{d_{\mathsf{gap}}+j-n_h i})_{j=0}^{2})_{i=0}^{\lfloor d/n_h \rfloor}]_2$.

Thus, $\mathsf{vk}_{\mathcal{R}}$ has to contain $([1, \tau]_1, [1, \sigma]_2, \mathsf{srs}_{\mathcal{R}})$ (where $\mathsf{srs}_{\mathcal{R}}$ is as in Fig. 2), while $[(\sigma^i)_{i=0}^{n_k}, ((\sigma^{d_{\mathsf{gap}}+j-n_h i})_{j=0}^{2})_{i=0}^{\lfloor d/n_h \rfloor}]_2$ is needed in the preprocessing phase to compute $\mathsf{srs}_{\mathcal{R}}$. This explains the definition of srs, $\mathsf{srs}_{\mathcal{R}}$, $\mathsf{ek}_{\mathcal{R}}$, and $\mathsf{vk}_{\mathcal{R}}$ in Fig. 2.

Altogether, srs contains $(d_{\mathsf{gap}} + d) + (n_k - 2 + 1) = (6(n_h - 1)) + (n_k - 1) = 6n_h + n_k - 7 = 12m + n_k + 6b - 7 = 12m + n_k + 17$ elements of $\mathbb{G}_1$ and at most $n_k + 1 + 3 \cdot 3 = n_k + 10$ elements of $\mathbb{G}_2$.

If $n_k \gg m$ (in most applications, $n_k > 4m$ and choosing an even larger n_k seems to be prudent), then the SRS length is dominated by n_k elements of $\mathbb{G}_1$ and $\mathbb{G}_2$.

Computational Complexity of KGen**.** The key generation is dominated by the need to compute all SRS elements and is thus the same as SRS length but in different units (scalar multiplications).

Complexity of Derive**.** In Derive, one needs to compute $[f(\sigma)]_2$ for seven degree-$(\leq n_k - 1)$ polynomials (thus $7(n_k - 1)$ scalar multiplications in $\mathbb{G}_2$), three degree-$\approx d_{\mathsf{gap}}$ polynomials $S(Y), YS(Y), Y^2S(Y)$ ($\approx 3d_{\mathsf{gap}} = 3 \cdot 3(n_h - 1) = 9(2m + b - 1) = 18m + 27$ scalar multiplications in $\mathbb{G}_2$), and $[\mathcal{Z}_{\mathbb{K}}(\sigma)]_1$ (one additional scalar multiplication). Thus, the computational complexity is $\approx 18m + 7n_k$ scalar multiplications in $\mathbb{G}_2$.

Prover's Computation. The prover's computation is quasilinear, like in other efficient updatable zk-SNARKs with constant communication. More precisely, P needs a quasilinear number of $\mathbb{F}$ operations to compute $z(Y) \leftarrow \mathcal{Z}_{\mathsf{inp}}(Y)\tilde{z}(Y) + \mathsf{inp}(Y)$, $R(Z)$ and $Q(Z)$, and polynomial openings $\tilde{z}_{\mathsf{pc}}(Y)$ and $\Phi_{\mathsf{pc}}(Y)$.

In addition, the prover needs a linear number of scalar multiplications to compute the values $[\tilde{z}(\sigma), \psi_{\mathsf{ipc}}(\sigma), R(\sigma)\tau, Q(\sigma), B_{\mathsf{pc}}(\sigma)]_1$. More precisely, to compute $[f(\sigma)]_1$ for some f, the prover has to execute $\deg f+1$ scalar multiplications. Now, note that $\tilde{z} \in \mathbb{F}_{\leq n_h-2m_0-3}[Y]$, $\psi_{\mathsf{ipc}} \in \mathsf{PolyPunc}_{\mathbb{F}}(d, d_{\mathsf{gap}}, Y)$, $R \in \mathbb{F}_{\leq n_k-2}[Z]$, $Q \in \mathbb{F}_{\leq n_k-3}[Z]$, and $B_{\mathsf{pc}} \in \mathbb{F}_{\leq d-1}[Z]$ correspondingly. Moreover, $d = 3(n_h-1)$ (see Eq. (10)), $d_{\mathsf{gap}} = 3(n_h-1)$ (see Eq. (12)), $n_h = 2m+b$, and $b = 4$. Hence, the total number of scalar multiplications is $(n_h - 2m_0 - 3 + 1) + (d_{\mathsf{gap}} + d - 1 + 1) + (n_k - 2 + 1) + (n_k - 3 + 1) + (d - 1 + 1) = d_{\mathsf{gap}} + 2d + n_h + 2n_k - 2m_0 - 5 = 10n_h + 2n_k - 2m_0 - 14 = 20m + 2n_k - 2m_0 + 10b - 14 = 20m + 2n_k - 2m_0 + 26$.

Using $\mathfrak{Count}$ instead of Aurora's sumcheck eliminates one FFT but adds cryptographic operations. Decreasing the prover's computation is an interesting open question.

Verifier's Computation. The verifier's computation is dominated by the computation of $\mathsf{inp}(\beta)$, $\mathcal{Z}_{\mathsf{inp}}(\beta)$ (see Eq. (8)), and $\Lambda^b_{\mathbb{H}}(\alpha, \beta)$ (see Eq. (6)), in total $\Theta(m_0 + \log n_h)$ field multiplications. Otherwise, the verifier executes three scalar multiplications in $\mathbb{G}_1$, 13 scalar multiplications in $\mathbb{G}_2$ (this number can be slightly optimized), and seven pairings.

Summatory Efficiency Comparison. See Table 2 for efficiency comparison with previous work. Essentially, we copied the efficiency comparison table of Table 2 from [RZ21b] and added one additional entry for $\mathfrak{Vampire}$. Clearly, it makes sense to compare the efficiency of zk-SNARKs for the constraint system that underlies $\mathfrak{Vampire}$, that is, R1CSLite with sparse matrices. (We denote corresponding rows in Table 2 by bold font.) Pink cells contain the absolutely best (most optimal) entries while yellow cells contain the absolute best entries as functions of n_k only.

As we see from Table 2, $\mathfrak{Vampire}$ has very good efficiency when measured as a function of N_k / n_k and somewhat worse efficiency as a function of M / m. In other words, $\mathfrak{Vampire}$ is very competitive when N_k / n_k are relatively large compared to M / m. In applications, we think it is reasonable to assume that $N_k \gg M$ since it allows to implement circuits of high fan-in. Inefficiency in m follows from our strategy of optimizing the argument length. For example, the fact that we use a single polynomial to commit both to $\boldsymbol{z}_{\mathsf{l}}$ and $\boldsymbol{z}_{\mathsf{r}}$ increases n_h twice from $m+b$ to $2m+b$.

Let us now ignore n_k-independent terms (we can do it when say $N_k > 20M$). Then, $\mathfrak{Vampire}$'s SRS has the same length as LunarLite's and is three times shorter than Marlin's; the same holds for the complexity of KGen. The complexity of Derive is $7n_k$ in the case of $\mathfrak{Vampire}$, which is only beaten in the case of the RZ21 zk-SNARK for the same constraint system; however, the latter has a twice longer argument. The prover's computation is $2n_k$, compared to $4n_k$ in the case of Basilisk (for the same arithmetization), $3n_k$ in the case of LunarLite, and $8n_k$ in the case of Marlin. We emphasize that $\mathfrak{Vampire}$ has the best prover computation,

Table 2. Efficiency comparison from [RZ21a]: updatable and universal zk-SNARKs. Here, m_0 is the number of public input wires, m is the number of multiplicative gates, n_g is the number of total gates, v: bounded fan-out, n_k: non-zero elements of the matrix that describe the circuit, a is the number of additive gates, n_g, N_k, A, M, V are maximum supported values for n_g, n_k, a, m, v.

Scheme	$\lvert\mathsf{srs}\rvert$	$\lvert\mathsf{srs}_{\mathcal{R}}\rvert$	$\lvert\pi\rvert$	KGen	Derive	Prove	Verify	Language
Sonic	$\mathbb{G}_1$ $4M$	–	20	$4M$	$36m$	$273m$	$7P$	
[MBKM19]	$\mathbb{G}_2$ $4M$	3	–	$4M$	–	–		[BCC+16] constraints
	$\mathbb{F}$ –	–	16	–	$O(n_k \log n_k)$	$O(n_k \log n_k)$	$O(m_0 + \log n_k)$	
Marlin	$\mathbb{G}_1$ $3N_k$	12	13	$3N_k$	$12n_k$	$14m + 8n_k$	$2P$	**R1CS with sparse matrices**
[CHM+20]	$\mathbb{G}_2$ 2	2	–	–	–	–		
	$\mathbb{F}$ –	–	8	–	$O(n_k \log n_k)$	$O(n_k \log n_k)$	$O(m_0 + \log n_k)$	
Plonk	$\mathbb{G}_1$ $3n_g$	8	7	$3n_g$	$8n_g$	$11n_g$	$2P$	
[GWC19]	$\mathbb{G}_2$ 1	1	–	–	–	–		Plonk constraints
	$\mathbb{F}$ –	–	7	–	$O(n_g \log n_g)$	$O(n_g \log n_g)$	$O(m_0 + \log n_g)$	
LunarLite2x	$\mathbb{G}_1$ N_k	16	11	N_k	$16n_k$	$8m + 4n_k$	$2P$	**R1CSLite with sparse matrices**
[CFF+21]	$\mathbb{G}_2$ 1	1	–	1	–	–		
	$\mathbb{F}$ –	–	3	–	$O(n_k \log n_k)$	$O(n_k \log n_k)$	$O(m_0 + \log n_k)$	
LunarLite	$\mathbb{G}_1$ N_k	–	10	N_k	–	$8m + 3n_k$	$7P$	**R1CSLite with sparse matrices**
[CFF+21]	$\mathbb{G}_2$ N_k	27	–	N_k	$24n_k$	–		
	$\mathbb{F}$ –	–	2	–	$O(n_k \log n_k)$	$O(n_k \log n_k)$	$O(m_0 + \log n_k)$	
RZ21	$\mathbb{G}_1$ N_k	4	10	N_k	$6n_k$	$6m + 4n_k$	$2P$	**R1CSLite with sparse matrices**
(sparse matrices)	$\mathbb{G}_2$ –	–	–	–	–	–		
[RZ21a], §5.3	$\mathbb{F}$ –	–	3	–	$O(n_k \log n_k)$	$O(n_k \log n_k)$	$O(m_0 + \log n_k)$	
RZ21 ("Plonk")	$\mathbb{G}_1$ n_g	11	8	n_g	$11n_g$	$8n_g$	$2P$	
[RZ21b], Fig 11	$\mathbb{G}_2$ 1	1	–	–	–	–		Plonk constraints
	$\mathbb{F}$ –	–	4	–	$O(n_g \log n_g)$	$O(n_g \log n_g)$	$O(m_0 + \log n_g)$	
Basilisk	$\mathbb{G}_1$ M	$3V + 1$	6	M	$(3v + 1)m$	$6m$	$2P$	weighted R1CS with bounded fan-out
[RZ21b], App F	$\mathbb{G}_2$ 1	1	–	–	–	–		
	$\mathbb{F}$ –	–	2	–	$O(m \log m)$	$O(m \log m)$	$O(m_0 + \log m)$	
Vampire	$\mathbb{G}_1$ $12M + N_k + 17$	–	5	$12M + N_k + 17$	–	$20m + 2n_k$	$3\mathbb{G}_1 + 13\mathbb{G}_2 + 7P$	**R1CSLite with sparse matrices**
(sparse matrices)	$\mathbb{G}_2$ $N_k + 10$	11	–	$N_k + 10$	$18m + 7n_k$	–		
current paper	$\mathbb{F}$ –	–	2	–	$O(n_k \log n_k)$	$O(n_k \log n_k)$	$O(m_0 + \log m)$	

as a function of n_k, among any known updatable zk-SNARKs. The verifier needs to execute seven pairings, compared to seven in LunarLite and two in most of the other SNARKs.