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

Version 2.0, Thursday 23rd June, 2022, 12:58

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 (four group and two finite field elements) for NP. In addition, $\mathfrak{Vampire}$ uses the aggregated polynomial commitment scheme of Boneh *et al.*

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) [18,26,17,30] 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) [17], 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, say, verifiable computation and cryptocurrencies.

Non-interactive zero-knowledge 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 [19] for R1CS with a relation-specific SRS has an argument that consists of only three group elements.

A significant practical downside of such "non-universal" SNARKs is that one has to construct a new SRS for every instance of the constraint system. This observation has spurred an enormous effort to design universal zk-SNARKs, i.e.,

* This is the second, more optimized, version of $\mathfrak{Vampire}$. We point out the main optimization in Footnote 8. The old version can be recovered from [27].

zk-SNARKs with an SRS that only depends on an upper bound on $\mathcal{R}$'s size. In addition, it is crucial to decrease the amount of trust in the SRS creator. A popular approach is to design *updatable and universal zk-SNARKs* [20,29], 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, by "updatable" we will sometimes mean "updatable and universal".

Plonk [16] and Marlin [12] are the first genuinely efficient universal zk-SNARKs. Marlin and many subsequent updatable and universal zk-SNARKs [31,11] work for sparse R1CS instances, where the underlying matrices contain a linear (instead of quadratic) number of non-zero elements. Chiesa *et al.* [12] first define an information-theoretic model, algebraic holographic proof (AHP). An AHP is an interactive protocol, where at each step, the prover sends polynomial oracles, and the verifier sends to the prover random field elements. Polynomial oracles are usually implemented using polynomial commitments [22]. In the end, the verifier queries the polynomial oracles and performs some low-degree tests. After that, [12] proposes a new AHP for sparse R1CS, and then compiles it to a zk-SNARK named Marlin.

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 [28] were for multivariate polynomials but small domains. Ben-Sasson *et al.* [5] proposed a univariate sumcheck argument for large domains 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 [11] improves on Marlin in several aspects. First, they define PHPs, a generalization 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 of $\mathcal{R}$ being the identity matrix. Moreover, they provide a more fine-grained analysis of the zero-knowledge property and implement several additional optimizations.

Basilisk [31] gains additional efficiency by using a different technique to obtain zero-knowledge and constructing a "free" low-degree test. In addition, [31] constructs even more efficient zk-SNARKs for somewhat more limited constraint systems. Both Lunar and Basilisk introduce new theoretical frameworks; e.g., Basilisk introduces checkable subspace 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.

[34] proposed Vector Oracle proofs (VOProofs), a new information-theoretic model based on vector operations. They use it to construct efficient zk-SNARKs for several well-known constraint systems such as R1CS (VOR1CS) and Plonk's constraint system (VOPlonk).

Finally, we mention attempts to add updatability and universality to Groth's zk-SNARK [19]. [23] proves that Groth's zk-SNARK is two-phase updatable (a

Table 1. Comparison of some known updatable and universal zk-SNARKs.

Scheme	Argument length		Arithmetization
	Elements	Bits	
Updatable and universal zk-SNARKs			
Sonic [29]	$20\|\mathbb{G}_1\| + 16\|\mathbb{F}\|$	11776	[8] constraints
Marlin [12]	$13\|\mathbb{G}_1\| + 8\|\mathbb{F}\|$	7040	R1CS, sparse matrices
Basilisk [31]	$10\|\mathbb{G}_1\| + 3\|\mathbb{F}\|$	4608	R1CSLite, sparse matrices
Plonk [16]	$7\|\mathbb{G}_1\| + 7\|\mathbb{F}\|$	4480	Plonk constraints
LunarLite [11]	$10\|\mathbb{G}_1\| + 2\|\mathbb{F}\|$	4352	R1CSLite, sparse matrices
Basilisk [31]	$8\|\mathbb{G}_1\| + 4\|\mathbb{F}\|$	4096	Plonk constraints
VOR1CS* [34]	$9\|\mathbb{G}_1\| + 2\|\mathbb{F}\|$	3968	R1CS, sparse matrices
VOPlonk* [34]	$7\|\mathbb{G}_1\| + 2\|\mathbb{F}\|$	3200	Plonk constraints
Basilisk (full version, [32])	$6\|\mathbb{G}_1\| + 2\|\mathbb{F}\|$	2816	Weighted R1CS with bounded fan-out
Vampire (this work)	$4\|\mathbb{G}_1\| + 2\|\mathbb{F}\|$	2048	R1CSLite, sparse matrices
Non-universal zk-SNARKs (relation-specific SRS)			
Groth16 [19]	$2\|\mathbb{G}_1\| + 1\|\mathbb{G}_2\|$	1536	R1CS

weaker version of updatability) but *not* universal. [24] proposes a modification of Groth's zk-SNARK, which is universal (but *not* updatable); this comes with significant overhead in the prover's running time.

In Table 1, we overview the argument lengths of the most efficient updatable and universal zk-SNARKs. Here, $|X|$ denotes the representation length of an element from X in bits, given the BLS12-381 curve [10], with $|\mathbb{G}_1| = 384$, $|\mathbb{G}_2| = 768$, and $|\mathbb{F}| = 256$. Thus, even the most efficient updatable and universal zk-SNARK has an approximately two times longer argument than Groth16 [19].

Moreover, Groth16 works for QAP [17] (i.e., full R1CS), while the most efficient variant of Basilisk works for instances of R1CS 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). Thus, there is still a non-trivial difference between the communication efficiency of relation-specific zk-SNARKs and updatable and universal zk-SNARKs.

1.1 Our Contributions

The current paper has three related contributions of independent interest:

1. The combined use of polynomial commitments and inner-product commitments in the sumcheck and updatable and universal 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 commitments. In particular, ILV inner-product commitments [21] use a SRS made of non-consequent monomial powers.[3]
2. A new updatable (and universal) 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, [9]), we believe $\mathfrak{Count}$ will have wider interest.
3. A new updatable and universal zk-SNARK $\mathfrak{Vampire}$ for sparse R1CSLite with the smallest argument length among all known updatable and universal zk-SNARKs for NP-complete languages. (See Table 1.) $\mathfrak{Vampire}$ uses $\mathfrak{Count}$ and thus inner-product commitments.

1.2 Our Techniques

Non-Consequent Monomial SRSs. Groth *et al.* [20] 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[4] $[1, \sigma, \sigma^3, \sigma^4]_1 \in \mathbb{G}_1^4$ for a trapdoor σ, 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 gaps. Similarly, the SRS can contain multivariate monomials. However, most of the known updatable and universal zk-SNARKs ([20,29] being exceptions) use SRSs that consist of consequent univariate monomials only, i.e., are of the shape $([(\sigma^i)_{i=0}^{m_1}]_1, [(\sigma^i)_{i=0}^{m}]_2)$ for some m_i.

One reason why efficient updatable and universal zk-SNARKs use a consequent monomial SRS is their reliance on polynomial commitment schemes like KZG [22] that have such SRSs. While other polynomial commitment schemes are known, up to our knowledge, no efficient one uses non-consequent monomial SRSs. In particular, AHP [12] and PHP [11] model polynomial commitments as polynomial oracles and allow the parties to perform operations (e.g., queries to committed oracles and low-degree tests) related to such oracles. Low-degree tests model consequent monomial SRSs: a committed polynomial is a degree-$\leq m$ polynomial iff it is in the span of X^i for $i \leq m$.

One can use non-consequent monomial SRSs to efficiently construct protocols like broadcast encryption and inner-product commitments [25,21]. We use non-consequent monomial SRSs in the context of sumchecks and updatable and universal zk-SNARKs. We will not define an information-theoretic model, but we mention two possible approaches that both have their limitations. First, the pairing-based setting can be modeled as linear interactive proofs (LIPs, [6]) or non-interactive LIPs [19]. 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 natural in such a setting (e.g., one can efficiently

[3] 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.

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

"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 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 restricts one to a limited number of cryptographic tools (polynomial and inner-product commitment schemes), with a need to redefine the model when more tools are found 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. 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 value $v_M \in \mathbb{F}$ over $\mathbb{H}$.

Let $n_h := |\mathbb{H}|$ and $\mathcal{Z}_{\mathbb{H}}(X) := \prod_{\chi \in \mathbb{H}}(X - \chi)$. Aurora's sumcheck [5] relies on the fact that $\sum_{\chi \in \mathbb{H}} f(\chi) = n_h f(0)$, when $f \in \mathbb{F}_{\leq n_h - 1}[X]$ is a polynomial with $\deg f \leq n_h - 1$. Then, for $f \in \mathbb{F}[X]$ 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) $f(X) = v_f/n_h + XR(X) + Q(X)\mathcal{Z}_{\mathbb{H}}(X)$. In a cryptographic implementation of Aurora's sumcheck argument in say Marlin [12], the prover uses KZG [22] 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.

Based on the ILV inner-product commitment [21], we construct a new sumcheck argument $\mathfrak{Count}$ for $f \in \mathbb{F}_{\leq d}[X]$. ILV's 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 $\boldsymbol{\mu} \in \mathbb{Z}_p^N$ as $[\mu(\sigma)]_1 \leftarrow \sum_{j=1}^{N} \mu_j [\sigma^j]_1$. When the verifier outputs $\boldsymbol{\nu} \in \mathbb{Z}_p^N$, the prover returns the inner product $v \leftarrow \boldsymbol{\mu}^\top \boldsymbol{\nu}$ together with a short evaluation proof (a single group element $[\mathsf{op}]_1$) that v is correctly computed. ILV's security depends on the fact that $[\sigma^N]_1$ is not in the SRS.

We present an alternative extension of the equality $\sum_{\chi \in \mathbb{H}} f(\chi) = n_h f(0)$ 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 $n_h \cdot i$ for $i \leq \lfloor d/n_h \rfloor$.

In $\mathfrak{Count}$, 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) in Aurora's sumcheck. Moreover, there is no need for a low-degree test, making $\mathfrak{Count}$ even more efficient. In addition, in the application to $\mathfrak{Vampire}$, $\boldsymbol{s}$ has a small constant number of non-zero elements. Thus, differently from Aurora's sumcheck, the prover's computation is linear in both field operations and group operations. An explicit cost of using ILV is that the SRS becomes larger: if the SRS, without $\mathfrak{Count}$, contains $[(\sigma^i)_{i=0}^{d}]_1$ (where d is some constant, fixed by the rest of the zk-SNARK), it now has to contain also $[(\sigma^i)_{i=d+2}^{2d}]_1$ and $[(\sigma^i)_{i=0}^{d}]_2$. (Although, in our construction, we will add significantly less elements to $\mathbb{G}_2$.)

Since sumchecks have ubiquitous applications [9], $\mathfrak{Count}$ is of independent interest. In particular, univariate sumcheck is used in both updatable and universal zk-SNARKs and transparent zk-SNARKs. As an important application, we will design a new updatable and universal zk-SNARK. We leave it an interesting open question to apply $\mathfrak{Count}$ in transparent zk-SNARKs.

New zk-SNARK. We use $\mathfrak{Count}$ to design a new pairing-based updatable and universal zk-SNARK $\mathfrak{Vampire}$ for the sparse R1CSLite constraint system [11]. $\mathfrak{Vampire}$'s argument length is four elements of $\mathbb{G}_1$ and two elements of $\mathbb{F}$, which is less than in any known updatable and universal zk-SNARK. While Basilisk [31] (as improved in the full version, [32]) has just 37.5% 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 arithmetization handled by $\mathfrak{Vampire}$ is less communication-efficient than LunarLite or VOR1CS*.

Let us now give a very brief glimpse to the structure of $\mathfrak{Vampire}$. (The real description, with a very long intuition behind $\mathfrak{Vampire}$'s construction, is given in Section 4.) 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. Let m be the number of constraints. 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 $\mathbb{F}$.

Let z be the interpolating polynomial of $(\mathbb{x}, \mathbb{w}, \boldsymbol{r}_z)$, where $\boldsymbol{r}_z$ is a short random vector needed for zero-knowledge. The prover starts by committing to $\tilde{z}$, where $\tilde{z}$ is a polynomial related to z. Using $\tilde{z}$ helps one efficiently check that the prover used the correct public input. The verifier replies with a random field element α. We reformulate the check that $(\mathbb{x}, \mathbb{w})$ (where $\mathbb{w}$ is encoded in $\tilde{z}$) satisfies the R1CSLite instance as a univariate sumcheck argument that $\sum_{y \in \mathbb{H}} \psi_\alpha(y) = 0$, for a well-chosen polynomial ψ_α. We then run $\mathfrak{Count}$, letting the prover send an ILV-opening $[\psi_{\mathsf{ipc}}(\sigma)]_1$ of ψ_α to the verifier. The verifier replies with another random field element β. The prover's final message consists of two field elements and two group elements. These elements are needed to batch-open three polynomial commitments at different locations, two of which are related to β. It involves a complicated but by now standard step of proving the correctness of the arithmetization of a sparse matrix. This step involves using a univariate sumcheck the second time. However, since here the summed polynomial is of a small degree, we do not need to use $\mathfrak{Count}$. We refer to Section 4 for more details.

$\mathfrak{Vampire}$ is based on the ideas of Marlin (e.g., we use a similar arithmetization of sparse matrices), but it uses optimizations of both Lunar [11] and Basilisk [31]. 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 finite field elements, which is already comparable to prior shortest updatable and universal zk-SNARKs for any NP-complete constraint system.

$\mathfrak{Count}$ helps to remove one more group element from the argument of $\mathfrak{Vampire}$. This step is not trivial: the sumcheck argument requires that the sumchecked 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 [7]. Intuitively, we commit to a single polynomial $\tilde{z}$ 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 three polynomials at different points. In particular, we aggregate the commitment of the second sumcheck, further reducing the proof size by one group element. For batching, we use a technique of Boneh *et al.* [7]. However, differently from [7], our batching is not randomized since the two opening points are different.

In Theorem 1, we prove that $\mathfrak{Vampire}$ is knowledge-sound in the Algebraic Group Model (AGM, [15]). The proof structure is standard, involving two branches depending on whether the verifier's equations hold as polynomials (we get a reduction to the well-known Power Discrete Logarithm assumption if not). However, the proof of the former case is quite complicated, partially since one has to consider several different polynomials sent by the prover, which depend on different verifier's challenge values.

We prove that $\mathfrak{Vampire}$ is perfectly zero-knowledge by constructing a simulator that uses the knowledge of trapdoor to make the sumcheck argument acceptable for any, even an all-zero witness. For a simulated argument to be indistinguishable from the real one, we add random terms ($\boldsymbol{r}_z$) to polynomial $\tilde{z}(X)$ which, in the case of real argument, encodes the witness, 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 apart commitments to $\tilde{z}(X)$ in real and simulated arguments. In Appendix B, we prove that $\mathfrak{Vampire}$ is also Sub-ZK (i.e., zero-knowledge even if the SRS generation is compromised, [4,1,14,2]) under the BDH-KE knowledge assumption [1].

On Efficiency. We study how much the argument length can be reduced in updatable and universal SNARKs while only allowing minimal relaxations in other efficiency parameters. We achieve the shortest argument by far. The SRS size of our zk-SNARK is a constant factor larger than in the previous work, which we believe is a reasonable compromise as the SRS needs to be transferred only once. $\mathfrak{Vampire}$ is especially advantageous when the R1CSLite instance is not so sparse. As a function of the number of non-zero elements in R1CSLite matrices only, $\mathfrak{Vampire}$ has the best prover's computation as any previous updatable and universal zk-SNARK. Importantly, the verifier has only to execute $O(|\mathbb{x}|)$ field operations as opposed to $O(|\mathbb{x}|)$ group operations in [19].

However, differently from the prior work, prover's computation time in $\mathfrak{Vampire}$ depends on the largest supported R1CSLite size. We discuss this issue further and give a thorough efficiency comparison in Appendix A.

Demaking $\mathfrak{Vampire}$. It is possible to "demake" $\mathfrak{Vampire}$ by removing some of the aggressive length-optimization to obtain a larger argument size but better (say) the SRS size. We leave it as an open question about which optimization should be removed first or whether this is needed at all.

2 Preliminaries

Let $\mathbb{F} = \mathbb{Z}_p$ be a finite field of prime order p, and let $\mathbb{F}_{\le d}[X] \subset \mathbb{F}[X]$ be the set of degree $\le d$ polynomials. 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. 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,

- For any $T \subset \mathbb{F}$, the *vanishing polynomial* $\mathcal{Z}_T(X) := \prod_{i\in T}(X - i)$ is the degree-$|T|$ monic polynomial, such that $\mathcal{Z}_T(i) = 0$ for all $i \in T$. $\mathcal{Z}_{\mathbb{H}}(Y) = Y^{n_h} - 1$ can be computed in $\Theta(\log n_h)$ field operations.
- 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 $\ell_i^{\mathbb{H}}(Y) = \mathcal{Z}_{\mathbb{H}}(Y)/(\mathcal{Z}'_{\mathbb{H}}(\omega^{i-1}) \cdot (Y - \omega^{i-1})) = \mathcal{Z}_{\mathbb{H}}(Y)\omega^{i-1}/(n_h(Y - \omega^{i-1}))$. Here, $\mathcal{Z}'_{\mathbb{H}}(X) = d\mathcal{Z}_{\mathbb{H}}(X)/dX$.
- $L_X^{\mathbb{H}}(Y) := \mathcal{Z}_{\mathbb{H}}(Y)X/\left(n_h(Y - X)\right) \in \mathbb{F}(X, Y)$ (a lifted Lagrange rational function), 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 [11,31] is a variant of the Rank 1 Constraint System [17,12]. An R1CSLite instance $\mathcal{I} = (\mathbb{F}, m, m_0, \mathbf{L}, \mathbf{R})$ 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.

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

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

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 $\mathcal{A}$, $r \leftarrow\!\$\ \mathsf{RND}_\lambda(\mathcal{A})$ samples random coins of sufficient length for $\mathcal{A}$ for fixed

λ. 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, $\mathbb{G}_1$, $\mathbb{G}_2$, 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 this paper, $\mathbb{F} = \mathbb{Z}_p$ has always 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; then, $|\mathbb{K}|, |\mathbb{H}| \mid 2^{32}$.

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 [22], 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 [22] polynomial commitment scheme consists of the following algorithms:

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

Commitment key generation: Given a system parameter p and an upper-bound d on the polynomial degree, 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 $f_{\mathsf{pc}}(X) \leftarrow (f(X) - v)/(X - \beta)$. The evaluation proof is $[f_{\mathsf{pc}}(\sigma)]_1 \leftarrow \sum_{j=0}^{d-1} (f_{\mathsf{pc}})_j [\sigma^j]_1$. 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 $[f(\sigma) - v]_1 \bullet [1]_2 = [f_{\mathsf{pc}}(\sigma)]_1 \bullet [\sigma - \beta]_2$.

KZG's 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 [25,21], 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 [21] 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 a system parameter 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 $\mu(X) \leftarrow \sum_{j=1}^N \mu_j X^j \in \mathbb{F}_{\le N}[X]$; $[\mu(\sigma)]_1 = \sum_{j=1}^N \mu_j [\sigma^j]_1$. 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 $\mu_{\mathsf{ipc}}(X) \leftarrow \mu(X)\nu^*(X) - vX^{N+1} \in \mathsf{PolyPunc}_{\mathbb{F}}(N-1, N+1, X)$. The evaluation proof is $[\mu_{\mathsf{ipc}}(\sigma)]_1 \leftarrow \sum_{i=1, i\neq N+1}^{2N} \mu_{\mathsf{ipc}}[\sigma^i]_1$. Return $(v, [\mu_{\mathsf{ipc}}(\sigma)]_1)$.

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

ILV's security follows since the coefficient of X^{N+1} in $\mu_{\mathsf{ipc}}(X)$ is $\boldsymbol{\mu}^\top \boldsymbol{\nu} - v = 0$ iff v is correctly computed. In this paper, the vector $\boldsymbol{\nu}$ is public and known in advance. Then, the verifier only has to compute two pairings and no exponentiations.

Succinct Zero-Knowledge Arguments. The following definition is based on [11]. Groth *et al.* [20] introduced the notion of (preprocessing) zk-SNARKs with specializable universal structured reference string (SRS). This notion formalizes the idea that the key generation for $\mathcal{R} \in \mathcal{UR}$, where $\mathcal{UR}$ is a universal relation, can be seen as the sequential combination of two steps. First, a probabilistic algorithm generating an SRS for $\mathcal{UR}$ and second, a deterministic algorithm specializing 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 $N \in \mathsf{poly}(\lambda)$.[5] A *succinct zero-knowledge argument* $\Pi = (\mathsf{Pgen}, \mathsf{KGen}, \mathsf{Derive}, \mathsf{P}, \mathsf{V})$ *with specializable universal SRS for a relation family* $(\mathsf{Pgen}, \{\mathcal{UR}_{\mathsf{p},N}\}_{\mathsf{p}\in\{0,1\}^*,N\in\mathbb{N}})$ consists of the following algorithms.

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 and an upper bound N on the relation size, and outputs $\mathsf{srs} = (\mathsf{ek}, \mathsf{vk})$ together with a trapdoor. We assume implicitly that 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 a 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 $\mathbb{w}$, s.t. $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, and V takes a verification key for a relation $\mathcal{R}$ and a statement $\mathbb{x}$, and either accepts ($b = 1$) or rejects ($b = 0$) the argument.

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 $\mathsf{poly}(\lambda + \log|\mathbb{w}|)$.

[5] $\mathfrak{Count}$ and $\mathfrak{Vampire}$ have several size bounds. The definitions generalize naturally.

Knowledge-Soundness. Π is *knowledge-sound*, if for every non-uniform PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a non-uniform PPT extractor $\mathsf{Ext}_{\mathcal{A}}$ [6], s.t.

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

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 $N \in \mathsf{poly}(\lambda)$,

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

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

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, [4]), 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), SRS verifiability (there exists a PPT algorithm that checks that the SRS belongs to range(KGen)), and a SNARK-specific knowledge assumption, [1,2]. We will provide the formal definition in Appendix B.

Π is *updatable* [20], 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. [20] 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 [13] to obtain a zk-SNARK.

Sumcheck Arguments. In a sumcheck argument [28] 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, [9], 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 [5].

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.* [5] proposed an efficient univariate sumcheck ("Aurora's sumcheck") for multiplica-

[6] Note that although the protocol is interactive, extraction is done non-interactively. This is sometimes called straight-line extractability.

tive 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.

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, Ben-Sasson *et al.* [5] used Fact 2 to construct Aurora's sumcheck argument for proving that $\sum_{\chi\in\mathbb{H}} f(\chi) = v_f$.

Fact 2 (Core Lemma of Aurora's Sumcheck) *Let $f \in \mathbb{F}[X]$ with $d = \deg f \geq n_h$. 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 $f(X) = v_f/n_h + R(X)X + Q(X)\mathcal{Z}_{\mathbb{H}}(X)$.*

Assume that $d = \deg f = \mathsf{poly}(\lambda)$ while $p = 2^{\Theta(\lambda)}$. In Aurora's sumcheck argument, the prover sends to the verifier polynomial commitments to f, R, and Q. 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 $\deg R \leq n_h - 2$. As the low-degree test, Aurora uses an interactive oracle proof for testing proximity to the Reed–Solomon code, resulting in additional costs. The full version of Basilisk [32] implementes a low-degree test in a partially costless way (without added argument size or verifier's computation); however, one may need to add a large number of elements to the SRS for their low-degree test to succeed.

Assumptions. Let $d_1(\lambda), d_2(\lambda) \in \mathsf{poly}(\lambda)$. Pgen is (d_1, d_2)-*PDL (Power Discrete Logarithm [26]) 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 [15] 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 provides an algebraic representation of the latter in terms of the former. More precisely, if $\mathcal{A}$ has received group elements $[\boldsymbol{x}_1]_1, [\boldsymbol{x}_2]_2$ so far and outputs $[y_1]_1, [y_2]_2$, then there exists an extractor $\mathsf{Ext}_{\mathcal{A}}$ which on the same input and random coins outputs integer vectors $\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2$ such that $[y_1]_1 = \sum_i \gamma_{1,i}[\boldsymbol{x}_{1,i}]_1$ and $[y_2]_2 = \sum_j \gamma_{2,j}[\boldsymbol{x}_{2,j}]_2$.

3 Count: New Univariate Sumcheck Argument

In this section, we propose Count, a new sumcheck argument with 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,* $\sum_{\chi\in\mathbb{H}} f(\chi) = n_h \cdot \sum_{i=0}^{\lfloor d/n_h \rfloor} f_{n_h i}$.

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* $\mathbb{H}$ *be an order-*$n_h > 1$ *multiplicative subgroup of* $\mathbb{F}^*$. *Let* $d_{\mathsf{gap}}, d > 0$ *with* $d_{\mathsf{gap}} \geq n_h \cdot \lfloor d/n_h \rfloor$, *and* $f \in \mathsf{PolyPunc}_{\mathbb{F}}(d, d_{\mathsf{gap}}, X)$. *Define* $S(X) := \sum_{i=0}^{\lfloor d/n_h \rfloor} X^{d_{\mathsf{gap}} - n_h i} \in \mathbb{F}_{\leq d_{\mathsf{gap}}}[X]$. *Then,* $\sum_{\chi\in\mathbb{H}} f(\chi) = v_f$ *and* $\deg f \leq d$ *iff there exists* $f_{\mathsf{ipc}} \in \mathsf{PolyPunc}_{\mathbb{F}}(d, d_{\mathsf{gap}}, X)$, *s.t.*

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

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 $\mathfrak{Count}$. Next, we describe $\mathfrak{Count}$ as a zk-SNARK for the sumcheck relation; if needed, it is straightforward to modify it to the language of polynomial oracles. 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. Here, Derive does only preprocessing and does not do any specialization.

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, for the sake of completeness, we provide proof sketches.

Lemma 3. *The 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* $d_2 = d_{\mathsf{gap}}$.

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

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

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

Proof. Completeness follows from Lemma 2.

We sketch a knowledge-soundness proof in the AGM [15]. Since $\mathcal{A}$ is algebraic, $f(X), f_{\mathsf{ipc}}(X)$ are in the span of X^i for $i \in \mathcal{S}_1(X)$, 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 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 $\mathcal{V}$. We 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 construct a simulator that on input $(\mathsf{srs}, \mathsf{td} = \sigma, ([f(\sigma)]_1, v_f))$ outputs an argument indistinguishable from the real argument. The simulator just computes $[f_{\mathsf{ipc}}(\sigma)]_1$, such that the verification equation holds. That is, $[f_{\mathsf{ipc}}(\sigma)]_1 \leftarrow S(\sigma)[f(\sigma)]_1 - v_f/n_h \cdot \sigma^{d_{\mathsf{gap}}}[1]_1$. Zero-knowledge follows since in the real argument, $[f_{\mathsf{ipc}}(\sigma)]_1$ is computed the same way. □

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 monomial evaluations 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 field 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 field operations. Both univariate sumchecks however require the prover to use a linear number of $\mathbb{G}_1$ operations. Linear-time *multivariate* sumchecks are well-known, [33] 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. In $\mathfrak{Vampire}$, $d_{\mathsf{gap}} = d$.

As a drawback, $\mathfrak{Count}$'s 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 And Universal zk-SNARK

In this section, we will use $\mathfrak{Count}$ to construct an efficient updatable and universal zk-SNARK $\mathfrak{Vampire}$ for the sparse R1CSLite constraint system. At a very high level, we use the general approach of Marlin [12], taking into account optimizations of Lunar [11] and Basilisk [31]. 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 one, the sumcheck is for a low-degree polynomial and thus one can just use Fact 1.) Second, we use a variant of the aggregated polynomial commitment scheme of [7] to batch together the openings of three different polynomials at different points. While [7] 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. All the techniques together remove four group elements from the communication. As the end result, $\mathfrak{Vampire}$ is the most communication-efficient updatable and universal zk-SNARK for any NP-complete constraint system. (See Table 1 and Appendix A for an efficiency comparison.)

4.1 Formulating R1CSLite as Sumcheck

Let $\mathbb{F} = \mathbb{Z}_p$. As in [12,31,11], 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 the R1CSLite instance $\mathcal{I} = (\mathbb{F}, \mathbb{H}, \mathbb{K}, m, m_0, \mathbf{L}, \mathbf{R})$ includes descriptions of $\mathbb{H}$ and $\mathbb{K}$.

We want to demonstrate the satisfiability of $\mathcal{I}$. 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 $\boldsymbol{z}^* = (\boldsymbol{z}_{\mathsf{l}}^\top \| \boldsymbol{z}_{\mathsf{r}}^\top \| (\boldsymbol{z}_{\mathsf{l}} \circ \boldsymbol{z}_{\mathsf{r}})^\top)^\top$, where $\boldsymbol{z}_{\mathsf{l}}$ and $\boldsymbol{z}_{\mathsf{r}}$ 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 [32]. 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

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

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

$$\mathbf{W}^b \cdot \begin{pmatrix} \boldsymbol{z} \\ \boldsymbol{z} \circ \boldsymbol{z}' \end{pmatrix} = \mathbf{0} \ , \tag{3}$$

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 $z(Y) := \sum_{\chi \in \mathbb{H}} z[\chi] L^{\mathbb{H}}_{\chi}(Y) \in \mathbb{F}_{\le n_h - 1}[Y]$. 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, since $z(y\omega^m) = z[y\omega^m] = z'[y]$ for $y \in \{\omega^0, \ldots, \omega^{m-1}\}$, we get $\mathbf{M}^b[x,y] z'[y] z[y] = M^b(x,y) z(y\omega^m) z(y)$. On the other hand, 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 we multiply it by $M^b(x,y) = 0$.

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

$$P(X) := \textstyle\sum_{y \in \mathbb{H}} \psi(X,y), \tag{4}$$

$$\psi(X,Y) := \left(\Lambda^b_{\mathbb{H}}(X,Y) - M^b(X,Y) z(Y\omega^m)\right) z(Y) \quad . \tag{5}$$

To simplify it 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 [11], we interpolate $\mathbf{I}$ with the function

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

$\Lambda_{\mathbb{H}}$ satisfies the following properties: (1) $\Lambda_{\mathbb{H}}(x,y)$ is PPT computable, (2) $\Lambda_{\mathbb{H}}$ is a polynomial (this follows since $\mathcal{Z}_{\mathbb{H}}(X) Y - \mathcal{Z}_{\mathbb{H}}(Y) X = X - Y + XY(X^{n_h - 1} - Y^{n_h - 1}) = (X - Y)(1 + XY(\sum_{i=0}^{n_h - 2} X^{n_h - 2 - i} Y^i))$ divides by $X - Y$), (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., $\forall x, y \in \mathbb{H}. \Lambda_{\mathbb{H}}(x,y) = \mathbf{I}[x,y]$ (this follows since $\mathcal{Z}_{\mathbb{H}}(x) y - \mathcal{Z}_{\mathbb{H}}(y) x = 0$ for all $x \neq y \in \mathbb{H}$ and $1 + XY(\sum_{i=0}^{n_h - 2} X^{n_h - 2 - i} Y^i) = 1 + (n_h - 1) x^{n_h} = 1 + n_h - 1 = n_h$ when $X = Y = x \in \mathbb{H}$), (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 matrix encoding of $\mathbf{M}^b$ from [12] that keeps track of the matrix's non-zero entries. Let $\mathcal{NZ} := \{(i,j) \in \mathbb{H} \times \mathbb{H} : \mathbf{M}^b[i,j] \neq 0\}$ be the set of indices where $\mathbf{M}^b$ is non-zero. Let $\mathbb{K}$ be the minimum-size multiplicative subgroup of $\mathbb{F}$ such that $n_k := |\mathbb{K}| \geq |\mathcal{NZ}|$.[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 (by using the natural ordering of $\mathbb{H}^2$) 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

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

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,

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

We define $M^b \in \mathbb{F}[X,Y]$ that interpolates $\mathbf{M}^b$, as the low-degree extension of

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

Next, $\Lambda_{\mathbb{H}}(\mathsf{row}(\kappa), x) = (\mathcal{Z}_{\mathbb{H}}(\mathsf{row}(\kappa))x - \mathcal{Z}_{\mathbb{H}}(x)\mathsf{row}(\kappa))/(n_h(\mathsf{row}(\kappa) - x))$. 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,

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

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

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

[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.

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

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

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

Proof. ($\Rightarrow$) Assume $\forall x \in \mathbb{H}.P(x) = 0$. Recall from Eq. (5) that $\psi(X,y) = (\Lambda^b_{\mathbb{H}}(X,y) - M^b(X,y)z(y\omega^m))z(y)$. Since $\deg_X \Lambda_{\mathbb{H}}(X,Y), \deg_X M^b(X,Y) \leq |\mathbb{H}|-1$, then also $\deg_X \psi(X,y) \leq |\mathbb{H}|-1$. Thus, $\sum_{y\in\mathbb{H}} \psi(X,y) = \sum_{y\in\mathbb{H}}\sum_{x\in\mathbb{H}} \psi(x,y)L_x(X) \overset{4}{=} \sum_{x\in\mathbb{H}} P(x)L_x(X) \overset{(*)}{=} 0$, where (*) follows from $\forall x \in \mathbb{H}.P(x) = 0$.
($\Leftarrow$) Let $\sum_{y\in\mathbb{H}} \psi(X,y) = 0$. By Eq. (4), $\forall x \in \mathbb{H}.P(x) = \sum_{y\in\mathbb{H}} \psi(x,y) = 0$. □

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

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

Since $\ell^{\mathbb{H}}_i(Y) = \prod_{j\neq i}(Y-\omega^{j-1})/(\omega^{i-1}-\omega^{j-1})$, $\tilde{z}(Y) \in \mathbb{F}_{\leq n_h-2m_0-3}[Y]$. 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 $(\mathbf{0}^\top_{m_0+1}\|\boldsymbol{z}^\top_{\mathsf{a}}\|\mathbf{0}^\top_{m_0+1}\|\boldsymbol{z}^\top_{\mathsf{b}}\|\boldsymbol{r}^\top_z)^\top$. 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 a polynomial $\tilde{z}(Y)$, such that Eq. (9) holds, guarantees that $z(Y)$ interpolates $(1\|\mathbb{x}^\top\|\boldsymbol{z}^\top_{\mathsf{a}}\|\mathbf{1}^\top_{m_0+1}\|\boldsymbol{z}^\top_{\mathsf{b}}\|\boldsymbol{r}^\top_z)^\top$ for some $\boldsymbol{z}_{\mathbf{a}}$, $\boldsymbol{z}_{\mathbf{b}}$, and $\boldsymbol{r}_z$.

4.2 From Sumcheck to Vampire

According to the preceding discussion, one can handle R1CSLite by proving that $\sum_{y\in\mathbb{H}} \psi(X,y) = 0$. In the current subsection, we construct an argument for the latter. We replace X with a random α chosen by the verifier, obtaining the polynomial $\psi_\alpha(Y) := \psi(\alpha,Y)$. We use Count to show that $\sum_{y\in\mathbb{H}} \psi_\alpha(y) = 0$. For this, as in Section 3, the prover computes the 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 done by KZG-opening all involved polynomials (e.g., $\tilde{z}(Y)$; see Eq. (5)), 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 $v_z \leftarrow \tilde{z}(\beta\omega^m)$ and $v_M \leftarrow M^b(\alpha, \beta)$. This requires us to open a polynomial related to the ILV-opening of $\psi_\alpha(Y)$. We aggregate two KZG-openings by using the technique of [7]. Finally, we use a univariate sumcheck to check the correctness of v_M; this step is complicated, but it follows closely [12,11]. Importantly, we also show that one of the two commitments from the second sumcheck can be considered an aggregated KZG-opening and thus batched with other KZG-openings.

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 $\alpha \leftarrow\!\!\$\ \mathbb{F} \setminus \mathbb{H}$ be sampled by the verifier. (We explain later why $\alpha \notin \mathbb{H}$.) To test that $\sum_{y\in\mathbb{H}} \psi(X, y) = 0$, define $\psi_\alpha(Y) := \psi(\alpha, Y) \in \mathbb{F}_{\leq d}[Y]$.

From Eqs. (5) and (9), we get

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

Clearly, one can set

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

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

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

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

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.,

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

is a zero polynomial.

The prover sends to the verifier KZG-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 $\beta \in C_\beta \subset \mathbb{F}\setminus\mathbb{H}$ is sampled by the verifier. (We will define and motivate C_β later.) More precisely, we verify that $\varphi(\beta) = 0$, where $M^b(\alpha, \beta)$ is substituted by a value v_M computed by the prover. (The latter means that the verifier does not have to compute $M^b(\alpha, \beta)$ itself.) 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 $(X,Y) = (\alpha,\beta)$ itself.)

To improve on efficiency, we *implicitly* KZG-commit to Φ, where

$$\begin{aligned}\Phi(Y) :=& (\Psi(Y)S(Y) - \psi_{\mathsf{ipc}}(Y))/S(Y) = \Psi(Y) - \psi_\alpha(Y) \in \mathbb{F}_{\le d}[Y] \ , \text{ and} \\ \Psi(Y) :=& \left(\Lambda^b_{\mathbb{H}}(\alpha,\beta) - v_M \cdot z(\beta\omega^m)\right)(\mathcal{Z}_{\mathsf{inp}}(\beta)\tilde{z}(Y) + \mathsf{inp}(\beta)) \in \mathbb{F}_{\le n_h - 2m_0 - 3}[Y] \ .\end{aligned} \tag{13}$$

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

We open KZG-commitments to $\tilde{z}(Y)$ at $Y = \beta\omega^m$ (in order to compute $z(\beta\omega^m)$) and $\Phi(Y)$ at $Y = \beta$. For this, the prover sends $v_z \leftarrow \tilde{z}(\beta\omega^m) \in \mathbb{F}$. Since $\Phi(\beta) = 0$, $\Phi(\beta)$ is not transferred. We can open and verify the KZG-commitment to Φ (see Eq. (13)) since we have KZG-commitments to $\tilde{z}$ and ψ_{ipc} (the need for the latter becomes apparent soon), 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

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

Since the prover batches these openings together with one more opening, we will explain the batching process later.

Second (correctness of v_M). We modify a technique from [12,11] by using batching. Recall that M^b satisfies Eq. (7). Moreover, $\deg_X M^b(X,Y)$, $\deg_Y M^b(X,Y) \le n_h - 1$. Thus, $M^b(\alpha,\beta) = \sum_{\kappa\in\mathbb{K}} T(\kappa) \in \mathbb{F}$, where

$$\begin{aligned}\mathsf{num}(Z) :=& \mathcal{Z}_{\mathbb{H}}(\alpha)\mathcal{Z}_{\mathbb{H}}(\beta)/n_h^2 \cdot \mathsf{rcv}(Z) \in \mathbb{F}_{\le n_k - 1}[Z] \ , \\ \mathsf{den}(Z) :=& \alpha\beta - \alpha \cdot \mathsf{col}(Z) - \beta \cdot \mathsf{row}(Z) + \mathsf{rc}(Z) \in \mathbb{F}_{\le n_k - 1}[Z] \ , \\ T(Z) :=& \tfrac{\mathsf{num}(Z)}{\mathsf{den}(Z)} \in \mathbb{F}(Z) \ .\end{aligned} \tag{14}$$

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

We use a sumcheck to check that $v_M = M^b(\alpha,\beta)$. Since this sumcheck is over a low-degree polynomial, we do not need to use $\mathfrak{Count}$'s full power. Let $\widehat{T}(Z) := \sum_{\kappa\in\mathbb{K}} T(\kappa) L^{\mathbb{K}}_\kappa(Z) \in \mathbb{F}_{\le n_k - 1}[Z]$. 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 Fact 1, $\widehat{T}(Z) = ZR(Z) + v_M/n_k$ for some $R \in \mathbb{F}_{\le n_k - 2}[Z]$. 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. Let $Q \in \mathbb{F}_{\le n_k - 3}[Z]$ be such that

$$\mathsf{num}(Z) - R(Z) \cdot \mathsf{zden}(Z) - v_M/n_k \cdot \mathsf{den}(Z) = Q(Z)\mathcal{Z}_{\mathbb{K}}(Z) \ , \text{ where} \tag{15}$$

$$\mathsf{zden}(Z) := \alpha\beta Z - \alpha\mathsf{zcol}(Z) - \beta\mathsf{zrow}(Z) + \mathsf{zrc}(Z) \in \mathbb{F}_{\leq n_k-1}[Z] \ . \tag{16}$$

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

[12,11] now transferred polynomial commitment to $Q(Z)$. We improve on it, by interpreting Eq. (15) as saying that the polynomial $\mathsf{num}(Z) - R(Z) \cdot \mathsf{zden}(Z) - v_M/n_k \cdot \mathsf{den}(Z)$ opens to 0 at all points $Z \in \mathbb{K}$. Thus, $Q(Z)$ is an aggregated polynomial *opening* of the left-hand side of Eq. (15) at all points of $\mathbb{K}$. Importantly, we can aggregate this opening with the openings $\tilde{z}_{\mathsf{pc}}(Z)$ and $\Phi_{\mathsf{pc}}(Z)$ from before. Hence, we can save an additional one group element.[8] (We will explain batching in a few paragraphs.)

Thus, the prover only commits to R. When we add to $\mathsf{srs}_{\mathcal{R}}$ elements like $[\mathsf{rcv}(\sigma), \mathsf{col}(\sigma)]_2$, the verifier can compute the $\mathbb{G}_2$ elements 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 use $[R(\sigma)\tau, Q(\sigma)\tau]_1$ 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 $a \in \mathrm{span}(\sigma^i\tau)_{i\in\mathcal{S}'}$.

Batching. The prover batches the openings of $\tilde{z}(Y)$ at $Y = \beta\omega^m$, $\Phi(Y)$ at $Y = \beta$, and the left-hand side of Eq. (15) at all points $Y \in \mathbb{K}$ as $[B_{\mathsf{pc}}(\sigma,\tau)]_1 \leftarrow [\tilde{z}_{\mathsf{pc}}(\sigma) + \Phi_{\mathsf{pc}}(\sigma) + Q(\sigma)\tau]_1$. Notably, since the three polynomial openings are at different locations (β, $\beta\omega^m$, and all points of $\mathbb{K}$, correspondingly), one does not have to randomize this check. (See Section 5.1 for formal proof.) The latter is a general fact, not mentioned in [7] and is thus an independent contribution.

Following the aggregation of [7], the verifier must check that $[\tilde{z}(\sigma) - v_z]_1 \bullet [(\sigma-\beta)\mathcal{Z}_{\mathbb{K}}(\sigma)]_2 + [\Phi(\sigma)]_1 \bullet [(\sigma-\beta\omega^m)\mathcal{Z}_{\mathbb{K}}(\sigma)]_2 + [\mathsf{num}(\sigma) - R(\sigma)\mathsf{zden}(\sigma) - v_M/n_k \cdot \mathsf{den}(\sigma)]_1 \bullet [(\sigma-\beta)(\sigma-\beta\omega^m)]_2 = [B_{\mathsf{pc}}(\sigma)]_1 \bullet [(\sigma-\beta)(\sigma-\beta\omega^m)\mathcal{Z}_{\mathbb{K}}(\sigma)]_2$, 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 $S(\sigma)$. We also modify the last addend on the left-hand side to allow the prover and the verifier to compute it given the terms given in the SRS. Finally, we use the trapdoor τ because we need to do a low-degree test.

As part of $[B_{\mathsf{pc}}(\sigma)]_1$, 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).

The batch opening reduces the communication by two group elements.

[8] The first version of $\mathfrak{Vampire}$ ($\mathfrak{OldVampire}$), [27], differs from the current $\mathfrak{Vampire}$ exactly at this point: in $\mathfrak{OldVampire}$, we also transferred a polynomial commitment to $Q(Z)$. This resulted in an argument that is longer by one group element but in a shorter SRS.

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. For the sake of completeness, Fig. 2 defines all used polynomials. Since this argument is public-coin and has a constant number of rounds, we can apply the Fiat-Shamir heuristic ([13], we omit the details) to obtain the zk-SNARK $\mathfrak{Vampire}$.

We sample the challenge β from the set

$$C_\beta = \left\{ \beta \in \mathbb{F} \middle| \begin{array}{c} \beta \notin (\mathbb{H} \cup \mathbb{K} \cup \{0, \sigma, \sigma/\omega^m\}) \wedge \\ S(\beta) \neq 0 \ \wedge \ S(\beta\omega^m) \neq 0 \ \wedge \ \beta\omega^m \notin \mathbb{K} \end{array} \right\} .$$

We need $\beta \notin \{\sigma, \sigma/\omega^m\}$ to get perfect zero-knowledge (see the proof of Theorem 2). 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 $S(Y)$, $\mathcal{Z}_{\mathbb{K}}(Y)$, $Y-\beta$, and $Y-\beta\omega^m$ are coprime. Hence, we need that (1) $S(\beta) \neq 0$, $S(\beta\omega^m) \neq 0$, $\mathcal{Z}_{\mathbb{K}}(\beta) \neq 0$, $\mathcal{Z}_{\mathbb{K}}(\beta\omega^m) = 0$ (the latter two conditions hold iff $\beta \notin \mathbb{K}$ and $\beta\omega^m \notin \mathbb{K}$) for coprimeness with $Y - \beta$ and $Y - \beta\omega^m$, and (2) $\beta \neq 0$ (otherwise $\beta = \beta\omega^m$, and thus $Y - \beta$ and $Y - \beta\omega^m$ cannot be coprime). 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. Thus, in practice, β can be sampled from $\mathbb{F}$, resulting in only a negligible security risk.

Since $\mathcal{S}_1(X, X_\tau)$ and $\mathcal{S}_2(X)$ consist of monomials and one can verify the correctness of its SRS efficiently, $\mathfrak{Vampire}$ is updatable. We will prove the latter in Theorem 3. See Appendix A for a thorough efficiency analysis.

5 Security Proofs

We first provide 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.* [3]) *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 $\mathcal{P}(Z) := \mathcal{V}(S_1 Z + R_1, \ldots, S_c Z + R_c)$. 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.*

The following lemma (that is based on [22,7]) allows us to batch-open several polynomials in distinct locations. [7] presented a more general version where the locations do not have to be distinct; the cost of it is a randomized verification that involves a value $\gamma \leftarrow\!\!\$\ \mathbb{F}$ sampled by the verifier. On the other hand, [7] did not involve the polynomial $S(Y)$ and worked only with univariate polynomials.

Lemma 5 (Aggregation lemma). *Let $f_i \in \mathbb{F}[Y, X_\tau]$, $T_i \subset \mathbb{F}$ be mutually disjoint sets, and let $T := \cup_i T_i$. Let $S(Y) \in \mathbb{F}[Y]$ be such that $\forall s \in T. S(s) \neq 0$.*

$\mathsf{Pgen}(1^\lambda)$: generate p as usually, assuming that $n_h, n_k \mid (p-1)$ and $3 \nmid n_k$

$\mathsf{KGen}(\mathsf{p}, n_h, n_k)$: $\mathcal{S}_1(X, X_\tau) = \{(X^i)_{i=0:i\neq d_{\mathsf{gap}}}^{d_{\mathsf{gap}}+d}, (X^i X_\tau)_{i=0}^{n_k-2}\}$;
$\mathcal{S}_2(X) = \{1, X, X^2, X^{n_k}, X^{n_k+1}, (X^{d_{\mathsf{gap}}-jn_h+i})_{j\in\{0,1,2\}, i\in[0,n_k+2]}\}$;
$\sigma, \tau \leftarrow\!\!\$\ \mathbb{F}^*$; $\mathsf{td} \leftarrow (\sigma, \tau)$; $\mathsf{srs} \leftarrow (\mathsf{p}, n_h, n_k, [g(\sigma,\tau) : g \in \mathcal{S}_1(X, X_\tau)]_1, [g(\sigma) : g \in \mathcal{S}_2(X)]_2)$;

$\mathsf{Derive}(\mathsf{srs}, \mathcal{I})$: $\mathsf{ek}_\mathcal{R} \leftarrow (\mathsf{p}, \mathcal{I}, [g(\sigma,\tau) : g \in \mathcal{S}_1(X, X_\tau)]_1)$;

$$\mathsf{srs}_\mathcal{R} \leftarrow \begin{pmatrix} \left[(\sigma^i S(\sigma))_{i=0}^3, \mathsf{rcv}(\sigma)S(\sigma), \mathsf{col}(\sigma)S(\sigma), \mathsf{row}(\sigma)S(\sigma), \mathsf{rc}(\sigma)S(\sigma), \mathcal{Z}_\mathbb{K}(\sigma), \sigma\mathcal{Z}_\mathbb{K}(\sigma)\right]_2, \\ \left[(\sigma^i \mathsf{zcol}(\sigma)S(\sigma), \sigma^i \mathsf{zrow}(\sigma)S(\sigma), \sigma^i \mathsf{zrc}(\sigma)S(\sigma), \sigma^i \mathcal{Z}_\mathbb{K}(\sigma)S(\sigma))_{i=0}^2\right]_2 \end{pmatrix};$$

$\mathsf{vk}_\mathcal{R} \leftarrow (\mathsf{p}, \mathcal{I}, [1, \tau, \sigma\tau, \sigma^2\tau]_1, [1]_2, \mathsf{srs}_\mathcal{R})$;

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

$\mathcal{Z}_{\mathsf{inp}}(Y) \leftarrow \prod_{i=1}^{m_0+1}(Y - \omega^{i-1})(Y - \omega^{m+i-1}) \in \mathbb{F}_{\le 2m_0+2}[Y]$;
$\mathsf{inp}(Y) \leftarrow \ell_1^\mathbb{H}(Y) + \sum_{i=1}^{m_0} \mathbb{x}_i \ell_{i+1}^\mathbb{H}(Y) + \sum_{i=1}^{m_0+1} \ell_{m+i}^\mathbb{H}(Y) \in \mathbb{F}_{\le n_h-1}[Y]$;
$\boldsymbol{r}_z \leftarrow\!\!\$\ \mathbb{F}^b$; $\tilde{z}(Y) \leftarrow \sum_{i=1}^{m-m_0-1} \boldsymbol{z}_\mathsf{a}[i] \frac{\ell_{m_0+1+i}^\mathbb{H}(Y)}{\mathcal{Z}_{\mathsf{inp}}(Y)} + \sum_{i=1}^{m-m_0-1} \boldsymbol{z}_\mathsf{b}[i] \frac{\ell_{m+m_0+1+i}^\mathbb{H}(Y)}{\mathcal{Z}_{\mathsf{inp}}(Y)} + \sum_{i=1}^{b} \boldsymbol{r}_z[i] \frac{\ell_{2m+i}^\mathbb{H}(Y)}{\mathcal{Z}_{\mathsf{inp}}(Y)}$;
$z(Y) \leftarrow \mathcal{Z}_{\mathsf{inp}}(Y)\tilde{z}(Y) + \mathsf{inp}(Y)$; // $\tilde{z}(Y) \in \mathbb{F}_{\le n_h-2m_0-3}[Y]$; $z(Y) \in \mathbb{F}_{\le n_h-1}[Y]$

$$\xrightarrow{[\tilde{z}(\sigma)]_1} \qquad \alpha \leftarrow\!\!\$\ \mathbb{F} \setminus \mathbb{H}$$
$$\xleftarrow{\alpha}$$

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

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

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

$$\xrightarrow{[\psi_{\mathsf{ipc}}(\sigma)]_1} \qquad \beta \leftarrow\!\!\$\ C_\beta$$
$$\xleftarrow{\beta}$$

..... Low-degree 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 $R \in \mathbb{F}_{\le n_k-2}[Z], Q \in \mathbb{F}_{\le n_k-3}[Z]$, such that
$\mathsf{num}(Z) - R(Z)\mathsf{zden}(Z) - v_M/n_k \cdot \mathsf{den}(Z) = Q(Z)\mathcal{Z}_\mathbb{K}(Z)$;
$z(\beta\omega^m) \leftarrow \mathcal{Z}_{\mathsf{inp}}(\beta\omega^m)v_z + \mathsf{inp}(\beta\omega^m)$;
$\Psi(Y) \leftarrow \left(\Lambda_\mathbb{H}^b(\alpha, \beta) - v_M \cdot z(\beta\omega^m)\right)(\mathcal{Z}_{\mathsf{inp}}(\beta)\tilde{z}(Y) + \mathsf{inp}(\beta)) \in \mathbb{F}_{\le n_h-m_0-3}[Y]$;
$\tilde{z}_{\mathsf{pc}}(Y) \leftarrow (\tilde{z}(Y) - v_z)/(Y - \beta\omega^m) \in \mathbb{F}_{\le n_h-m_0-4}[Y]$;
$\Phi_{\mathsf{pc}}(Y) \leftarrow (\Psi(Y) - \psi_\alpha(Y))/(Y - \beta) \in \mathbb{F}_{\le d-1}[Y]$;
$B_{\mathsf{pc}}(Y, X_\tau) \leftarrow \tilde{z}_{\mathsf{pc}}(Y) + \Phi_{\mathsf{pc}}(Y) + Q(Y)X_\tau \in \mathbb{F}_{\le d-1}[Y] \cup (\mathbb{F}_{\le n_k-3}[Y])[X_\tau]$;

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

$[\zeta_1(\sigma)]_2 \leftarrow \mathcal{Z}_\mathbb{H}(\alpha)\mathcal{Z}_\mathbb{H}(\beta)/n_h^2 \cdot [\mathsf{rcv}(\sigma)S(\sigma)]_2$; // $= [\mathsf{num}(\sigma)S(\sigma)]_2$
$[\zeta_2(\sigma)]_2 \leftarrow \alpha\beta[S(\sigma)]_2 - \alpha[\mathsf{col}(\sigma)S(\sigma)]_2 - \beta[\mathsf{row}(\sigma)S(\sigma)]_2 + [\mathsf{rc}(\sigma)S(\sigma)]_2$; // $= [\mathsf{den}(\sigma)S(\sigma)]_2$
$[\zeta_3(\sigma)]_2 \leftarrow \alpha\beta[(\sigma-\beta)(\sigma-\beta\omega^m)\sigma S(\sigma)]_2 - \alpha[(\sigma-\beta)(\sigma-\beta\omega^m)\mathsf{zcol}(\sigma)S(\sigma)]_2$
$-\beta[(\sigma-\beta)(\sigma-\beta\omega^m)\mathsf{zrow}(\sigma)S(\sigma)]_2 + [(\sigma-\beta)(\sigma-\beta\omega^m)\mathsf{zrc}(\sigma)S(\sigma)]_2$;
// $= [(\sigma-\beta)(\sigma-\beta\omega^m)\mathsf{zden}(\sigma)S(\sigma)]_2$
$\mathsf{inp}(\beta\omega^m) \leftarrow \ell_1^\mathbb{H}(\beta\omega^m) + \sum_{i=1}^{m_0} \mathbb{x}_i \ell_{i+1}^\mathbb{H}(\beta\omega^m) + \sum_{i=1}^{m_0+1} \ell_{m+i}^\mathbb{H}(\beta\omega^m)$;
$\mathcal{Z}_{\mathsf{inp}}(\beta\omega^m) \leftarrow \prod_{i=1}^{m_0+1}(\beta\omega^m - \omega^{i-1})(\beta\omega^m - \omega^{m+i-1})$; $z(\beta\omega^m) \leftarrow \mathcal{Z}_{\mathsf{inp}}(\beta\omega^m)v_z + \mathsf{inp}(\beta\omega^m)$;
$[\Psi(\sigma)]_1 \leftarrow \left(\Lambda_\mathbb{H}^b(\alpha, \beta) - v_M \cdot z(\beta\omega^m)\right)(\mathcal{Z}_{\mathsf{inp}}(\beta)[\tilde{z}(\sigma)]_1 + \mathsf{inp}(\beta)[1]_1)$;
(♯♯) Check $([\tilde{z}(\sigma) - v_z]_1 \bullet [(\sigma-\beta)\mathcal{Z}_\mathbb{K}(\sigma)S(\sigma)]_2)$
$+([\Psi(\sigma)]_1 \bullet [(\sigma-\beta\omega^m)\mathcal{Z}_\mathbb{K}(\sigma)S(\sigma)]_2 - [\psi_{\mathsf{ipc}}(\sigma)]_1 \bullet [(\sigma-\beta\omega^m)\mathcal{Z}_\mathbb{K}(\sigma)]_2)$
$+\left([(\sigma-\beta)(\sigma-\beta\omega^m)\tau]_1 \bullet [\zeta_1(\sigma) - v_M/n_k \cdot \zeta_2(\sigma)]_2 - [R(\sigma)\tau]_1 \bullet [\zeta_3(\sigma)]_2\right)$
$=^?\ [B_{\mathsf{pc}}(\sigma,\tau)]_1 \bullet [(\sigma-\beta)(\sigma-\beta\omega^m)\mathcal{Z}_\mathbb{K}(\sigma)S(\sigma)]_2$;

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

Fix $v_s \in \mathbb{F}$ for all $s \in T$. Let $\hat{v}_i \in \mathbb{F}[Y]$ be a polynomial, such that $\hat{v}_i(s) = v_s$ for all $s \in T_i$. Let $b_i \in \{0,1\}$. If there exists a polynomial $B_{\mathsf{pc}} \in \mathbb{F}[Y, X_\tau]$, such that

$$\textstyle\sum_i (f_i(Y, X_\tau) - \hat{v}_i(Y)) \mathcal{Z}_{T \setminus T_i}(Y) S(Y)^{b_i} = B_{\mathsf{pc}}(Y, X_\tau) \mathcal{Z}_T(Y) S(Y) \ , \tag{17}$$

then $\forall i. \forall s \in T_i. f_i(s, X_\tau) = v_s$.

Proof. Since T_i are disjoint and the roots of $S(Y)$ are not in T, Eq. (17) implies that $\forall i. (\mathcal{Z}_{T_i}(Y) \mid (f_i(Y, X_\tau) - \hat{v}_i(Y)))$. The lemma follows. □

In our use, $\hat{v}_i(Y)$ is either constant or the unique monic polynomial (e.g, Lagrange's polynomial) of degree $|T_i| - 1$, such that $\hat{v}_i(s) = v_s$ for all $s \in T_i$.

5.1 Knowledge-Soundness Proof

We start by proving two lemmas about coprimeness of some of the polynomials used in $\mathfrak{Vampire}$. We need them later in the knowledge soundness proof.

Lemma 6. *Recall that $\mathcal{Z}_{\mathbb{K}}(Y) = \prod_{\kappa \in \mathbb{K}} (Y - \kappa)$ and $S(Y) = Y^{d_{\mathsf{gap}}} + Y^{d_{\mathsf{gap}} - n_h} + Y^{d_{\mathsf{gap}} - 2n_h}$. If $\mathrm{char}(\mathbb{F}) \neq 3$, $n_h \geq 3$, and $3 \nmid n_k$, then $\gcd(S(Y), \mathcal{Z}_{\mathbb{K}}(Y)) = 1$.*

Proof. Clearly, $\gcd(S(Y), \mathcal{Z}_{\mathbb{K}}(Y)) = 1$ iff $\mathbb{K}$ does not contain roots of $S(Y)$. Since $S(Y) = Y^{3n_h - 3} + Y^{2n_h - 3} + Y^{n_h - 3} = Y^{n_h - 3}(Y^{2n_h} + Y^{n_h} + 1)$, roots of $S(Y)$ are 0 (when $n_h > 3$), which is not in $\mathbb{K}$, and roots of $S^*(Y) := Y^{2n_h} + Y^{n_h} + 1$.

Consider the polynomial $P(X) = X^2 + X + 1$ where Y^{n_h} from $S^*(Y)$ is substituted by X. Let a be a root of $P(X)$. Since $a^2 = -a - 1$, we have $a^3 = -a^2 - a = a + 1 - a = 1$. Thus, the order of a divides three. The order cannot be one since then $a^1 = a = 1$, but $P(1) = 1 + 1 + 1 \neq 0$ when $\mathrm{char}(\mathbb{F}) \neq 3$. Thus, the order of a is three. If $a \in \mathbb{K}$, then by Lagrange's theorem, $3 \mid n_k$, violating the assumption $3 \nmid n_k$. Thus, $a \notin \mathbb{K}$. Finally, suppose that b is a root of $Y^{2n_h} + Y^{n_h} + 1$. If $b \in \mathbb{K}$, then $b^{n_h} \in \mathbb{K}$ and $P(b^{n_h}) = 0$. We already showed that $P(X)$ does not have roots in $\mathbb{K}$ and thus, $S(Y)$ does not have roots in $\mathbb{K}$. □

Lemma 7. *If $\mathrm{char}(\mathbb{F}) \neq 3$, $n_h \geq 3$, $3 \nmid n_k$, and $\beta \in C_\beta$, then $S(Y)$, $\mathcal{Z}_{\mathbb{K}}(Y)$, $Y - \beta$, and $Y - \beta\omega^m$ are pair-wise coprime.*

Proof. Let us look at all the pairs one-by-one.

1. We proved in Lemma 6 that $S(Y)$ and $\mathcal{Z}_{\mathbb{K}}(Y)$ are coprime assuming $\mathrm{char}(\mathbb{F}) \neq 3$, $n_h \geq 3$, and $3 \nmid n_k$.
2. Suppose that $\beta = \beta\omega^m$. Then, $\beta(\omega^m - 1) = 0$ and thus either $\beta = 0$ or $\omega^m = 1$. However, $0 \notin C_\beta$. Moreover, $\omega^m \neq 1$ since $m < n_h$. Thus, $Y - \beta$ and $Y - \beta\omega^m$ are coprime.
3. $\mathcal{Z}_{\mathbb{K}}(Y)$ is coprime with $Y - \beta$ and $Y - \beta\omega^m$ since β and $\beta\omega^m$ are not roots of $\mathcal{Z}_{\mathbb{K}}(Y)$ by the definition of C_β.
4. For the same reason, $S(Y)$ is coprime with $Y - \beta$ and $Y - \beta\omega^m$. □

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

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

Theorem 1. *Assume that* $\mathrm{char}(\mathbb{F}) \neq 3$, $n_h \geq 3$ *and* $3 \nmid n_k$. *Then,* $\mathfrak{Vampire}$ *is knowledge-sound in the AGM under the PDL assumption. More precisely, an algebraic* $\mathcal{A}$ *breaks the knowledge-soundness of* $\mathfrak{Vampire}$ *with probability at most*

$$\mathsf{Adv}^{\mathrm{pdl}}_{d_1,d_2,\mathsf{Pgen},\mathcal{B}}(\lambda) \cdot \tfrac{|\mathbb{F}|^2}{|\mathbb{F}|^2-q} + \tfrac{16n_h+4m_0-12}{|C_\beta|} + \tfrac{n_h-1}{|\mathbb{F}|-n_h} \quad , \tag{18}$$

where $\mathcal{B}$ *is some PPT adversary,* $d_1 = \max(d_{\mathsf{gap}} + d, n_k - 1)$, $d_2 = n_k + d_{\mathsf{gap}} + 2$, *and* $q \leq 2 + n_k + d_{\mathsf{gap}} + d_{\max}$ *such that* $d_{\max} = \max(d_{\mathsf{gap}} + d, n_k - 1)$.

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

In Fig. 3, we depict the knowledge extractor Ext. Ext runs $\mathsf{Ext}_{\mathcal{A}}$[9] to obtain coefficients of $\tilde{z}(Y, X_\tau)$. Ext then evaluates $\tilde{z}(Y, 0) \cdot \mathcal{Z}_{\mathsf{inp}}(Y)$ at points of $Y \in \mathbb{H}$, corresponding to $\boldsymbol{z}_{\mathsf{a}}$ and $\boldsymbol{z}_{\mathsf{b}}$ in the honest argument. Ext then returns those vectors. In the rest of this proof, we show that the value outputted by Ext is a valid witness for $\mathbb{x}$ with an overwhelming probability.

We have one verification check that guarantees $\mathcal{V}(\sigma, \tau) = 0$, where

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

where $\Psi(Y, X_\tau) = \left(\Lambda^b_{\mathbb{H}}(\alpha, \beta) - v_M \cdot z(\beta\omega^m)\right)\left(\mathcal{Z}_{\mathsf{inp}}(\beta)\tilde{z}(Y, X_\tau) + \mathsf{inp}(\beta)\right)$ for $z(\beta\omega^m) = \mathcal{Z}_{\mathsf{inp}}(\beta\omega^m)v_z + \mathsf{inp}(\beta\omega^m)$. ($\mathcal{V}(\sigma, \tau) = 0$ follows from (♯♯) in Fig. 2 when one allows polynomials like $\tilde{z}(Y, X_\tau)$ to be maliciously chosen.)

Clearly, $\Pr[\mathcal{A} \text{ wins}] \leq \Pr[\mathcal{A} \text{ wins} \mid \mathcal{V}(Y, X_\tau) = 0] + \Pr[\mathcal{A} \text{ wins} \mid \mathcal{V}(Y, X_\tau) \neq 0]$. Below, we will analyze both conditional probabilities.

[9] Even though $\mathcal{A}$ is interactive, since we extract only from the first round message of $\mathcal{A}$, the knowledge soundness extractor is still non-interactive.

Lemma 8. *Assume* $\mathrm{char}(\mathbb{F}) \neq 3$, $n_h \geq 3$, *and* $3 \nmid n_k$. *For an algebraic* $\mathcal{A}$, $\Pr[\mathcal{A} \text{ wins} \mid \mathcal{V}(Y, X_\tau) = 0] \leq (16n_h + 4m_0 - 12)/|C_\beta| + (n_h - 1)/(|\mathbb{F}| - n_h)$.

Proof. Assume $\mathcal{V}(Y, X_\tau) = 0$. Recall that by Lemma 7, $S(Y)$, $\mathcal{Z}_{\mathbb{K}}(Y)$, $Y - \beta$, and $Y - \beta\omega^m$ are pair-wise coprime. Hence, we can use Lemma 5 with $f_1(Y) = \tilde{z}(Y, X_\tau)S(Y)$, $f_2(Y) = \Psi(Y, X_\tau)S(Y) - \psi_{\mathsf{ipc}}(Y, X_\tau)$, $f_3(Y) = (\mathsf{num}(Y) - v_M/n_k \cdot \mathsf{den}(Y))X_\tau - R(Y, X_\tau)\mathsf{zden}(Y)$, $T_1 = \{\beta\omega^m\}$, $T_2 = \{\beta\}$, $T_3 = \mathbb{K}$, $v_{\beta\omega^m} = v_z$, $v_\beta = 0$, and $v_y = 0$ for $y \in \mathbb{K}$. It follows from $\mathcal{V} = 0$ and Lemma 5 that

$$\tilde{z}(\beta\omega^m, X_\tau) = v_z \ , \tag{19}$$

$$\Psi(\beta, X_\tau)S(\beta) - \psi_{\mathsf{ipc}}(\beta, X_\tau) = 0 \ , \tag{20}$$

$$\forall y \in \mathbb{K}.\big(\mathsf{num}(y) - \tfrac{v_M}{n_k}\mathsf{den}(y)\big)X_\tau - R(y, X_\tau)\mathsf{zden}(y) = 0 \ . \tag{21}$$

We analyze each of the three equations separately.

Equation (19). Denote $\tilde{z}(Y, X_\tau) = \tilde{z}'(Y)X_\tau + \tilde{z}''(Y)$. It follows from Eq. (19) that $\tilde{z}'(\beta\omega^m)X_\tau + \tilde{z}''(\beta\omega^m) = v_z$. Thus, $\tilde{z}''(\beta\omega^m) = v_z$ and $\tilde{z}'(\beta\omega^m) = 0$.

Equation (21). Write $R(Y, X_\tau) = R'(Y)X_\tau + R''(Y)$ and $Q(Y, X_\tau) = Q'(Y)X_\tau + Q''(Y)$. In particular, $\deg R'(Y) \leq n_k - 2$ since the only X_τ-dependent monomials in $\mathcal{S}_1(Y)$ are $(Y^i X_\tau)_{i=0}^{n_k-2}$. Thus, from Eq. (21),

$$\forall y \in \mathbb{K}.\big(\mathsf{num}(y) - \tfrac{v_M}{n_k}\mathsf{den}(y) - R'(Y)\mathsf{zden}(y)\big)X_\tau - R''(y)\mathsf{zden}(y) = 0.$$

Hence, $\forall y \in \mathbb{K}.\mathsf{num}(y) - v_M/n_k \cdot \mathsf{den}(y) - yR'(y)\mathsf{den}(y) = 0$, that is, $\forall y \in \mathbb{K}.T(y) := \mathsf{num}(y)/\mathsf{den}(y) = v_M/n_k + yR'(y)$. 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 Fact 1,

$$M^b(\alpha, \beta) = \textstyle\sum_{y\in\mathbb{K}} T(y) = \sum_{y\in\mathbb{K}} \widehat{T}(y) = v_M \ . \tag{22}$$

Equation (20). 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 express $\Psi(Y, X_\tau)$ as

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

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

Thus, Eq. (20) implies $(\Psi'(\beta)S(\beta) - \psi'_{\mathsf{ipc}}(\beta))X_\tau + \Psi''(\beta)S(\beta) - \psi''_{\mathsf{ipc}}(\beta) = 0$. Hence, $\Psi''(\beta)S(\beta) = \psi''_{\mathsf{ipc}}(\beta)$.

Denote $\psi(Y) := \big(\Lambda^b_{\mathbb{H}}(\alpha, Y) - M^b(\alpha, Y) \cdot z(Y\omega^m)\big) \cdot (\mathcal{Z}_{\mathsf{inp}}(Y)\tilde{z}''(Y) + \mathsf{inp}(Y))$. Let $\mathcal{V}_3(Y) := \psi(Y)S(Y) - \psi''_{\mathsf{ipc}}(Y)$. By Eq. (22), $\psi(\beta) = \Psi''(\beta)$ and thus $\mathcal{V}_3(\beta) = 0$. Since $\psi(Y)$ and $\psi_{\mathsf{ipc}}(Y)$ were fixed before the adversary received β, we can apply the Schwartz-Zippel lemma to $\mathcal{V}_3$. Recall that (1) $\deg \tilde{z}'' \leq d_{\mathsf{gap}} + d$, (2) $\deg \mathsf{inp} \leq n_h - 1$, (3) $\deg \mathcal{Z}_{\mathsf{inp}} \leq 2(m_0 + 1)$, (4) $\deg z \leq d_{\mathsf{gap}} + d + 2(m_0 + 1)$, (5) $\deg_Y \Lambda_{\mathbb{H}}(\alpha, Y) \leq n_h - 1$, $\deg_Y M^b(\alpha, Y) \leq n_h - 1$, (6) $\deg \psi''_{\mathsf{ipc}} \leq d_{\mathsf{gap}} +$

d, (7) $\deg\psi \le (n_h-1)+2\,(d_{\mathsf{gap}}+d+2(m_0+1)) = 13n_h+4m_0-9$. Thus, $\deg\mathcal{V}_3 \le \max(\deg\psi+d_{\mathsf{gap}}, \deg\psi''_{\mathsf{ipc}}) \le \max(16n_h+4m_0-12, 6(n_h-1)) = 16n_h+4m_0-12$. If $\mathcal{V}_3(Y)\neq 0$, then the verifier's acceptance implies that $\mathcal{V}_3(\beta)=0$, which according to Schwartz-Zippel lemma can only happen with probability $(16n_h+4m_0-12)/|C_\beta|$.

Let us consider the case $\mathcal{V}_3(Y)=0$. Since $\psi(Y)S(Y)=\psi''_{\mathsf{ipc}}(Y)$, $\deg\psi''_{\mathsf{ipc}} \le d_{\mathsf{gap}}+d$ and $\deg S = d_{\mathsf{gap}}$, then $\deg\psi(Y)\le d$. Since $Y^{d_{\mathsf{gap}}}\notin\mathcal{S}_1(Y,X_\tau)$, the coefficient of $Y^{d_{\mathsf{gap}}}$ in $\psi(Y)S(Y)=\psi''_{\mathsf{ipc}}(Y)=\sum_{i=0}^{d_{\mathsf{gap}}+d}(\psi''_{\mathsf{ipc}})_iY^i$ is 0. But this coefficient is $\psi_0+\psi_{n_h}+\psi_{2n_h}=0$. Thus, from Lemma 1, it follows that $\sum_{y\in\mathbb{H}}\psi(y)=0$.

Let us express $\psi(Y)$ as $\psi(X,Y)$, where X corresponds to α. We established that $\sum_{y\in\mathbb{H}}\psi(\alpha,y)=0$. For any $y\in\mathbb{H}$, $\deg\psi(X,y)=n_h-1$. 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. (4). In the beginning of Section 4, we established that this equation is equivalent to R1CSLite. Since $z(Y)=\mathcal{Z}_{\mathsf{inp}}(Y)\tilde{z}''(Y)+\mathsf{inp}(Y)=\tilde{z}''(Y)\prod_{i=1}^{m_0+1}(Y-\omega^{i-1})(Y-\omega^{m+i-1})+\ell_1^{\mathbb{H}}(Y)+\sum_{i=1}^{m_0}\mathbb{x}_i\ell_{i+1}^{\mathbb{H}}(Y)+\sum_{i=1}^{m_0+1}\ell_{m+i}^{\mathbb{H}}(Y)$, then $z(\omega^{i-1})$ for $i\in\{1,\ldots,m_0+1\}$ correctly encodes $(1,\mathbb{x}_1,\ldots,\mathbb{x}_{m_0})$. 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 9. *Let* $d_1:=\max(d_{\mathsf{gap}}+d, n_k-1)$, $d_2:=n_k+d_{\mathsf{gap}}+2$, *and* $q\le 2+n_k+d_{\mathsf{gap}}+d_{\max}$ *for* $d_{\max}:=\max(d_{\mathsf{gap}}+d,n_k-1)$. *For an algebraic* $\mathcal{A}$ *and* $\mathcal{V}(Y,X_\tau)$ *as above, there exists a PPT* $\mathcal{B}$, *such that* $\Pr[\mathcal{A}\ \textit{wins}\mid\mathcal{V}(Y,X_\tau)\neq 0]\le \mathsf{Adv}^{\mathrm{pdl}}_{d_1,d_2,\mathsf{Pgen},\mathcal{B}}(\lambda)\cdot|\mathbb{F}|^2/(|\mathbb{F}|^2-q)$.

Proof. This part of the proof is standard and similar to [15]'s AGM proof for Groth16 SNARK. Thus, 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 $\mathcal{V}(Y)\neq 0$.

$\mathcal{B}$ gets as an input $(\mathsf{p};[(x^i)_{i=0}^{d_1}]_1,[(x^i)_{i=0}^{d_2}]_2)$. $\mathcal{B}$ samples s_1,s_2,r_1,r_2 and defines $\sigma=s_1x+r_1$ and $\tau=s_2x+r_2$. 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 $[\sigma^i]_\iota$ and $[\sigma^i\tau]_\iota$ (e.g., $[\sigma]_1=s_1[x]_1+r_1[1]_1$). The degrees d_1 and d_2 are sufficiently high 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. $\mathcal{B}$ now knows coefficients of verification polynomial $\mathcal{V}(Y,X_\tau)$.

When $\mathcal{A}$ wins, $\mathcal{V}(\sigma,\tau)=0$. We define $\mathcal{P}(X):=\mathcal{V}(S_1X+R_1,S_2X+R_2)\in(\mathbb{F}[S_1,S_2,R_1,R_2])[X]$. According to Fact 4, if $\mathcal{V}(Y,X_\tau)\neq 0$ has degree q, then the coefficient of the maximal degree of $\mathcal{P}(X)$ is some polynomial $C(S_1,S_2)\in\mathbb{F}[S_1,S_2]$ of degree q. Thus, the coefficient of the leading term of $\mathcal{P}'(X):=\mathcal{V}(s_1X+r_1,s_2X+r_2)\in\mathbb{F}[X]$ 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 $q/|\mathbb{F}|^2$. Thus, with an overwhelming probability, $C(s_1,s_2)\neq 0$ and $\mathcal{P}'(X)\neq 0$. Thus, $\mathcal{B}$ can find the roots of $\mathcal{P}'(X)$. One of the roots must be σ since $\mathcal{P}'(\sigma)=\mathcal{V}(s_1\sigma+r_1,s_2\sigma+r_2)=\mathcal{V}(\sigma,\tau)=0$. Finally, $\mathcal{B}$ outputs σ.

The total degree q of $\mathcal{V}$ is $\leq 2 + d_{\mathsf{gap}} + n_k + d_{\max}$, where $d_{\max} := \max(d_{\mathsf{gap}} + d, n_k - 2)$. Thus, $\Pr[\mathcal{A} \text{ wins} \mid \mathcal{V}(Y) \neq 0](1 - q/|\mathbb{F}|^2) \leq \mathsf{Adv}^{\mathrm{pdl}}_{d_1,d_2,\mathsf{Pgen},\mathcal{B}}(\lambda)$. Hence, $\Pr[\mathcal{A} \text{ wins} \mid \mathcal{V}(Y) \neq 0] \leq \mathsf{Adv}^{\mathrm{pdl}}_{d_1,d_2,\mathsf{Pgen},\mathcal{B}}(\lambda) \cdot |\mathbb{F}|^2/(|\mathbb{F}|^2 - q)$. □

It follows from these lemmas that Eq. (18) 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 $\boldsymbol{z}_{\mathsf{a}} = \boldsymbol{z}_{\mathsf{b}} = \mathbf{0}$. We argue that no adversary can distinguish an argument with the all-zero witness from an argument with the 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 $\mathcal{R} \in \mathcal{UR}_{\mathsf{p},n_h,n_k}$. 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 $\boldsymbol{z}_{\mathsf{a}}, \boldsymbol{z}_{\mathsf{b}}$ as in the real protocol. Then, Sim 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 the 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 $\beta \notin C_\beta$.

After that, Sim follows the protocol. In particular, Sim computes $[R(\sigma)\tau]_1$ and $[B_{\mathsf{pc}}(\sigma, \tau)]_1$ honestly. The latter is computed by setting $B_{\mathsf{pc}}(\sigma, \tau) \leftarrow (\tilde{z}(\sigma) - v_z)/(\sigma - \beta\omega^m) + (\Psi(\sigma) - \psi_\alpha(\sigma))/(\sigma - \beta) + Q(\sigma)\tau$; this is possible since $\beta \notin \{\sigma, \sigma/\omega^m\}$ and $\sigma \notin \mathbb{K}$. (Since v_z can be maliciously chosen, $(\tilde{z}(\sigma) - v_z)/(\sigma - \beta\omega^m)$ does not have to be polynomial.) Hence, if there are no aborts, the argument transcript is $[\tilde{z}(\sigma)]_1, \alpha, [\psi_{\mathsf{ipc}}(\sigma)]_1, \beta, v_z, v_M, [R(\sigma)\tau]_1, [B_{\mathsf{pc}}(\sigma)]_1$.

We show that $\mathcal{A}$ cannot tell apart the real and simulated arguments by showing that each argument element has identical distribution in the real 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 $\mathbb{x}$. Next, $\tilde{z}$ 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 $b = 4$, $\tilde{z}(\sigma)$ (resp., $z(\sigma)$) has the same distribution as in the real case.

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 $\tilde{z}(\sigma)$, $[\psi_{\mathsf{ipc}}(\sigma)]_1$ has the same distribution in the honest and in the simulated argument.

The values $v_M = M^b(\alpha, \beta)$ and $[R(\sigma)\tau]_1$ are independent of the witness and computed honestly by the simulator. Finally, $[B_{\mathsf{pc}}(\sigma, \tau)]_1$ 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. □

We prove subversion zero-knowledge in Appendix B.

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

References

1. Abdolmaleki, B., Baghery, K., Lipmaa, H., Zajac, M.: A subversion-resistant SNARK. In: ASIACRYPT 2017, Part III. LNCS, vol. 10626, pp. 3–33
2. Abdolmaleki, B., Lipmaa, H., Siim, J., Zajac, M.: On subversion-resistant SNARKs. Journal of Cryptology **34**(3) (2021) p. 17
3. Bauer, B., Fuchsbauer, G., Loss, J.: A classification of computational assumptions in the algebraic group model. In: CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 121–151
4. Bellare, M., Fuchsbauer, G., Scafuro, A.: NIZKs with an untrusted CRS: Security in the face of parameter subversion. In: ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 777–804
5. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent succinct arguments for R1CS. In: EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 103–128
6. Bitansky, N., Chiesa, A., Ishai, Y., Ostrovsky, R., Paneth, O.: Succinct non-interactive arguments via linear interactive proofs. In: TCC 2013. LNCS, vol. 7785, pp. 315–333
7. Boneh, D., Drake, J., Fisch, B., Gabizon, A.: Efficient polynomial commitment schemes for multiple points and polynomials. Cryptology ePrint Archive, Report 2020/081 (2020) `https://eprint.iacr.org/2020/081`.
8. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 327–357
9. Bootle, J., Chiesa, A., Sotiraki, K.: Sumcheck arguments and their applications. In: CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 742–773
10. Bowe, S.: BLS12-381: New zk-SNARK Elliptic Curve Construction. Blog post, `https://blog.z.cash/new-snark-curve/`, last accessed in July, 2018 (2017)
11. Campanelli, M., Faonio, A., Fiore, D., Querol, A., Rodríguez, H.: Lunar: a toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. In: ASIACRYPT 2021 (3). LNCS, vol. 13092, pp. 3–33
12. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.P.: Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In: EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 738–768
13. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: CRYPTO'86. LNCS, vol. 263, pp. 186–194
14. Fuchsbauer, G.: Subversion-zero-knowledge SNARKs. In: PKC 2018, Part I. LNCS, vol. 10769, pp. 315–347
15. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 33–62
16. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953 (2019) `https://eprint.iacr.org/2019/953`.

17. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645
18. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340
19. Groth, J.: On the size of pairing-based non-interactive arguments. In: EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 305–326
20. Groth, J., Kohlweiss, M., Maller, M., Meiklejohn, S., Miers, I.: Updatable and universal common reference strings with applications to zk-SNARKs. In: CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 698–728
21. Izabachène, M., Libert, B., Vergnaud, D.: Block-wise P-signatures and non-interactive anonymous credentials with efficient attributes. In: 13th IMA International Conference on Cryptography and Coding. LNCS, vol. 7089, pp. 431–450
22. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: ASIACRYPT 2010. LNCS, vol. 6477, pp. 177–194
23. Kohlweiss, M., Maller, M., Siim, J., Volkhov, M.: Snarky ceremonies. In: Advances in Cryptology – ASIACRYPT 2021, pp. 98–127
24. Kosba, A.E., Papadopoulos, D., Papamanthou, C., Song, D.: MIRAGE: Succinct arguments for randomized algorithms with applications to universal zk-SNARKs. In: USENIX Security 2020, pp. 2129–2146
25. Libert, B., Yung, M.: Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In: TCC 2010. LNCS, vol. 5978, pp. 499–517
26. Lipmaa, H.: Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In: TCC 2012. LNCS, vol. 7194, pp. 169–189
27. Lipmaa, H., Siim, J., Zajac, M.: Counting Vampires: From Univariate Sumcheck to Updatable ZK-SNARK. Technical Report 2022/406, IACR (2022)
28. Lund, C., Fortnow, L., Karloff, H.J., Nisan, N.: Algebraic methods for interactive proof systems. In: 31st FOCS, pp. 2–10
29. Maller, M., Bowe, S., Kohlweiss, M., Meiklejohn, S.: Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In: ACM CCS 2019, pp. 2111–2128
30. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy, pp. 238–252
31. Råfols, C., Zapico, A.: An algebraic framework for universal and updatable SNARKs. In: CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 774–804
32. Råfols, C., Zapico, A.: An Algebraic Framework for Universal and Updatable SNARKs. Technical Report 2021/590, IACR (2021) Last checked modification from August 19, 2021.
33. Thaler, J.: Time-optimal interactive proofs for circuit evaluation. In: CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 71–89
34. Zhang, Y., Szepieniec, A., Zhang, R., Sun, S., Wang, G., Gu, D.: VOProof: Efficient zkSNARKs from Vector Oracle Compilers. Technical Report 2021/710, IACR (2021) Last checked modification from Mar 12, 2022.

A Vampire's Efficiency Comparison

Next, we establish the efficiency of Vampire. In Table 2, we provide an extensive comparison with other recent updatable and universal zk-SNARKs. We will

Table 2. Efficiency comparison of updatable and universal zk-SNARKs. 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 is the bounded fan-out, n_k is the number of non-zero elements of the matrix that describe the circuit, a is the number of additive gates, N_k, A, M, M_0, V are maximum supported values for n_k, a, m, m_0, v. We omitted constant terms in $|\mathsf{srs}|$ and KGen.

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	$4M$	$36m$	$273m$	$7P$	
[29]	$\mathbb{G}_2$	$4M$	3	–	$4M$	–	–		[8] 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**
[12]	$\mathbb{G}_2$	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	$3n_g$	$8n_g$	$11n_g$	$2P$	Plonk constraints
[16]	$\mathbb{G}_2$	1	1	–	–	–	–		
	$\mathbb{F}$	–	–	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**
[11]	$\mathbb{G}_2$	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**
[11]	$\mathbb{G}_2$	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)	$\mathbb{G}_2$	–	–	–	–	–	–		
[31], §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	$11n_g$	$8n_g$	$2P$	Plonk constraints
[32], Fig 11	$\mathbb{G}_2$	1	1	–	–	–	–		
	$\mathbb{F}$	–	–	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
[32], App F	$\mathbb{G}_2$	1	1	–	–	–	–		
	$\mathbb{F}$	–	–	2	–	$O(m \log m)$	$O(m \log m)$	$O(m_0 + \log m)$	
Vampire	$\mathbb{G}_1$	$12M + N_k$	–	4	$12M + N_k$	–	$20M + 2N_k$	$5\mathbb{G}_1 + 21\mathbb{G}_2 + 6P$	**R1CSLite with sparse matrices**
(sparse matrices)	$\mathbb{G}_2$	$4M + N_k$	22	–	$4M + N_k$	$120M + 18N_k$	–		
current paper	$\mathbb{F}$	–	–	2	–	$O(N_k \log N_k)$	$O(N_k \log N_k)$	$O(M_0 + \log M)$	

provide some more background to understand the table and reproduce some of the numbers given there.

Recall that $n_h = |\mathbb{H}|$ is equal to $2m + b = 2m + 4$, m is the dimension of the matrices $\mathbf{L}$ and $\mathbf{R}$ (the number of multiplication gates), and $n_k = |\mathbb{K}|$ is equal to the total number of non-zero entries in $\mathbf{L}$ and $\mathbf{R}$.

$\mathbf{M/N_k}$ versus $\mathbf{m/n_k}$. In most of the recent efficient updatable and universal zk-SNARKs [12,31,11], KGen depends on $M \geq m$ and $N_k \geq n_k$, a priori fixed upper bounds on m and n_k, while the actual values of m and n_k are fixed only when invoking Derive. This has the added benefit that one can share a single long SRS between many applications that have varied instance sizes. At the same time, the complexity of the prover and the verifier depends on n_k and m and not on N_k and M.

Table 3. $\mathbb{G}_1$ elements, needed in $\mathfrak{Vampire}$'s SRS for prover's computation.

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

Since our main goal was to minimize the argument length, we had to make several trade-offs. One of such trade-offs of $\mathfrak{Vampire}$ is that m and n_k cannot be adjusted in $\mathfrak{Vampire}$ after the SRS has been generated. Thus, the prover's complexity depends on M and N_k. Because of that, in Table 2, we differentiate between N_k and n_k, and M and m.

From a practical perspective we do not expect this to result in a major increase in the prover's computation time. For example, in a typical blockchain application, the zk-SNARK must support different relations since there might be updates to the underlying blockchain protocol. That does not however mean that the constraint system's size will necessarily change by a large extent. If there indeed is a need to support constraint systems of very different sizes, we recommend to simply generate SRSs (each using different trapdoors) for several different values of m (say, 2^{28}, 2^{24}, and 2^{20}) and n_k (say, 2^{32}, 2^{28}, and 2^{24}). This will increase the total length of the SRS by a small constant factor while enabling one to scale the prover's work more precisely with the actual values of m and n_k.

In the following discussion, we assume for simplicity that $N_k = n_k$ and $M = m$.

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 a number of $\mathbb{G}_1$ elements that we explain in Table 3 (we list the group elements output by P, important underlying polynomials, and the needed SRS elements).

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

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

- To check (♯♯), $\mathsf{srs}_{\mathcal{R}}$ needs to contain 22 polynomial evaluations in $\mathbb{G}_2$. Those values can be computed from $[(\sigma^i)_{i=0}^{n_k+d_{\mathsf{gap}}+2}]_2$. For a more precise computation, we will enlist Table 4 which $\mathbb{G}_2$ elements are needed and why. Here, $f \in \{\mathsf{rcv}, \mathsf{col}, \mathsf{row}, \mathsf{rc}\}$, $g \in \{\mathsf{zcol}, \mathsf{zrow}, \mathsf{zrc}\}$, $i_1 \in \{0, 1\}$, $i_2 \in \{0, 1, 2\}$, $i_3 \in \{0, 1, 2, 3\}$.
- To verify that the SRS is well-formed, $[\sigma^2]_2$ must be in the SRS. We use it to check that $[\sigma^{n_k}]_2$ is correctly computed with the equation $[\sigma^{n_k-2}\tau]_2 \bullet [\sigma^2]_2 = [\tau]_1 \bullet [\sigma^{n_k}]_2$ (see the proof of Theorem 3).

Table 4. $\mathbb{G}_2$ elements, needed in $\mathfrak{Vampire}$'s SRS for verifier's computation.

Group element	SRS elements
$[\sigma^{i_3}S(\sigma)]_2$	$[(\sigma^{d_{\mathsf{gap}}-jn_h+i})_{j\in\{0,1,2\},i\in\{0,1,2,3\}}]_2$
$[f(\sigma)S(\sigma)]_2$	$[(\sigma^{d_{\mathsf{gap}}-jn_h+i})_{j\in\{0,1,2\},i\in[0,n_k-1]}]_2$
$[\sigma^{i_2}g(\sigma)S(\sigma)]_2$	$[(\sigma^{d_{\mathsf{gap}}-jn_h+i})_{j\in\{0,1,2\},i\in[0,n_k+1]}]_2$
$[\sigma^{i_2}\mathcal{Z}_{\mathbb{K}}(\sigma)S(\sigma)]_2$	$[(\sigma^{d_{\mathsf{gap}}-jn_h+in_k+i_2})_{i_2,j\in[0,2],i\in\{0,1\}}]_2$
$[\sigma^{i_1}\mathcal{Z}_{\mathbb{K}}(\sigma)]_2$	$[1,\sigma,\sigma^{n_k},\sigma^{n_k+1}]_2$

Thus, the SRS must contain $[\sigma^i]_2$ for $i \in \mathcal{T}$, where

$$\mathcal{T} = \mathcal{U} \cup (\mathcal{U}+n_h) \cup (\mathcal{U}+2n_h) \cup \mathcal{V} \ ,$$

$\mathcal{U} = [d_{\mathsf{gap}} - 2n_h, d_{\mathsf{gap}} - 2n_h + n_k + 2]$ and $\mathcal{V} = \{0, 1, 2, n_k, n_k + 1\}$. Let us first compute the size of the set $\mathcal{T}' := \mathcal{U} \cup (\mathcal{U} + n_h) \cup (\mathcal{U} + 2n_h)$. The size of $\mathcal{T}'$ depends on whether these three intervals overlap or not, which depends on how n_k and n_h are related. Let $A = d_{\mathsf{gap}} - 2n_h$, $B = n_k + 2$, and $C = n_h$. Note that $|\mathcal{U}| = B + 1$ and C is the step between the start points of the consequent intervals $\mathcal{U} + in_h$ and $\mathcal{U} + (i+1)n_h$. Clearly, $|\mathcal{U} \cup (\mathcal{U} + n_h) \cup (\mathcal{U} + 2n_h)|$ is equal to

- $B+2C+1$ if $B \geq C$, that is, $(n_k+2)+2n_h+1 = 2n_h+n_k+3$ if $n_k \geq n_h-2$ (this is the usual case),
- $3(B+1) = 3C$ if $B = C-1$, that is, $3n_h$ if $n_k = n_h - 3$,
- $3(B+1)$, if $B < C-1$, that is, $3n_k + 9$, if $n_k < n_h - 2$.

Finally, the set $\mathcal{V}$ contains five elements, each of which may overlap with the rest of the elements.

Thus, srs contains $(d_{\mathsf{gap}} + d + 1 - 1) + (n_k - 2 + 1) = 6(n_h - 1) + (n_k - 1) = 6n_h + n_k - 7 = 12m + n_k + 6b - 7 = 12m + n_k + 17$ elements of $\mathbb{G}_1$. Assuming $n_k \geq n_h - 2$, srs contains $\leq 2n_h + n_k + 8 = 4m + n_k + 2b + 8 = 4m + n_k + 16$ elements of $\mathbb{G}_2$.

Notably, if $n_k \gg m$, the SRS length is dominated by n_k elements of both groups. In most applications, $n_k > 6m$, but for the sake of freedom, one probably wants to choose a larger n_k usually.

Computational Complexity of KGen. Key generation is dominated by the need to compute all SRS elements, and is thus the same as SRS length (in scalar multiplications).

Complexity of Derive. In Derive, one needs to compute a number of polynomials. The computational complexity is the sum of the degrees plus the number of polynomials, that is, $(\sum_{i=0}^{3}(d_{\mathsf{gap}} + i) + 4 \cdot (n_k - 1 + d_{\mathsf{gap}}) + n_k + (n_k + 1) + \sum_{i=0}^{2}(3 \cdot (i + n_k - 1 + d_{\mathsf{gap}}) + (i + n_k + d_{\mathsf{gap}}))) + 22 = 20d_{\mathsf{gap}} + 18n_k + 28 = 60n_h + 18n_k - 32 = 120m + 18n_k + 208$ scalar multiplications in $\mathbb{G}_2$.

Prover's Computation. Like in other efficient updatable and universal zk-SNARKs with constant communication, the prover's computation is quasilinear.

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 polynomial openings $\tilde{z}_{\mathsf{pc}}(Y)$, $\Phi_{\mathsf{pc}}(Y)$, and $Q(Z)$.

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

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 5 scalar multiplications in $\mathbb{G}_1$, 21 scalar multiplications in $\mathbb{G}_2$, and six pairings.

We note that using $\mathfrak{Count}$ eliminates one FFT but adds cryptographic operations. Decreasing the prover's computation is an interesting open question.

Summatory Efficiency Comparison of $\mathfrak{Vampire}$. See Table 2 for efficiency comparison with previous work. Essentially, we copied the efficiency comparison table of Table 2 from [32] 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. In the case of the sparse R1CSLite constraint system, yellow cells contain the absolute best entries as functions of n_k (or N_k) only.

As we see from Table 2, $\mathfrak{Vampire}$ has very good efficiency when measured as a function of n_k and somewhat worse efficiency as a function of m. In other words, $\mathfrak{Vampire}$ is very competitive when n_k is relatively large compared to m. In practice, 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 $2m+b$. On the other hand, using aggregated polynomial openings and $\mathfrak{Count}$ increases the SRS size.

It is possible to "demake" $\mathfrak{Vampire}$ by removing some of the aggressive length-optimization to obtain a larger argument size but better (say) the SRS size. We leave it as an open question about which optimization should be removed first or whether this is needed at all.

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 Marlin's; the same holds for the complexity of KGen. The complexity of Derive is $18n_k$ in the case of $\mathfrak{Vampire}$, which is noticeably beaten only by [31], but comparable to other zk-SNARK for the same constraint system. This is also likely to be one of the less important parameters in practice since it only has to be done once per a

relation. The prover's computation is $2n_k$, compared to $4n_k$ in the case of [31] (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, as a function of n_k, among any known updatable and universal zk-SNARKs. The verifier's computation is dominated by $\Theta(m_0 + \log m)$ finite field multiplications, with a very small constant.

Arguably, the argument size and verifier's computation are long-term most important measures in blockchain-like applications where one stores and re-verifies arguments many times. Many other papers optimize the prover's computation but sacrifice argument length. Since our goal is diametrically opposite, we are not competing with such papers.

B Subversion Zero Knowledge

We show that $\mathfrak{Vampire}$ is subversion zero-knowledge (Sub-ZK, [4,1,14,2]), i.e., $\mathfrak{Vampire}$ stays zero-knowledge even when the SRS generator is compromized. For this, we first modify the Sub-ZK definition of [2] 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. 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 $N \in \mathsf{poly}(\lambda)$,

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

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

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 [1,2]. Intuitively, BDH-KE states that if an adversary, given p, outputs $([\sigma]_1, [\sigma]_2)$, then one can extract σ. 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 assumption.*

Proof. As proven in [2] (for NIZKs, but their proof generalizes to interactive arguments), 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.

Subversion zero-knowledge follows from these properties since if SrsVer accepts then the SRS is correct (belongs to the range of KGen for some trapdoor) 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 $\mathcal{S}_2(X) = \{1, X, X^2, X^{n_k}, X^{n_k+1}, (X^{d_{\mathsf{gap}}-n_h i+j})_{i\in\{0,1,2\},j\in[0,n_k+2]}\}$. 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 (here, $[1]_1, [1]_2$ are pairing generators and $[\sigma]_1$ and $[\tau]_1$ are group elements that correspond to arbitrarily chosen values of the trapdoors). After a check of $\underline{[x]_\iota}$ in an equation where all other variables are already verified, we think of $\underline{[x]_\iota}$ being verified too, that is, it will not be underlined in the following equations. 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. Check $[\sigma]_1 \bullet [\sigma]_2 = [1]_1 \bullet \underline{[\sigma^2]_2}$.
3. For $i \in [1, d_{\mathsf{gap}} - 1]$: check $[\sigma^{i-1}]_1 \bullet [\sigma]_2 = \underline{[\sigma^i]_1} \bullet [1]_2$.
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 $i \in [1, n_k - 2]$: check $[\sigma^{i-1}\tau]_1 \bullet [\sigma]_2 = \underline{[\sigma^i\tau]_1} \bullet [1]_2$.
7. Check $[\sigma^{n_k-2}\tau]_1 \bullet [\sigma^2]_2 = [\tau]_1 \bullet \underline{[\sigma^{n_k}]_2}$.
8. Check $[\sigma]_1 \bullet [\sigma^{n_k}]_2 = [1]_1 \bullet \underline{[\sigma^{n_k+1}]_2}$.
9. Check $[\sigma^{d_{\mathsf{gap}}-1}]_1 \bullet [\sigma]_2 = [1]_1 \bullet \underline{[\sigma^{d_{\mathsf{gap}}}]_2}$.
10. Check $[\sigma^{d_{\mathsf{gap}}-n_h}]_1 \bullet [1]_2 = [1]_1 \bullet \underline{[\sigma^{d_{\mathsf{gap}}-n_h}]_2}$.
11. Check $[\sigma^{d_{\mathsf{gap}}-2n_h}]_1 \bullet [1]_2 = [1]_1 \bullet \underline{[\sigma^{d_{\mathsf{gap}}-2n_h}]_2}$.
12. For $k \in \{d_{\mathsf{gap}} - n_h i + j\}_{i=\{0,1,2\},j\in[1,n_k+2]}$; check $[\sigma]_1 \bullet [\sigma^{k-1}]_2 = [1]_1 \bullet \underline{[\sigma^k]_2}$.

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 $\mathsf{SrsVer}(\mathsf{srs}) = 1$, 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. □