A Crack in the Firmament: Restoring Soundness of the Orion Proof System and More

Thomas den Hollander* Daniel Slamanig†

Abstract

Orion (Xie et al. CRYPTO'22) is a recent plausibly post-quantum zero-knowledge argument system with a linear time prover. It improves over Brakedown (Golovnev et al. ePrint'21 and CRYPTO'23) by reducing the proof size and verifier complexity to be polylogarithmic and additionally adds the zero-knowledge property. The argument system is demonstrated to be concretely efficient with a prover time being the fastest among all existing succinct proof systems and a proof size that is an order of magnitude smaller than Brakedown. Since its publication in CRYPTO 2022, two revisions have been made to the zk-SNARK. First, there was an issue with how zero-knowledge was handled. Second, Orion was discovered to be unsound, which was then repaired through the use of a commit-and-prove SNARK as an "outer" SNARK.

As we will show in this paper, unfortunately, Orion in its current revision is still unsound (with and without the zero-knowledge property) and we will demonstrate practical attacks on it. We then show how to repair Orion without additional assumptions, with the resuling polynomial commitment denoted as Scorpius, which requires non-trivial fixes when aiming to preserve the linear time prover complexity. The proposed fixes lead to an even improved efficiency, i.e., smaller proof size and verifier time, over the claimed efficiency of the initial version of Orion. We also apply the recent ideas of Diamond and Posen (CiC'24) to make the challenge in Orion logarithmically sized. Moreover, we provide the first rigorous security proofs and explicitly consider multi-point openings and non-interactivity. While revisiting Orion we make some additional contributions which might be of independent interest, most notable an improved code randomization technique that retains the minimum relative distance.

1 Introduction

Zero-knowledge proof systems are a fascinating concept and were introduced by Goldwasser, Micali, and Rackoff [18] in the 1980s. The basic idea is that a prover who holds a witness to an NP statement can convince any verifier, who only knows the statement, of the validity of the statement without revealing any information about the witness. Besides being an interesting theoretical object of study on its own, it has been shown to have numerous practical cryptographic applications, e.g., as a tool to prevent malicious behavior in multi-party protocols, enabling privacy-friendly authentication or to construct digital signatures. While for many years most of these concepts were only of theoretical interest, nowadays zero-knowledge proofs see widespread adoption. One famous (implicit) long-term use of zero-knowledge proofs are Schnorr signatures [26], which can be seen as a particular example of Sigma protocols made non-interactive via the Fiat-Shamir heuristic [16]. However, also their explicit construction has gained in popularity with the increased need for post-quantum replacements for existing signatures. One particularly popular approach is via the multi-party computation in the head (MPCitH) paradigm [20], enabling schemes that do not require structured hardness assumptions, e.g., Picnic [11] or FAEST [3], but which meanwhile turned into a popular paradigm also for structured hardness assumptions.

The arguably most quickly developing domain related to zero-knowledge is that of zk-SNARKs (succinct non-interactive arguments of knowledge) [5]. Those are computationally sound zero-knowledge proofs that are knowledge sound (i.e., if a proof is accepted by a verifier it is guaranteed that the prover knows the witness), non-interactive (i.e., there is only a single message sent from the

*Research Institute CODE, Universität der Bundeswehr München; thomasdh@unibw.de
†Research Institute CODE, Universität der Bundeswehr München; daniel.slamanig@unibw.de

prover to the verifier) and they are succinct, i.e., the size of their proofs is sublinear in the size of the witness to be proven. A lot of the developments are spurred by practical applications in blockchains and cryptocurrencies.

There are various techniques to construct zk-SNARKS that come with different trade-offs in proof sizes as well as prover and verifier complexities (cf. [25, 28] for a comprehensive overview). In the following we will focus on the construction of SNARKS that achieve linear time prover complexity and square root (or even polylogarithmic) proof complexity. While the first schemes that fall into this category [6, 7, 8] seemed to be only of theoretical interest, recent works [19, 29] have shown how following these ideas can yield practically efficient schemes. We recall that a popular paradigm to construct SNARKs is to rely on an information theoretic primitive often abstracted in the form of a polynomial IOP (PIOP) [9] which is then combined with a polynomial commitment scheme [21] and made non-interactive using the Fiat-Shamir heuristic. Brakedown [19] observes that [7] implicitly describe a polynomial commitment scheme with a linear time commitment phase (relying on linear-time encodable codes), which they make explicit and combine it with the PIOP for R1CS from Spartan [27] to obtain a SNARK. The so obtained SNARK is not yet zero-knowledge, but can be made so using one layer of recursive proof composition, i.e., using an "outer SNARK" that is zero-knowledge to prove knowledge of a proof of the "inner" SNARK (that does not need to be zero-knowledge).

In this paper we are focusing on Orion [29], which improves over Brakedown [19]. Due to the used building blocks Orion like Breakdown is plausibly post-quantum secure and is transparent, i.e., does not require a trusted setup, both being highly desirable properties in practice. Moreover, to the best of our knowledge, under those SNARKs that are plausibly post-quantum, Orion has the fastest prover time among all existing schemes focusing on linear-time provers in the literature.

High-level overview of Orion. The Orion proof system [29] improves on Brakedown in two ways. First, by improving the linear-time encodable code used, to have a good minimum distance except with negligible probability. Second, a "code-switching" proof composition technique is employed with the goal of reducing the proof size and verifier time from $O(\sqrt{N})$ to $O(\log^2(N))$. Since the verifier work for Brakedown is $O(\sqrt{N})$, it is possible to use an outer zk-SNARK that is quasilinear while maintaining the overall linear-time prover. Specifically, Virgo [32] is used as the outer zk-SNARK.

To commit, the prover first parses the polynomial coefficients as a $k \times k$ matrix W. The matrix is then encoded by first encoding every row using the linear code $\mathrm{E_C}$. To achieve zero-knowledge, a random row $\vec{r}_i$ is sampled by the prover, which is then added as well as appended to every codeword. This yields matrix C. After that, every column of the resulting matrix is again encoded using $\mathrm{E_C}$, giving matrix D. The prover finally creates a Merkle tree commitment to D to obtain the commitment $\mathcal{R}$ to the polynomial.

In the evaluation procedure, the prover attempts to convince the verifier of a certain evaluation of the committed polynomial.[1] The verifier sends a random vector $\vec{\gamma}$ to the prover, which the prover uses to compute $\vec{y_\gamma}$, a linear combination of the rows of W. In Brakedown, this row is sent to the verifier, who checks correctness by opening a number of columns at random and verifying that for every selected column index j this linear combination has been computed correctly. Furthermore, due to the use of linear codes, any inconsistencies will be caught with overwhelming probability. Orion uses the same idea, but performs these checks in the outer zk-SNARK. In particular, the prover computes

$$\vec{c_\gamma}_j = \sum_i \vec{\gamma}_i C_{ij}, \quad \vec{y_\gamma}_j = \sum_i \vec{\gamma}_i W_{ij}, \quad \vec{r_\gamma} = \sum_i \vec{\gamma}_i \vec{r}_i,$$

and sends a commitment to $\vec{c_\gamma}$ to the verifier. At this point the verifier samples a random column set J, after which the prover additionally commits to $\vec{y_\gamma}$ and $\vec{r_\gamma}$, as well as $C_{\cdot j}$ for all columns $j \in J$. For this it uses the outer SNARK, which is in the form of a commit-and-prove SNARK (CP-SNARK), i.e., a part of the witness is hidden in a commitment and the zk-SNARK proves the relation and the consistency of the witness in the commitments. The verifier then provides a row index set I, after which the prover generates the zk-SNARK proof. Here, $\mathcal{P}$ checks that $\langle \vec{\gamma}, \vec{C_{\cdot j}} \rangle = \mathrm{E_C}(\vec{y_\gamma} + \vec{r_\gamma}) || \vec{r_\gamma}$ at columns $j \in J$. This is identical to the check done in Brakedown, modulo the added

[1] In [29], `Eval` is conceptually split into a prove and a verify phase. We merge these, since $\mathcal{V}$ should only be convinced if the challenges were sampled honestly: in neither the interactive nor the non-interactive setting does `Verify` suffice as a standalone verification procedure.

zero-knowledge. The zk-SNARK also outputs $\vec{c_{\gamma_j}}$ so that the verifier can check consistency with the earlier commitment. Lastly, the proof outputs the column encoding using $\mathrm{E}_C(D_{\cdot j})_i$ at indices $i \in I$, with the goal that these can be checked with the commitment to D without having to open the columns entirely. Since we have a constant number of openings, the proof size is asymptotically dominated by the proof size of the outer SNARK.

Issues with the soundness of Orion. Since its publication in CRYPTO 2022 [29], two revisions have been made to the zk-SNARK in the paper.[2] First, there was an issue with how zero-knowledge was handled. In short, the polynomial $\phi(\vec{x})$ was masked by sampling a random polynomial $r(\vec{x})$ and proving evaluations of $(\phi + r)(\vec{x})$ as well as $r(\vec{x})$. However, since encoding the polynomials added redundancy, this could not be simulated. This was fixed in a revised version [30].

Second, due to an issue in the challenge order, Orion was discovered to be unsound. In particular, the row index set I would be provided too early on in the proof generation, causing the prover to be able to cheat. The authors of Hyperplonk [12] are credited with discovering the problem. Indeed, the same issue is not present in their polynomial commitment scheme Orion+, which has a much smaller proof size but cannot be post-quantum secure because of its reliance on pairing friendly elliptic curve groups. The mentioned soundness issue was subsequently repaired by the authors of Orion in another revised version [31] through the use of a CP-SNARK, which forces the prover to commit to the witness before the index set I is provided by the verifier. This indeed prevents some ways in which the prover could cheat. Unless noted otherwise, when we henceforth talk about Orion we refer to this latest revision [31].

As we will demonstrate, unfortunately the Orion polynomial commitment in its current form, and by extension the Orion proof system, is still unsound. This is due to a second issue in the challenge-response order. In this paper, we will demonstrate that this issue indeed exists, and that it is practical to forge a proof such that *any* committed polynomial can be opened to *any* desired evaluation. The issue is inherent to the protocol and persists when considering its non-zero-knowledge version. We also provide an implementation of this attack, where we use additional techniques to make it efficient enough to be practical.

The issue lies in the fact that unlike the row set I, the column set J is known to the prover before committing to the CP-SNARK witness, which enables the prover to adaptively choose a witness that will pass the subsequent verification procedure. In fact, this is required for efficiency: if this index set was not known at this point, the prover would need to commit to C in its entirety.

> *Consequently, when using Virgo as the outer zk-SNARK as Orion does, this would cost $O(N \log(N))$ prover time, meaning that the prover no longer runs in overall linear time and thus invalidating the claims of having a linear time prover.*

Resurrecting Orion. To remedy the unsoundness and simultaneously retain succinctness, we find a way to split the column index set J between rounds. This way, we can use one challenge set J that is unknown to the prover when committing to the witness of the CP-SNARK, while using another challenge set $\hat{J}$ such that the prover can commit to $C_{\cdot j}$ instead of C in its entirety.

However, this is not possible without breaking the zero-knowledge property of the scheme, which is why we also provide a new method of adding zero-knowledge while retaining the linear-time prover. For this we use zero-knowledge codes [8], which we instantiate concretely using masking polynomials of constant degree. This way, we can retain the linear-time prover. In the process we obtain a new polynomial commitment scheme, which we call Scorpius, with smaller proof sizes and a more efficient verifier, that also works for polynomials of lower degree (beyond just padding up to a supported degree).

Lastly, Orion is actually not fully succinct due to the use of a challenge of size $\sqrt{N}$, making the verifier who needs to sample this challenge have complexity $O(\sqrt{N})$. A recent work by Diamond and Posen [14] addresses this issue by considering the tensor product $\bigotimes_{i=1}^{\log(k)}(1 - \vec{\eta}_i, \vec{\eta}_i)$ instead of a uniformly random vector from $\mathbb{F}^k$. We adapt their results for Orion to obtain a fully succinct polynomial commitment scheme.

[2] Actually, the paper had three revisions. The third revision, however, is concerned with the testing whether a random bipartite graph is a lossless expander (required for the generalized Spielman codes) and is not relevant to our treatment of Orion in this paper.

1.1 Summarizing our Contributions

Subsequently, we will briefly summarize the contributions of this work.

- We take another look at the Orion proof system [31] and show its unsoundness, which we also support by giving explicit and efficient attacks.
- We show how to overcome these existing soundness issues and thus resurrect Orion. In order to not introduce additional assumptions and to maintain a linear time prover though, this requires fixes that are non-trivial. Notably, with our fixes we achieve a smaller proof size and verifier time while keeping prover time comparable. We also apply the recent ideas in [14] to make the proof Orion system fully succinct.
- We provide rigorous formal proofs for the modified Orion construction to show that our fixes indeed solve the soundness issues. We provide a new simulator and the first rigorous analysis of the zero-knowledge property. This is necessary because the current simulator used in [31] does not satisfy their own definition, since it requires knowledge of the evaluation point at the time of committing. Lastly, we provide an explicit treatment of how to make the Orion polynomial commitment scheme non-interactive using the Fiat-Shamir transformation [16].
- In the course of fixing Orion, we obtain a new polynomial commitment scheme, which we call Scorpius, with smaller proof sizes and a more efficient verifier, that also works for polynomials of lower degree and does not require padding up to a supported degree. We extend the polynomial commitment to open at multiple points by taking a random linear combination of all evaluation vectors while retaining zero-knowledge. Moreover, zero-knowledge holds even if the evaluation point is not known at the time of committing.
- We provide a general transformation to perfectly zero-knowledge codes [8], which may be of independent interest. In brief, instead of a random vector to mask the witness we use a randomized polynomial. This retains the same relative minimum distance instead of reducing it by half as in the vector masking approach (which is used by Orion), while retaining linear encodability.

2 Preliminaries

For security parameter λ, we write $\text{negl}(\lambda)$ for the negligible function. We also write $a \approx_\lambda b$ as shorthand for $|a - b| \leq \text{negl}(\lambda)$ and $a||b$ for the concatenation of strings a and b.

We write $w\left(\vec{a}, \vec{b}\right)$ to denote the Hamming weight of vectors $\vec{a}$ and $\vec{b}$ of equal length n, and $\Delta\left(\vec{a}, \vec{b}\right) = \{i \in [n] | \vec{a}_i \neq \vec{b}_i\}$ as well as $d\left(\vec{a}, \vec{b}\right) = \left|\Delta\left(\vec{a}, \vec{b}\right)\right|$. For vector space L, we define $w(\vec{a}, L)$ as $w(\vec{a}, \vec{c})$ for the $c \in L$ that minimizes the weight—$\Delta(\vec{a}, L)$ and $d(\vec{a}, L)$ are defined analogously. For matrix $M \in \mathbb{F}^{n \times m}$, we write $d(M, L^n)$ for the number of columns at which M differs from C for the $C \in L^m$ for which this number is minimal, and $\Delta(M, L)$ for the corresponding column indices.

2.1 Randomized Linear Codes

Following [8], we make use of *randomized linear codes*, which are encoding functions $\tilde{\text{E}_\text{C}}(\vec{m}; \vec{r}) = \text{E}_\text{C}(\vec{m} \mid\mid \vec{r})$, where E_C is some linear code $\mathbb{F}^{k_m + k_r} \to \mathbb{F}^{n_m + n_r}$ and the message $\vec{m} \in \mathbb{F}^{k_m}$ and randomness $\vec{r} \in \mathbb{F}^{k_r}$ are logically separated. We simply write $\tilde{\text{E}_\text{C}}(\vec{m})$ if $\vec{r} \in_R \mathbb{F}^{k_r}$.

Definition 2.1. A randomized linear code $\tilde{\text{E}_\text{C}}$ is *q-query uniform* if for any message $m \in \mathbb{F}^{k_m}$, any q adaptively chosen locations of $\text{E}_\text{C}\tilde{(m)}$ are distributed uniformly at random.

An example of a q-query uniform linear code is the Reed-Solomon code $\text{RS}_{[n, k_m + q, d]}$ with $n = k_m + q + d - 1$. This follows from the fact that there is exactly one codeword that conforms to any $k_m + q$ evaluations.

2.2 Collision-Resistant Hashing and Merkle Trees

Let us consider a family $\{H_s\}_{s\in I}$ of functions for which there exists a PPT key generation algorithm $s \leftarrow \mathtt{Gen}(1^\lambda)$. Given the security parameter λ, it outputs an index $s \in I$, or equivalently a function $H_s : \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{\ell'(\lambda)}$ with $\ell(\lambda) > \ell'(\lambda)$. We require that there exists a PPT evaluation algorithm that on input $s \in I$ and $x \in \{0,1\}^{\ell(\lambda)}$ computes $H_s(x)$. The collision-resistance property for such a family states that every PPT algorithm $\mathcal{A}$ that is given $s \leftarrow \mathtt{Gen}(1^\lambda)$, and outputs a pair $x \neq x'$ such that $H_s(x) = H_s(x')$ only succeeds with negligible probability.

The following presentation is based on the one in [2]. For a collision-resistant hash function[3] H, a Merkle tree hash [24] is a succinct commitment to a list $k = 2^d$ of values represented by a single hash value h (the root hash) such that opening the commitment on any of the k values requires revealing $O(d)$ hash values.

Let $m_1, \ldots, m_k$ be k messages, then a Merkle tree is represented by a binary tree of depth d where the messages m_i are assigned to the leafs of the tree from left to right. The values assigned to the internal nodes of the tree are computed as $H(u||v)$ where u and v are the left and right children of the respective node. Consequently, the root of the tree represents a commitment to messages $m_1, \ldots, m_k$. To open the commitment to a message m_i, the opening proof is represented by all the values assigned to nodes on the path from the root to m_i, and the values assigned to the siblings of all these nodes. Verifying the opening proof for m_i recomputes the entire path from m_i to the root bottom-up and compares the so obtained root hash to the commitment value.

The binding of a Merkle hash tree follows from the collision-resistance of the hash function. Moreover, when replacing the hash function H used to compute the Merkle tree by a random oracle, statistical binding follows from the hardness of finding a collision in this model. Additionally, in the random oracle model setting the leaf for every m_i to $m_i||r_i$ where r_i is chosen uniformly at random from $\{0,1\}^\lambda$, the Merkle tree hash is also a statistically hiding commitment.

Throughout this paper, we will use the notation from [2] and denote committing to messages $m_1, \ldots, m_k$ by $h \leftarrow \mathtt{Commit}_M(m_1, \ldots, m_k)$ and opening of a message m_i by $(m_i, \mathrm{path}(i)) \leftarrow \mathtt{Open}_M(h, i)$. Moreover, we denote verification of an opening $(m_i, \mathrm{path}(i))$ by $b \leftarrow \mathtt{Verify}_M(h, m_i, \mathrm{path}(i))$.

2.3 (Commit-and-Prove) SNARKs

In the following we will formally introduce SNARKs as well as commit-and-prove SNARKs (CP-SNARKs).

Succinct non-interactive argument of knowledge (SNARK).

Definition 2.2 (SNARK)**.** A succinct non-interactive argument of knowledge (SNARK) for relation generator $\mathtt{RGen}$ is a tuple of algorithms $\Pi = (\mathtt{Setup}, \mathtt{Prove}, \mathtt{Verify})$:

- $\mathtt{crs} := (\mathtt{pk}, \mathtt{vk}) \leftarrow \mathtt{Setup}(R, \mathtt{aux}_R)$: given a relation and auxiliary information $(R, \mathtt{aux}_R) \leftarrow \mathtt{RGen}(1^\lambda)$ as input; outputs a common reference string $\mathtt{crs}$ consisting of a proving key $\mathtt{pk}$ and a verification key $\mathtt{vk}$.
- $\pi \leftarrow \mathtt{Prove}(\mathtt{pk}, R, \mathtt{aux}_R, \mathtt{u}, \mathtt{w})$ given a proving key $\mathtt{pk}$, a relation R, auxiliary information $\mathtt{aux}_R$, a statement $\mathtt{u}$ and witness $\mathtt{w}$ with $(\mathtt{u}, \mathtt{w}) \in R$; outputs a proof π.
- $b \leftarrow \mathtt{Verify}(\mathtt{vk}, R, \mathtt{aux}_R, \mathtt{u}, \pi)$ given a verification key $\mathtt{vk}$, a relation R, auxiliary information $\mathtt{aux}_R$, a statement $\mathtt{u}$ and a proof π; outputs a bit b indicating reject ($b = 0$) or accept ($b = 1$).

A SNARK satisfies completeness, knowledge soundness, and succinctness. If it is a zk-SNARK it in addition satisfies zero-knowledge. We formally introduce the properties below:

Completeness. Π is complete for $\mathtt{RGen}$, if for all λ, all $(R, \mathtt{aux}_R) \leftarrow \mathtt{RGen}(1^\lambda)$ and $(\mathtt{u}, \mathtt{w}) \in R$

$$\Pr\left[\begin{array}{c}(\mathtt{pk}, \mathtt{vk}) \leftarrow \mathtt{Setup}(R, \mathtt{aux}_R), \pi \leftarrow \mathtt{Prove}(\mathtt{pk}, R, \mathtt{aux}_R, \mathtt{u}, \mathtt{w}) : \\ \mathtt{Verify}(\mathtt{vk}, R, \mathtt{aux}_R, \mathtt{u}, \pi) = 1\end{array}\right] = 1.$$

[3] We will usually omit the key s for the sake of simplicity.

Knowledge soundness. Π is knowledge sound if for all λ, all $(R, \mathtt{aux}_R) \leftarrow \mathtt{RGen}(1^\lambda)$ and any PPT adversary $\mathcal{A}$ there exists a PPT extractor $\mathcal{E}$ with full access to $\mathcal{A}$ such that

$$\Pr\left[\begin{array}{c}(\mathtt{pk},\mathtt{vk}) \leftarrow \mathtt{Setup}(R,\mathtt{aux}_R), ((\mathtt{u},\pi);\mathtt{w}) \leftarrow \mathcal{A}||\mathcal{E}(\mathtt{pk},\mathtt{vk},R,\mathtt{aux}_R) : \\ (\mathtt{u},\mathtt{w}) \notin R \ \wedge\ \mathtt{Verify}(\mathtt{vk},R,\mathtt{aux}_R,\mathtt{u},\pi) = 1\end{array}\right] \approx_\lambda 0$$

Succinctness. Π is succinct if the size of the proof π is $\mathsf{poly}(\lambda + \log|\mathtt{w}|)$ and $\mathtt{Verify}$ runs in time $\mathsf{poly}(\lambda + |\mathtt{u}| + \log|\mathtt{w}|)$.

Zero-knowledge. Π is zero-knowledge (a zk-SNARK) if for all λ, all $(R, \mathtt{aux}_R) \leftarrow \mathtt{RGen}(1^\lambda)$ and $(\mathtt{u},\mathtt{w}) \in R$ there exists a PPT simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$ with $(R, \mathtt{aux}_R, \tau) \leftarrow \mathcal{S}_0(1^\lambda)$ such that for all PPT adversaries $\mathcal{A}$ it holds that

$$\Pr\left[\begin{array}{c}(\mathtt{pk},\mathtt{vk}) \leftarrow \mathtt{Setup}(R,\mathtt{aux}_R), \pi \leftarrow \mathtt{Prove}(\mathtt{pk},R,\mathtt{aux}_R,\mathtt{u},\mathtt{w}) : \\ \mathcal{A}(R,\mathtt{aux}_R,\mathtt{pk},\mathtt{vk},\mathtt{w},\pi) = 1\end{array}\right] \approx_\lambda$$

$$\Pr\left[\begin{array}{c}(\mathtt{pk},\mathtt{vk},) \leftarrow \mathtt{Setup}(R,\mathtt{aux}_R), \pi \leftarrow \mathcal{S}_1(\mathtt{pk},R,\mathtt{aux}_R,\mathtt{u},\tau) : \\ \mathcal{A}(R,\mathtt{aux}_R,\mathtt{pk},\mathtt{vk},\mathtt{w},\pi) = 1\end{array}\right].$$

Commit-and-Prove SNARKs. In a commit-and-prove SNARK (CP-SNARK) one wants to prove knowledge of $(\mathtt{u},\mathtt{w})$ such that $(\mathtt{u},\mathtt{w}) \in R$ holds with respect to a witness $\mathtt{w} = (w,\omega)$ and w is the value committed to in a commitment c_w. We follow the formalization of [10] Campanelli et al. so that only a part of the witness $\mathtt{w}$, i.e., w, is committed to and w splits over some subdomains. Before we start defining CP-SNARKs, we need to introduce commitment schemes.

Definition 2.3 (Commitment scheme)**.** A commitment scheme is a triple of efficient algorithms $(\mathtt{Setup},\mathtt{Commit},\mathtt{Open})$, which are defined as follows:

- $\mathtt{ck} \leftarrow \mathtt{Setup}(1^\lambda)$: takes as input a security parameter λ; outputs the commitment key $\mathtt{ck}$ which implicitly contains a description of the input space $\mathcal{D}$, commitment space $\mathcal{C}$ and opening space $\mathcal{O}$.
- $(c,o) \leftarrow \mathtt{Commit}(\mathtt{ck},m)$: takes as input the commitment key $\mathtt{ck}$ and a value m; outputs a commitment c and opening o.
- $b \leftarrow \mathtt{Open}(\mathtt{ck},c,m,o)$: takes as input the commitment key $\mathtt{ck}$, a commitment c, value m and opening o; outputs a bit b indicating reject $(b=0)$ or accept $(b=1)$.

A commitment scheme is required to provide correctness, binding and hiding. We omit a formal definition of correctness as it is straightforward. The remaining properties are defined as follows.

Binding. A commitment scheme is binding, if for all PPT adversaries $\mathcal{A}$

$$\Pr\left[\begin{array}{c}\mathtt{ck} \leftarrow \mathtt{Setup}(1^\lambda), (c^*,m^*,o^*,m'^*,o'^*) \leftarrow \mathcal{A}(\mathtt{ck}), \\ 1 \leftarrow \mathtt{Open}(\mathtt{ck},c^*,m^*,o^*), 1' \leftarrow \mathtt{Open}(\mathtt{ck},c^*,m'^*,o'^*) : m^* \neq m'^*\end{array}\right] \approx_\lambda 0.$$

Hiding. A commitment scheme is hiding, if for all PPT adversaries $\mathcal{A}$

$$\Pr\left[\begin{array}{c}\mathtt{ck} \leftarrow \mathtt{Setup}(1^\lambda), (m_0,m_1,\mathtt{state}) \leftarrow \mathcal{A}(\mathtt{ck}), b \xleftarrow{R} \{0,1\}, \\ (c,o) \leftarrow \mathtt{Commit}(\mathtt{ck},m_b), b^* \leftarrow \mathcal{A}(\mathtt{ck},c,\mathtt{state}) : b = b^*\end{array}\right] \approx_\lambda \frac{1}{2}.$$

We now present the definition of a CP-SNARK, which is adapted from [10].

Definition 2.4 (CP-SNARK [10])**.** Let $\mathtt{RGen}$ be a relation generator for relations R over $\mathcal{D}_\mathtt{u} \times \mathcal{D}_w \times \mathcal{D}_\omega$ such that $\mathcal{D}_w$ splits over ℓ arbitrary domains $(\mathcal{D}_1 \times \cdots \times \mathcal{D}_\ell)$ for some arity parameter $\ell \geq 1$. Let $\mathtt{Com}$ =$(\mathtt{Setup},\mathtt{Commit},\mathtt{Open})$ be a commitment scheme (as per Definition 2.3) whose input space $\mathcal{D}$ is such that $\mathcal{D}_i \subset \mathcal{D}$ for all $i \in [\ell]$. A commit and prove SNARK for $\mathtt{Com}$ and $\mathtt{RGen}$ is a zk-SNARK for a family of relations $\mathtt{RGen}^{\mathtt{Com}}$ such that:

- every $R^{\mathtt{Com}} \in \mathtt{RGen}^{\mathtt{Com}}$ is represented by a pair $(\mathtt{ck},R)$ where $\mathtt{ck} \in \mathtt{Com.Setup}(1^\lambda)$ and $R \in \mathtt{RGen}$;

- R^{Com} is over pairs $(\mathtt{u}, \mathtt{w})$ where the statement is $\mathtt{u} := (\mathtt{u}, (c_j)_{j\in[\ell]}) \in \mathcal{D}_x \times \mathcal{C}^\ell$, the witness is $\mathtt{w} := ((w_j)_{j\in[\ell]}, (o_j)_{j\in[\ell]}, \omega) \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_\ell \times \mathcal{O}^\ell \times \mathcal{D}_\omega$, and the relation R^{Com} holds iff

$$\bigwedge_{j\in[\ell]} \mathtt{Open}(\mathtt{ck}, c_j, w_j, o_j) = 1 \ \wedge \ R(\mathtt{u}, (w_j)_{j\in[\ell]}, \omega) = 1.$$

Furthermore, when we say that the CP-SNARK is knowledge-sound for a relation generator $\mathtt{RGen}$, we mean it is a knowledge-sound SNARK for the relation generator $\mathtt{RGen}^{\mathsf{Com}}$ that runs $\mathtt{ck} \leftarrow \mathtt{Com.Setup}(1^\lambda)$ and $(R, \mathtt{aux}_R) \leftarrow \mathtt{RGen}(1^\lambda)$, and returns $((\mathtt{ck}, R), \mathtt{aux}_R)$.

A CP-SNARK is a triple of algorithms $CP = (\mathtt{Setup}, \mathtt{Prove}, \mathtt{Verify})$:

- $\mathtt{crs} := (\mathtt{pk}, \mathtt{vk}) \leftarrow \mathtt{Setup}(\mathtt{ck}, R, \mathtt{aux}_R)$ generates a common reference string $\mathtt{crs}$ consisting of a proving key $\mathtt{pk}$ and a verification key $\mathtt{vk}$.
- $\pi \leftarrow \mathtt{Prove}(\mathtt{pk}, R, \mathtt{aux}_R, \mathtt{u}, (c_j)_{j\in[\ell]}, (w_j)_{j\in[\ell]}, (o_j)_{j\in[\ell]}, \omega)$ outputs the proof.
- $b \leftarrow \mathtt{Verify}(\mathtt{vk}, R, \mathtt{aux}_R, \mathtt{u}, (c_j)_{j\in[\ell]}, \pi)$ outputs a bit b indicating reject ($b = 0$) or accept $b = 1$.

2.4 Polynomial Commitments

We adapt the definition from [19], and add zero-knowledge. A polynomial commitment scheme for multivariate polynomials is a tuple of four (interactive) algorithms $\mathrm{PC} = (\mathtt{Gen}, \mathtt{Commit}, \mathtt{Open}, \mathtt{Eval})$:

- $\mathtt{pp} \leftarrow \mathtt{Gen}(1^\lambda, \mu)$: takes as input μ (the number of variables in a multilinear polynomial); produces public parameters $\mathtt{pp}$.
- $\mathcal{R} \leftarrow \mathtt{Commit}(\mathtt{pp}, \phi; r)$: takes as input a μ-variate polynomial ϕ over a finite field $\mathbb{F}$ and randomization r; produces a commitment $\mathcal{R}$. We omit r if it is sampled uniformly at random.
- $b \leftarrow \mathtt{Open}(\mathtt{pp}, \mathcal{R}, \phi, \mathtt{oh})$: given opening hint $\mathtt{oh}$, verifies the opening of commitment $\mathcal{R}$ to the μ-variate polynomial ϕ; outputs $b \in \{0, 1\}$.
- $b \leftarrow \mathtt{Eval}(\mathtt{pp}, \mathcal{R}, \vec{x}, y, \mu, \phi)$ is an interactive algorithm between a PPT prover $\mathcal{P}$ and verifier $\mathcal{V}$. Both $\mathcal{V}$ and $\mathcal{P}$ know a commitment $\mathcal{R}$, the number of variables μ, the evaluation point $\vec{x}$ and claimed evaluation y. $\mathcal{P}$ additionally knows a μ-variate polynomial ϕ. $\mathcal{P}$ attempts to convince $\mathcal{V}$ that $\phi(\vec{x}) = y$. At the end of the protocol, $\mathcal{V}$ outputs $b \in \{0, 1\}$.

Definition 2.5. A tuple of four (interactive) algorithms $(\mathtt{Gen}, \mathtt{Commit}, \mathtt{Open}, \mathtt{Eval})$ is a *zero-knowledge extractable polynomial commitment scheme* for μ-variate polynomials over a finite field $\mathbb{F}$ if the following conditions hold.

Completeness. For any μ-variate polynomial ϕ and all randomness r,

$$\Pr\left[\begin{array}{c} \mathtt{pp} \leftarrow \mathtt{Gen}(1^\lambda, \mu), \mathcal{R} \leftarrow \mathtt{Commit}(\mathtt{pp}, \phi) : \\ \mathtt{Eval}(\mathtt{pp}, \mathcal{R}, \vec{x}, y, \mu, \phi) = 1 \wedge y = \phi(\vec{x}) \end{array}\right] = 1.$$

Binding. For any PPT adversary $\mathcal{A}$, size parameter $\mu \geq 1$,

$$\Pr\left[\begin{array}{c} \mathtt{pp} \leftarrow \mathtt{Gen}(1^\lambda, \mu),\ (\mathcal{R}, \phi, \phi', \mathtt{oh}, \mathtt{oh}') = \mathcal{A}(\mathtt{pp}), \\ b_0 \leftarrow \mathtt{Open}(\mathtt{pp}, \mathcal{R}, \phi, \mathtt{oh}),\ b_1 \leftarrow \mathtt{Open}(\mathtt{pp}, \mathcal{R}, \phi', \mathtt{oh}') : \\ b_0 = b_1 = 1 \wedge \phi \neq \phi' \end{array}\right] \approx_\lambda 0$$

Knowledge Soundness. For any PPT adversary $\mathcal{A}$, size parameter $\mu \geq 1$, there exists a PPT extractor $\mathcal{E}$ with full access to $\mathcal{A}$ such that:

$$\Pr\left[\begin{array}{c} \mathtt{pp} \leftarrow \mathtt{Gen}(1^\lambda, \mu), (\mathcal{R}, \vec{x}, y) \leftarrow \mathcal{A}(\mathtt{pp}), 1 = \langle \mathcal{A}, \mathcal{V}_{\mathtt{Eval}} \rangle (\mathtt{pp}, \mathcal{R}, \vec{x}, y, \mu, \phi), \\ (\phi, \mathtt{oh}) \leftarrow \mathcal{E}^{\mathcal{A}}(\mathtt{pp}, (\mathcal{R}, \vec{x}, y)) : \phi(\vec{x}) \neq y \vee \mathtt{Open}(\mathtt{pp}, \mathcal{R}, \phi, \mathtt{oh}) \neq 1 \end{array}\right] \approx_\lambda 0$$

Zero-knowledge. For any μ-variate polynomial ϕ and any PPT adversary $\mathcal{A}$, there exists a simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$ for which it holds that

$$\Pr\left[\text{Real}_{\mathcal{A},\phi}(\texttt{pp})\right] \approx_\lambda \Pr\left[\text{Ideal}_{\mathcal{A},\phi,\mathcal{S}^{\mathcal{A}}}(\texttt{pp})\right],$$

where the experiments Real and Ideal are defined as follows:

$\text{Real}_{\mathcal{A},\prec}(\texttt{pp})$:
1. $\mathcal{R} \leftarrow \texttt{Commit}(\texttt{pp}, \phi)$
2. $\vec{x} \leftarrow \mathcal{A}(\mathcal{R}, \texttt{pp})$
3. $y \leftarrow \phi(\vec{x})$
4. $b \leftarrow \langle \mathcal{P}_{\texttt{Eval}}, \mathcal{A} \rangle(\texttt{pp}, \mathcal{R}, \vec{x}, y, \mu, \phi)$
5. Output b

$\text{Ideal}_{\mathcal{A},\prec,\mathcal{S}^{\mathcal{A}}}(\texttt{pp})$:
1. $\mathcal{R} \leftarrow \mathcal{S}_0(1^\lambda, \texttt{pp})$
2. $\vec{x} \leftarrow \mathcal{A}(\mathcal{R}, \texttt{pp})$
3. $y \leftarrow \phi(\vec{x})$
4. $b \leftarrow \langle \mathcal{S}, \mathcal{A} \rangle(\texttt{pp}, \mathcal{R}, \vec{x}, y, \mu)$
5. Output b

Here $\langle \mathcal{A}, \mathcal{B} \rangle$ indicates an interaction between parties $\mathcal{A}$ and $\mathcal{B}$, and $\mathcal{P}_{\texttt{Eval}}$ a prover running the `Eval` protocol.

Note that we make a distinction between the commitment randomization and the opening hint, because the prover may not generate the Merkle tree entirely honestly and know, say, all but a single opening. The scheme in this case is still binding, since ϕ is still the single closest codeword.

3 Orion and its Unsoundness

The full protocol for the Orion polynomial commitment is given in Figure 1, and the CP-SNARK statement is given in Figure 2. A CP-SNARK is used in the construction with the intention of making the prover commit to all values before the verifier provides the challenges. Unfortunately, there exists an issue with the challenge order. The row index set I is provided at step 6 in Figure 1, which is after the CP-SNARK commitment at step 4. However, the column challenge set J is provided *before* this commitment. This gives the prover the opportunity to choose the CP-SNARK witness based on J. For the columns of C this is not an issue, as these are already bound by the Merkle commitment $\mathcal{R}$ to D. However, the vectors $\vec{y}_\gamma$, $\vec{y}_1$, $\vec{r}_1$ and $\vec{r}_1$ are all committed to only after the verifier has sent their challenge set, when the prover executes the commitment phase of the CP-SNARK. Although their encodings $\vec{c}_\gamma$ and $\vec{c}_1$ are committed to beforehand, we will see that the prover can nonetheless cheat. In fact, it turns out that this enables the prover to open any committed polynomial in $\mathcal{R}$ to any point y. The commitment does not need to be forged specifically with this goal—any pre-existing commitment can be opened at any point $\vec{x}$ to any desired evaluation y.

Suppose the prover has some commitment $\mathcal{R}$ to a polynomial. In steps 1-4 of the `Eval` phase, $\mathcal{P}$ follows the protocol honestly, except for the fact that $\mathcal{P}$ sends the desired evaluation y in place of the honest evaluation corresponding to $\phi(x)$. At step 5, $\mathcal{P}$ needs to provide the CP-SNARK witness. Here the prover is able to forge a $\vec{y}_1$ such that for any I sent by the verifier in step 6, the prover will be able to complete the outer SNARK proof. This is sufficient for the verifier to accept this invalid evaluation proof.

To see that this is always possible, first note that the outer SNARK is the only place where $\vec{y}_1$ is validated. The vectors are also not committed to until the commit phase of the CP-SNARK in step 5. The CP-SNARK procedure constructs $\vec{c}_1$ directly from $\vec{y}_1$ and $\vec{r}_1$, so the goal of the prover is to find $\vec{y}_1{}'$ and $\vec{r}_1'$ such that together with the derived $\vec{c}_1{}' = \text{E}_\text{C}(\vec{y}_1{}') + \vec{r}_1' \mid\mid \vec{r}_1'$, the following conditions hold:

(1) $\forall j \in J: \ \vec{c}_{1j}{}' = \sum_{i \in [k]} \vec{x}_{0,i} C_{ij}$.

(2) $y = \sum_{j \in [k]} \vec{x}_{1j} \vec{y}_1{}'_j$

(3) $\forall j \in J: \ \mathcal{R}_{\vec{c}_{1j}}$ opens to $\vec{c}_{1j}{}'$

The first two conditions are checked in the CP-SNARK, while the last condition is checked directly by the verifier. Since $\vec{c}_1$ contained in the commitment $\mathcal{R}_{\vec{c}_1}$ has been honestly computed from C by

$\texttt{Commit}(\texttt{pp}, \phi; r)$**:** The prover $\mathcal{P}$ performs the following operations, where any randomness is deterministically derived from r:

1. Parse ϕ as a $k \times k$ matrix W of its coefficients.
2. For each $i \in [k]$, sample row $\vec{r_i} \in \mathbb{F}^n$ uniformly at random.
3. Compute $C \in \mathbb{F}^{k \times 2n}$ by encoding every row of W using the encoding function $\mathrm{E_C}$ and hiding with $\vec{r_i}$ as follows: $\forall i \in [k] : \; C_i = \mathrm{E_C}(W_i) + \vec{r_i} \;||\; \vec{r_i}$.
4. Compute $D \in \mathbb{F}^{n \times 2n}$ by encoding every column of C using $\mathrm{E_C}$.
5. Commit to columns: $\forall j \in [2n] : \; \mathcal{R}_j \leftarrow \texttt{Commit}_M(D_{.j})$.
6. Output final commitment $\mathcal{R} \leftarrow \texttt{Commit}_M(\mathcal{R}_1, \ldots, \mathcal{R}_{2n})$.

$\texttt{Eval}(\texttt{pp}, \mathcal{R}, X = \vec{x_0} \otimes \vec{x_1}, y, \mu, \phi)$**:** The prover and verifier execute the following protocol:

1. $\mathcal{V}$ sends a uniformly random vector $\vec{\gamma} \in \mathbb{F}^k$ to $\mathcal{P}$.
2. The prover computes

$$\begin{array}{llll} \vec{c_1} \leftarrow \sum_{i=1}^k \vec{x_0}_i C_i, & \vec{y_1} \leftarrow \sum_{i=1}^k \vec{x_0}_i W_i, & \vec{r_1} \leftarrow \sum_{i=1}^k \vec{x_0}_i \vec{r_i}, & \mathcal{R}_{\vec{c_1}} \leftarrow \texttt{Commit}_M(\vec{c_1}), \\ \vec{c_\gamma} \leftarrow \sum_{i=1}^k \vec{\gamma}_i C_i, & \vec{y_\gamma} \leftarrow \sum_{i=1}^k \vec{\gamma}_i W_i, & \vec{r_\gamma} \leftarrow \sum_{i=1}^k \vec{\gamma}_i \vec{r_i}, & \mathcal{R}_{\vec{c_\gamma}} \leftarrow \texttt{Commit}_M(\vec{c_\gamma}). \end{array}$$

3. $\mathcal{P}$ computes $y = \sum_{i=1}^k \vec{y_1}_j \vec{x_1}_j$ and sends $\mathcal{R}_{\vec{c_1}}$, $\mathcal{R}_{c_\gamma}$ and y to the verifier.
4. $\mathcal{V}$ samples and sends column indices $J \subset [2n]$ uniformly at random, under the condition that

$$|J| = 2t, \quad |J \cap [n]| = t, \quad |J \cap [n+1 \ldots 2n]| = t, \quad \neg \exists j : \; j, \, j+n \in \hat{I}.$$

5. The prover commits to the witness of the CP-SNARK (Figure 2), consisting of

$$\vec{y_1}, \quad \vec{y_\gamma}, \quad \forall_{(i,j) \in [k] \times J} : \; C_{ij}, \quad \vec{r_1}, \quad \vec{r_\gamma}.$$

6. The verifier samples row index set $I \subset [n]$, $|I| = t$ and sends it to the prover.
7. The prover computes the CP-SNARK argument π and sends it with

$$\begin{array}{ll} \forall j \in \hat{J} : & \texttt{Open}_M(\mathcal{R}_{\vec{c_1}}, j), \; \texttt{Open}_M(\mathcal{R}_{\vec{c_\gamma}}, j), \\ \forall j \in \hat{J} : & \texttt{Open}_M(\mathcal{R}, j), \quad \forall (i,j) \in I \times \hat{J} : \texttt{Open}_M(\mathcal{R}_j, i). \end{array}$$

8. $\mathcal{V}$ checks the proof π using the CP-SNARK verification procedure.
9. $\mathcal{V}$ checks the Merkle tree proofs of D_{ij} in $\mathcal{R}$ for all $(i,j) \in I \times J$.
10. $\mathcal{V}$ checks the Merkle tree proofs of $\vec{c_1}_j$ and $\vec{c_\gamma}_j$ in $\mathcal{R}_{\vec{c_1}}$ and $\mathcal{R}_{\vec{c_\gamma}}$.

The verifier accepts if all checks are successful.

Figure 1: The full description of the Orion polynomial commitment protocol.

the prover, conditions (1) and (3) can in fact be reduced to the single condition that $\forall j \in J : \; \vec{c_1}'_j = \vec{c_1}_j$, where $\vec{c_1}'$ is the vector derived from $\vec{y_1}$ and $\vec{r_1}$ in the outer SNARK. The protocol can be broken as

Committed witness: $\vec{y_1}, \vec{y_\gamma}, \vec{r_1}, \vec{r_\gamma}, \forall j \in J : C_{\cdot j}$.

Public input:

- $\vec{\gamma}, x_0, x_1, y, I, J$
- $\forall j \in J : \vec{c_1}_j, \vec{c_\gamma}_j$
- $\forall (i,j) \in I \times J : D_{ij}$

Proof circuit: The CP-SNARK circuit performs the following checks and computations:

- Build encodings of row vector combinations:
  - $\vec{c_\gamma} \leftarrow \mathrm{E_C}(\vec{y_\gamma}) + \vec{r_\gamma} \mid\mid \vec{r_\gamma}$.
  - $\vec{c_1} \leftarrow \mathrm{E_C}(\vec{y_1}) + \vec{r_1} \mid\mid \vec{r_1}$.
  - Check for $j \in J$ that $\vec{c_\gamma}_j$ and $\vec{c_1}$ match the input.
- Check that these are valid linear combinations:
  - Check $\forall j \in J : \vec{c_\gamma}_j \stackrel{?}{=} \sum_{i \in [k]} \vec{\gamma}_i C_{ij}$.
  - Check $\forall j \in J : \vec{c_1}_j \stackrel{?}{=} \sum_{i \in [k]} \vec{x_0}_i C_{ij}$.
- Encode columns and compare:
  - $\forall j \in J : D_{\cdot j} = \mathrm{E_C}(C_{\cdot j})$.
  - Check for $(i,j) \in I \times J$ that D_{ij} matches the input.
- Check claimed evaluation:
  - Check $y \stackrel{?}{=} \sum_{j \in [k]} \vec{x_1}_j \vec{y_1}_j$.

Figure 2: The procedure for the CP-SNARK. The committed witness contains the values that are part of the CP-SNARK commitment. Any other witness values arise only from the intermediate values due to the arithmetization.

follows: we first select any vector $\vec{y_1}$ that satisfies condition (2).[4] Then we need that

$$\begin{aligned} &\forall j \in J \cap 1 \ldots n : && \vec{r_1}'_j = \mathrm{E_C}(\vec{y_1})_j - \mathrm{E_C}(\vec{y_1}')_j + \vec{r_1}_j, \\ &\forall j \in J \cap n+1 \ldots 2n : && \vec{r_1}'_j = \vec{r_1}_j. \end{aligned}$$

To retain zero-knowledge, the challenge space is restricted so that these two sets are precisely disjoint. As such, we can just choose any $\vec{r_1}$ that satisfies these conditions.

One might then assume that the unsoundness is due to zero-knowledge, but in fact it is also possible to break (knowledge) soundness for the non-zero-knowledge version of the protocol. Even if we omit all $\vec{r_i}$ so that $C_i = \mathrm{E_C}(\vec{y_i})$ and J is just sampled uniformly random from $[n]$, the protocol can still be broken. Take the generator matrix G of $\mathrm{E_C}$. The condition $\vec{c_\gamma}'_j = \vec{c_\gamma}_j$ for $j \in J$ can be expressed through defining the matrix $B_{ij} \equiv \mathbb{1}\{J_i = j\} \in \mathbb{F}^{t \times n}$, where J_i denotes the ith element of the set J in some order. B can be interpreted as selecting t entries in J. Condition (2) can be appended as a column to the resulting matrix to obtain the linear system

$$\begin{bmatrix} \vec{c_1}_{(J_1)} \\ \vec{c_1}_{(J_2)} \\ \ldots \\ \vec{c_1}_{(J_t)} \\ y \end{bmatrix} = \begin{bmatrix} BG^T \\ \vec{x_1} \end{bmatrix} \begin{bmatrix} \vec{y_1}'_1 \\ \vec{y_1}'_2 \\ \ldots \\ \vec{y_1}'_k \end{bmatrix}. \tag{1}$$

[4] The only situation in which this is not possible, is if we structurally restrict ϕ to have a zero constant term, and we evaluate at $\vec{x} = \vec{0}$. Such an evaluation can only be 0 and as such a verifier cannot be convinced otherwise.

There are two reasons why there might not exist a solution. First, it is possible that there is no other codeword equal to $\vec{c_1}$ at J. Second, it may be possible that all candidate codewords have messages that are orthogonal to $\vec{x_1}$. To address both these possibilities: consider the vector space $A = \{\vec{m} \in \mathbb{F}^k : (BG^T\vec{m}) = \vec{0}\}$.

Lemma 1. Let $0 < l = k - t$. Then $\dim(A) \geq l$.

Proof. From the rank-nullity theorem $k = \text{rank}(BG^T) + \dim(\ker(BG^T))$, and so

$$\dim(A) = \dim(\ker(BG^T)) = k - \text{rank}(BG^T) \geq l.$$

□

It follows that the first possibility is never applicable if $k > t$. This is a very weak assumption corresponding exactly to those situations where the inner SNARK does useful work: if $k \leq t$, the CP-SNARK does more work than it would do by evaluating the polynomial directly. As such, in this situation one should always opt to commit to the polynomial using the CP-SNARK commitment, and evaluate the polynomial in its circuit.

For the second condition, the system is solvable if and only if there exists $\vec{m} \in A$, $\vec{m} \notin \ker(\vec{x_1})$. Here, $\ker(\vec{x_1})$ has dimension $k - 1$ by the rank-nullity theorem (or k if $\vec{x_1} = \vec{0}$, but then $y = 0$ would be the only possible evaluation regardless of ϕ). As k increases, so does $\dim(A) \geq k - t$, and therefore it becomes more likely that there exists such a message. We have the following lemma:

Lemma 2. Let $\vec{x_1} \in_R \mathbb{F}_p^{1\times k}$ and $k > t$. Then Eq. (1) has a solution except with negligible probability.

Proof. We know that the solutions to Eq. (1) are a subset of $\vec{y_1} + \vec{m}$ for $\vec{m} \in A$ and $\vec{y_1} = \sum_{i\in[k]} \vec{x_{0i}} W_i$ for the original witness. If there exists one message $\vec{m} \in A$ such that $\sum_{j\in[n]} \vec{x_{1j}}\vec{m_j} = \alpha \neq 0$, then $\vec{y_1} + \left(y - \sum_{j\in[n]} \vec{x_{1j}}\vec{y_{1j}}\right) \alpha^{-1}\vec{m}$ will be a solution to the system. Consider one message $\vec{m'} \in A$ and index j' where $\vec{m}'_{j'} \neq 0$. Then the probability that there exists no solution is given by

$$\begin{aligned}\Pr\left[\forall \vec{m} \in A : \left\langle \vec{x_{1j}}\vec{m_j} \right\rangle = 0\right] &\leq \Pr\left[\sum_{j\in[n]} \vec{x_{1j}}\vec{m'}_j = 0\right] \\ &= \Pr\left[\vec{x_{1j'}} = -\vec{m'}_{j'}^{-1} \sum_{j\in[n]\setminus\{j'\}} \vec{x_{1j}}\vec{m'}\right] = \frac{1}{|\mathbb{F}|}\end{aligned}$$

□

Similarly, the case where $\vec{x_{1j}} = x^j$ for some uniformly random $x \in \mathbb{F}$ has negligible failure probability by the Schwartz-Zippel lemma. Unfortunately it is not possible to give a general success probability without knowing a distribution of the evaluation point $\vec{x_1}$. It is likely that the adversary succeeds with high probability for evaluation points that are not adversarially chosen. Note that even a small probability is already sufficient in the non-interactive setting, since the prover can simply try for $\text{poly}(\lambda)$ different challenge sets J.

3.1 Attack Implementation

Since Orion uses a Generalized Spielman code, the encoding algorithm can be expressed much more efficiently than through its generator matrix G, both computationally as well as in terms of memory usage. Fortunately, we do not need to construct G explicitly for the attack either.

The encoding algorithm $\text{E}_\text{C}{}^{(d)}(\vec{x})$ consists of three parts. We use d to indicate the depth, since the algorithm is recursive. In the base case, $d = 0$ and we just copy the message directly. Otherwise, $\vec{m_1} = \vec{x}A^{(d)}$ is computed for some expander graph $A^{(d)}$. We then compute $\vec{c_1} = \text{E}_\text{C}{}^{(d-1)}(\vec{m_1})$ through a recursive call. Lastly $\vec{c_2} = \vec{c_1}A^{(d)'}$ is computed for another expander graph. The final codeword is $\vec{x}||\vec{c_1}||\vec{c_2}$.

Instead of computing G from this process through encoding basis vectors for the message space, we apply G^T to B from the right. Note that B has size $t \times n$, much smaller than G. We start by

inverting $A^{(d)'}$ to the right-most columns of B. We can view the result as a matrix that maps $\vec{x}||\vec{c_1}$ to the output at the chosen indices. The next step is a recursive call, creating a matrix that takes $\vec{x}||\vec{m}_1$ as a right input. Lastly, we invert $A^{(d)}$ and arrive at the final matrix that maps $\vec{x}$ to the correct evaluation of $\mathrm{E_C}$ at the selected indices.

While every lossless expander graph may be sparse, the resulting generator matrix is not. Therefore, this method is much more memory efficient: B has only t rows and starts with n columns, and BG^T will have dimension $t \times k$. During this process, the size of B has only shrunk from its original size. We therefore use only $O(|B|)$ memory. Computing is also efficient: inverting expander graphs is as efficient as applying them.

After having computed the resulting matrix BA, a valid codeword can be found by computing the pseudoinverse $Q^\dagger$ of $Q = \begin{bmatrix} BA \\ \vec{x_1} \end{bmatrix}$. We then find a solution to the system by setting

$$\vec{y_\gamma}' = Q^\dagger \begin{bmatrix} \vec{c_\gamma}_{(J_1)} & \vec{c_\gamma}_{(J_2)} & \dots & \vec{c_\gamma}_{(J_t)} & y \end{bmatrix}^T.$$

We provide an implementation of the above described algorithm, which demonstrates the efficiency and practicality of the attack. [5]

4 Restoring Soundness of Orion

The order in which challenges and commitments must be sent is naturally constrained by the protocol. First, we do not want to commit to C entirely and so the column set J needs to be known at this time. Second, to check that C is consistent with the commitment to D, we should sample I only after committing to these columns. Third, the vector $\vec{y_\gamma}$ needs to be compared to C, but here the column vector J must be sampled afterward to ensure $\vec{y_\gamma}$ cannot be chosen to open correctly at only these points.

4.1 One More Round of Commitments

One way to reconcile this, is by committing to $\vec{y_\gamma}$ before $\hat{I}$ is sampled, and opening this commitment in the SNARK. This can be generically achieved by creating a Merkle commitment and opening it inside the CP-SNARK. This is undesirable however, since hash-computations are expensive to perform inside the SNARK circuit. Alternatively, we could switch to a CP-SNARK that supports multiple rounds of commitments: the vectors $\vec{y_1}$, $\vec{y_\gamma}$, $\vec{r_1}$ and $\vec{r_\gamma}$ would be committed to first, and when J is sampled, then the prover commits to $\vec{c_1}_{,j}$ as the second part of the witness.

Done naively, this would incur significant overhead on the CP-SNARK, since the committed witness now uses two polynomial commitments of half the size instead of one. If the verifier time and proof size incurred by this step are polylogarithmic, these would almost double, with the opening of polynomial commitments representing the majority of the total work. This naive approach can be improved upon through the use of batching. One option is to use FRI [4], which can be efficiently batched with minimal overhead.

4.2 Sampling a Different Column Set

In this paper we will instead focus on regaining soundness of the original construction, where we do not want to put additional requirements on the outer SNARK. As we will see, we can fix the protocol without requiring that the CP-SNARK supports efficient batching or even multiple rounds of commitments at all. The only cost associated with our approach is that the verifier samples a new column set $\hat{J}$. Additionally, we have to switch the zero-knowledge scheme, but this will actually improve verifier time and proof size, while keeping the prover time comparable.

If it were possible to sample such a second, independent column set $\hat{J}$, this set could be used for the columns $C_{\cdot j}$, while the other parts of the protocol could depend on a different J that is only provided by $\mathcal{V}$ after $\mathcal{P}$ has already committed to the CP-SNARK witness. This way, we can have the

[5] An implementation of this attack can be found at https://github.com/ThomasdenH/orion-attack.

column set $\hat{J}$ available when we commit to selected columns of C, while not allowing the prover to select $\vec{y_\gamma}$ such that it opens correctly precisely at J, which would be sampled later.

This approach has several advantages: first, we do not increase the witness size, just the statement size of the CP-SNARK. We have one additional challenge, which does not need to be sent with the proof. We don't create any new Merkle commitments or open existing commitments at additional points at all.

Unfortunately, this approach still has an issue. To retain zero-knowledge, the second column set J cannot be sampled independently of $\hat{J}$. It is necessary that $\neg\exists j : \ j, j+n \in J \cup \hat{J}$ or part of the witness could be learned. But this requirement implies that before committing to the CP-SNARK witness, $\mathcal{P}$ knows t entries that will never be queried by the verifier in J.

4.3 Using Zero-knowledge Linear Codes

To remedy this issue, the zero-knowledge scheme is changed to use a randomized polynomial instead of a vector to mask the witness. This has the desirable property that entries are distributed uniformly at random as long as the degree is at least $\left|J \cup \hat{J}\right| - 1$. This construction is an instance of a q-query uniform linear code, which in turn is a specific case of a zero-knowledge linear code. In fact, this general transformation of any code to a perfectly zero-knowledge code may be of independent interest. It retains the same relative minimum distance instead of reducing it by half as in the vector masking approach, while retaining linear encodability. Crucial is the observation that we require $\mathrm{E}_{\mathrm{C,ZK}}$ to be linearly encodable in k, not in q.

Lemma 3. Let E_C be a linear code $[n, k, d]$ and let $q < 2n - 4d$. Then ${\mathrm{E}_{\mathrm{C,ZK}}}^q$, defined for $(\vec{y}, \vec{r}) \in \mathbb{F}^k \times \mathbb{F}^q$ as

$$\forall i \in [2n] : \ {\mathrm{E}_{\mathrm{C,ZK}}}^q(\vec{y}; \vec{r})_i = (\mathrm{E}_\mathrm{C}(\vec{y}) \mid\mid \mathrm{E}_\mathrm{C}(\vec{y}))_i + \sum_{j=0}^{q-1} \vec{r}_j i^j,$$

is the randomized linear code $[2n, k+q, 2d]$. Furthermore, if E_C is encodable in $O(n)$ operations, then ${\mathrm{E}_{\mathrm{C,ZK}}}^q$ is encodable in $O\left(q(n+q)\right)$ operations.

Proof. The fact that ${\mathrm{E}_{\mathrm{C,ZK}}}^q$ is a linear code follows directly from the definition.

To see that the minimum distance is $2d$, note that the right-hand side is just $\mathrm{RS}_{[2n,q,2n-q+1]}$. For $(\vec{y} \mid\mid \vec{r}) \neq \vec{0}$ it holds that

$$\begin{aligned} w\left(\mathrm{E}_{\mathrm{C,ZK}}(\vec{y}; \vec{r})\right) &\geq \left|2w(\mathrm{E}_\mathrm{C}(\vec{y})) - \mathrm{RS}_{[2n,q,2n-q+1]}(\vec{r})\right| \\ &\geq \min(2d, 2n-q+1, 2n-2d-q-1) \geq 2d \end{aligned}$$

by the restriction that $q < 2n - 4d$.

To see that it is q-query uniform, note that for any message $\vec{y}$ and any q queried locations (i_j) and corresponding values (e_j) with $j \in [q]$, there is exactly one $\vec{r}$ such that $\mathrm{E}_{\mathrm{C,ZK}}(\vec{y}; \vec{r})_{i_j} = e_j$ for all $j \in [q]$, since q points uniquely determine all coefficients of a degree $q-1$ polynomial. This means that for $\vec{r}$ uniformly random, any q queries are also uniformly random.

Lastly, encoding a codeword $\mathrm{E}_{\mathrm{C,ZK}}(\vec{y}; \vec{r})$ takes $O(k)$ operations for encoding E_C, and then $q(n + q + d - 1) = O(q(n+q))$ operations for computing the polynomial. □

Using $\mathrm{E}_{\mathrm{C,ZK}}$ as the row encoding means that the challenge space can be simplified: $\mathcal{V}$ can simply select $\hat{J}$ and J independently uniformly at random. Additionally, for the same message space, our randomized linear code has twice the minimum distance as the code $\mathrm{E}_\mathrm{C}(\vec{y}) + \vec{r} \mid\mid \vec{r}$. This means that we get the same security level for smaller proof sizes.

Interestingly, only the prover ever needs to run $\mathrm{E}_{\mathrm{C,ZK}}$. This means that it might be possible to encode randomness in the (secret) generator matrix instead of having a random contribution, potentially shrinking codeword size and thereby proof size. One potential issue is that it must be efficiently verifiable that the code has a good minimum relative distance. It may be possible that an existing randomly sampled linear-time encodable code already has the desired properties. This leaves an interesting research question with potential to optimize this zk-SNARK further.

4.4 Using a Logarithmically Sized Challenge

Due to the use of the challenge $\vec{\gamma} \in \mathbb{F}^k$, the verifier has to sample a random vector of size $O(\sqrt{N})$. Since it is not included in the proof, this does not impact its size. The verifier complexity on the other hand *is* affected: $\mathcal{V}$ needs to provide $\vec{\gamma}$ as the public statement to the CP-SNARK, making the verifier complexity $O(\sqrt{N})$ as well.

This issue is addressed by [14], which improves on Brakedown by letting the verifier sample challenge $\vec{\eta} \in \mathbb{F}^{\log(k)}$, which is then used to compute the vector $\bigotimes_{i=1}^{\log(k)}(1 - \vec{\eta}_i, \vec{\eta}_i)$. In our case, this transformation can be done inside the outer SNARK circuit, which enables succinctness in both proof size as well as verifier time. The authors show that this incurs only a logarithmic soundness loss, which may be resolved by using a slightly larger field size. We adapt this challenge structure to our polynomial commitment scheme.

4.5 Scorpius: A New Polynomial Commitment Scheme

Our new polynomial commitment scheme can be found in Figure 3, and the corresponding CP-SNARK relation in Figure 4.

Theorem 4. *Let* $\mathrm{E_C}$ *be a linear code* $[n, k, d]$ *with* $\left(1 - \frac{d}{3n}\right)^t \leq negl(\lambda)$ *and let* $\mathrm{E_{C,ZK}}^{2t}$ *be the* $2t$*-query uniform linear code* $[\tilde{n}, \tilde{k}, \tilde{d}]$ *obtained from* $\mathrm{E_C}$ *through Lemma 3. Then the protocol in Figure 3, using a complete, knowledge-sound, zero-knowledge CP-SNARK to instantiate the outer SNARK relation in Figure 4, is a zero-knowledge polynomial commitment scheme.*

We provide a proof in Section 5.

In addition to our new polynomial commitment scheme, we also make an improvement to the simulator. The one currently used in [31] does not follow its own definition, since it requires knowledge of the evaluation point at the time of committing. This is likely an issue with the current simulator and proof as opposed to the scheme itself—it seems possible to describe a correct simulator. This difference has practical relevance too, however. The evaluation point may realistically not be known beforehand, even by the verifier, and in this case we still want zero-knowledge. The definition therefore also concerns a more realistic scenario than what is currently achieved. We therefore describe a new simulator that can open $\mathcal{R}$ to any evaluation point, anywhere.

4.6 Efficiency

Our construction achieves prover complexity of $O(N)$, and proof size and verifier complexity of $\log^2(N)$. Our solution also improves verifier efficiency as well as proof size, and is only slightly worse in terms of prover time.

First consider the impact on the CP-SNARK circuit. There, the only penalty in performance of our approach is the $2t \cdot 2n = O(\sqrt{N})$ multiplications to evaluate the randomized polynomial, instead of n additions, as well as the computation of the challenge $\vec{\gamma}$ from $\vec{\eta}$. Since we already perform t encodings in $O(\sqrt{N})$ operations (with a higher constant), the circuit does not grow significantly. Since the prover time is $O(\sqrt{N})$ and the proof size and verifier time are $O(\log^2(N))$, we only expect the prover to incur a noticeable (but still minor) overhead from the new CP-SNARK circuit, and the verifier time and proof size to not be noticeably impacted.

Outside the CP-SNARK, the prover additionally needs to do N constant size polynomial evaluations. We do not expect this to make the prover significantly less efficient either, since it already needs to perform $5k$ encodings of $\mathrm{E_C}$ in $O(N)$ time, as well as compute $2N$ hash functions for the Merkle commitment.

As we have seen before, for the verifier time and proof size, the impact of the new CP-SNARK circuit is tiny. At the same time, the minimum distance of the used code becomes larger due to the new randomization technique. This means that the number of opened columns and rows, t, more than halves, meaning that there are much fewer elements in the proof. The verifier time depends on verifying π and is otherwise linear in t. In addition, it now needs to compute and verify a challenge only of size $O(\log(N))$ instead of $O(\sqrt{N})$ and so it is expected to decrease significantly as well.

$\texttt{Commit}(\texttt{pp}, \phi; r)$: The prover $\mathcal{P}$ performs the following operations, where any randomness is deterministically derived from r:

1. Parse ϕ as a $k \times k$ matrix W of its coefficients.
2. Compute $C \in \mathbb{F}^{k \times \tilde{n}}$ by encoding every row of W using the encoding function $\mathrm{E}_{C,ZK}{}^{2t}(W_i; \vec{r}_i)$ for $\vec{r}_i \in_R \mathbb{F}^q$.
3. Compute $D \in \mathbb{F}^{n \times \tilde{n}}$ by encoding every column of C using E_C.
4. Commit to columns: $\forall j \in [\tilde{n}] : \mathcal{R}_j \leftarrow \texttt{Commit}_M(D_{\cdot j})$.
5. Output final commitment $\mathcal{R} \leftarrow \texttt{Commit}_M(\mathcal{R}_1, \ldots, \mathcal{R}_{\tilde{n}})$.

$\texttt{Eval}(\texttt{pp}, \mathcal{R}, X = \vec{x}_0 \otimes \vec{x}_1, y, \mu, \phi)$: The prover and verifier execute the following protocol:

1. $\mathcal{V}$ sends a uniformly random vector $\vec{\eta} \in \mathbb{F}^{\log(k)}$ to $\mathcal{P}$.
2. The prover computes $\vec{\gamma} = \bigotimes_{i=1}^{\log(k)} (1 - \vec{\eta}_i, \vec{\eta}_i)$ and

$$\begin{array}{llll} \vec{c_1} \leftarrow \sum_{i=1}^k \vec{x}_{0i} C_i, & \vec{y_1} \leftarrow \sum_{i=1}^k \vec{x}_{0i} W_i, & \vec{r_1} \leftarrow \sum_{i=1}^k \vec{x}_{0i} \vec{r}_i, & \mathcal{R}_{\vec{c_1}} \leftarrow \texttt{Commit}_M(\vec{c_1}), \\ \vec{c_\gamma} \leftarrow \sum_{i=1}^k \vec{\gamma}_i C_i, & \vec{y_\gamma} \leftarrow \sum_{i=1}^k \vec{\gamma}_i W_i, & \vec{r_\gamma} \leftarrow \sum_{i=1}^k \vec{\gamma}_i \vec{r}_i, & \mathcal{R}_{\vec{c_\gamma}} \leftarrow \texttt{Commit}_M(\vec{c_\gamma}). \end{array}$$

3. $\mathcal{P}$ computes $y = \sum_{i=1}^k \vec{y_1}_j \vec{x_1}_j$ and sends $\mathcal{R}_{\vec{c_1}}$, $\mathcal{R}_{c_\gamma}$ and y to the verifier.
4. $\mathcal{V}$ samples and sends column indices $\hat{J} \subset [\tilde{n}]$ uniformly at random.
5. The prover commits to the witness of the CP-SNARK (Figure 4), consisting of

$$\vec{y_1}, \quad \vec{y_\gamma}, \quad \vec{r_1}, \quad \vec{r_\gamma}, \quad \forall_{j \in \hat{J}} : C_{\cdot j}.$$

6. The verifier samples and sends row index set $I \subset [n]$, $|I| = t$, as well as a new column index set $J \subset [\tilde{n}]$, $|J| = t$, uniformly at random.
7. The prover computes the CP-SNARK argument π and sends

$$\begin{array}{ll} \forall j \in J : & \texttt{Open}_M(\mathcal{R}_{\vec{c_1}}, j), \texttt{Open}_M(\mathcal{R}_{\vec{c_\gamma}}, j), \\ \forall j \in \hat{J} : & \texttt{Open}_M(\mathcal{R}, j), \quad \forall (i,j) \in I \times \hat{J} : \texttt{Open}_M(\mathcal{R}_j, i). \end{array}$$

8. $\mathcal{V}$ checks the proof π using the CP-SNARK verification procedure.
9. $\mathcal{V}$ checks the Merkle tree proofs of D_{ij} in $\mathcal{R}$ for all $(i,j) \in I \times \hat{J}$.
10. $\mathcal{V}$ checks the Merkle tree proofs of $\vec{c_1}_j$ and $\vec{c_\gamma}_j$ in $\mathcal{R}_{\vec{c_1}}$ and $\mathcal{R}_{\vec{c_\gamma}}$.

The verifier accepts if all checks are successful.

Figure 3: The protocol for our new polynomial commitment scheme Scorpius.

4.7 Supporting Smaller Polynomials

Orion requires that $4k = n \geq 2t$ to be able to satisfy the constraints on the challenge space. For the relative minimum distance $\delta = 0.055$, we set $t = 4795$ in order that $\left(1 - \frac{\delta}{3}\right)^t \leq 2^{-\lambda}$.[6] Note that our code can be used for $n > \frac{t}{1-2\delta}$, or $k > \frac{t}{4(1-2\delta)} \approx 1347$ for the generalized Spielman code used in Orion. This means that we directly support polynomials of less than half the size.

As we have noted in Section 3 already, for polynomials where $k < t$ it would make sense to use

[6] Although Orion selects $t = 1568$, checking the linear combination gives rise to a success probability $\left(1 - \frac{d}{3 \cdot 2n}\right)^t$, implying a security level of $\lambda \approx 14$. For $\lambda = 128$, a value of $t = 9635$ would have been required instead.

Committed witness: $\vec{y_1}, \vec{y_\gamma}, \vec{r_1}, \vec{r_\gamma}, \forall j \in \hat{J} : C_{\cdot j}$.

Public input:

- $\vec{\eta}, \vec{x_0}, \vec{x_1}, y, \hat{J}, I, J$.
- $\forall j \in J : \vec{c_1}_j, \vec{c_\gamma}_j$.
- $\forall (i,j) \in I \times J : D_{ij}$.

Proof circuit: The CP-SNARK circuit performs the following checks and computations:

- Compute $\vec{\gamma} = \bigotimes_{i=0}^{\log(k)} (1 - \vec{\eta}_i, \vec{\eta}_i)$.
- Build encodings of row vector combinations:
  - $\vec{c_\gamma} \leftarrow \mathrm{E}_{\mathrm{C,ZK}}(\vec{y_\gamma}; \vec{r_\gamma})$
  - $\vec{c_1} \leftarrow \mathrm{E}_{\mathrm{C,ZK}}(\vec{y_1}; \vec{r_1})$
  - Check for $j \in J$ that $\vec{c_\gamma}_j$ and $\vec{c_1}_j$ match the input.
- Check that these are valid linear combinations:
  - Check $\forall j \in \hat{J} : \vec{c_\gamma}_j \stackrel{?}{=} \sum_{i \in [k]} \vec{\gamma}_i C_{ij}$.
  - Check $\forall j \in \hat{J} : \vec{c_1}_j \stackrel{?}{=} \sum_{i \in [k]} \vec{x_0}_i C_{ij}$.
- Encode columns:
  - $\forall j \in \hat{J} : D_{\cdot j} = \mathrm{E}_{\mathrm{C}}(C_{\cdot j})$.
  - Check for $(i,j) \in I \times \hat{J}$ that D_{ij} matches the input.
- Check claimed evaluation:
  - Check $y \stackrel{?}{=} \sum_{j \in [k]} \vec{x_1}_j \vec{y_1}_j$.

Figure 4: The procedure for the fixed CP-SNARK. The committed witness contains the values that are part of the CP-SNARK commitment. Any other witness values arise only from the intermediate values due to the arithmetization.

the CP-SNARK directly. There are however some additional ways of making the proof system more efficient even for small k. For one, as was already noted in [31], at some point it is faster to avoid using codes for the columns and instead open these directly. This removes the cost of encoding and shrinks the amount of data to be committed to. For each column, k elements are now opened instead of t, and so this approach is always better when $k < t$. Second, for the rows it is possible to switch to Reed-Solomon codes in place of $\mathrm{E}_{\mathrm{C,ZK}}$ in situations where k is small enough that the $O(n \log(n))$ cost is acceptable. This will allow for much lower values of t due to the larger relative minimum distance of the code. For example, for $k = 2000$ we can pick the code $\mathrm{RS}_{[7486,2000+2t,4847]}$ with $t = 365$ and for $k = 1000$ the code $\mathrm{RS}_{[9896,1000+2t,8585]}$ with $t = 156$ suffices. Since proof size and verifier time depend mostly on t, this will give a small and efficient proof system, although it comes at the cost of prover speed.

4.8 Open at Multiple Points

Since one of our goals is to support zero-knowledge, we cannot run the `Eval` procedure multiple times, even though this would be desirable in many protocols. Fortunately, the scheme is easily extended to open at multiple points by taking a random linear combination of all evaluation vectors. This protocol can be found in Figure 5.

Lemma 5. The protocol in Figure 5 is complete, binding and zero-knowledge. In addition, it is knowledge sound, i.e. for any adversary $\mathcal{A}$, size parameter $\mu \geq 1$, there exists a PPT extractor $\mathcal{E}$

with full access to $\mathcal{A}$ such that:

$$\Pr\left[\begin{array}{c} \mathtt{pp} \leftarrow \mathtt{Gen}(1^\lambda, \mu), (\mathtt{pp}, \mathcal{R}, \vec{x_{0,0}} \otimes \vec{x_{0,1}}, \ldots, \vec{x} y_0, \ldots, \vec{y_{m-1}}) \leftarrow \mathcal{A}(\mathtt{pp},) \\ 1 = \langle \mathcal{A}, \mathcal{V}_{\mathtt{Eval}} \rangle (\mathtt{pp}, \mathcal{R}, \vec{x}, y, \mu, \phi), (\phi, \mathtt{oh}) \leftarrow \mathcal{E}^{\mathcal{A}}(\mathtt{pp}, (\mathcal{R}, \vec{x}, y)) : \\ \phi(\vec{x_1}) \neq y_1 \vee \cdots \vee \phi(\vec{x_{m-1}}) \neq y_{m-1} \vee \mathtt{Open}(\mathtt{pp}, \mathcal{R}, \phi, \mathtt{oh}) \neq 1 \end{array}\right] \approx_\lambda 0.$$

We provide a proof in Section 5.

$\mathtt{Eval}_\mathrm{m}(\mathtt{pp}, \mathcal{R}, (X_0 = \vec{x_{1,0}} \otimes \vec{x_{1,1}}, \ldots, X_{m-1} = \vec{x_{m-1,0}} \otimes \vec{x_{m-1,1}}), (y_0, \ldots, y_{m-1}), \mu, \phi)$:
The prover and verifier execute the following protocol:

- $\mathcal{V}$ selects a random vector $\vec{\alpha} \in \mathbb{F}^m$ and sends it to the prover.
- $\mathcal{P}$ and $\mathcal{V}$ compute $X' = \sum_{i=0}^{m-1} \vec{\alpha}_i X_i$.
- $\mathcal{P}$ and $\mathcal{V}$ compute $y' = \sum_{i=0}^{m-1} \vec{\alpha}_i y_i$.
- $\mathcal{P}$ and $\mathcal{V}$ run the protocol $\mathtt{Eval}(\mathtt{pp}, \mathcal{R}, X', y', \mu, \phi)$.

Figure 5: A procedure to open the polynomial commitment at multiple points.

4.9 On (Weak) Simulation Extractability

Simulation Extractability (SE) is a strengthened knowledge-soundness property, which guarantees that the witness is extractable even if the adversary has access to simulated proofs for arbitrary adaptively chosen statements. This is a property often desirable in applications and intensively studied recently either generically for PIOPs [17, 22, 15] or for various concrete proof systems (e.g, [13, 23]). At least two variants can be distinguished: strong SE, where the adversary wins by providing a proof that was not simulated before, or weak SE, where the adversary wins by providing a proof for statement that was not simulated before.

Unfortunately, polynomial commitment schemes based on [7], such as those used in Orion, Brakedown [19] and Ligero [1, 2] cannot provide (strong) SE. The consistency check relies on the fact that a linear combination of vectors far from all codewords, will also be far from any codeword. This means that the closest codeword is correct. However, it cannot be enforced that this vector (i.e. $\vec{c_\gamma}$ in Orion) is *equal* to a codeword. In fact, if the prover changes just a single index they have a decent probability of passing all verifier checks. This means the prover can do something that resembles rerandomization and thus breaking strong SE. Note that the message corresponding to the closest codeword cannot be changed, and therefore it is still possible that Orion satisfies weak SE. In fact, this seems likely if the outer CP-SNARK is at least weak SE: if we include the transcript in the outer CP-SNARK statement, we will never be able to re-use a simulated outer proof. Therefore, $\mathcal{P}$ is always forced to go through the verification procedure. Proving that the Orion PC is weak SE, or more importantly that the Orion zk-SNARK is weak SE, is an interesting question for future work.

5 Security

Orion uses the result [2, Lemma A.1] that if a single row of matrix C is far away from a codeword, then a random linear combination will also be far away with high probability. Or, conversely, that if a random linear combination is e-close to a codeword, then every row of the matrix is equal to a codeword except at at most e columns. To enable a logarithmic verifier however, we will instead use the following related lemma due to Diamond and Posen.

Lemma 6. [14, Theorem 3.1] Fix an arbitrary $[n, k, d]$-code $\mathrm{E_C}$, and a proximity parameter $e \in$

$\left\{0, \ldots, \left\lfloor \frac{d-1}{3} \right\rfloor \right\}$. If $C \in \mathbb{F}^{k \times n}$ satisfies

$$\Pr_{\vec{\eta} \in \mathbb{F}^{\log(k)}} \left[d\left(\bigotimes_{i=1}^{\log(k)} (1 - \vec{\eta}_i, \vec{\eta}_i) \cdot C, \mathrm{E}_C \right) \leq e \right] > 2\log(k) \frac{e+1}{|\mathbb{F}|},$$

then $d(C, {\mathrm{E}_C}^k) \leq e$.

Furthermore, like [14], we need two extraction strategies. First, we can read ROM-queries to open the Merkle-tree commitment $\mathcal{R}$ and obtain the randomization used in the commitment. Second, because we do not assume (and in fact do not have) an efficiently decodable linear code, we require a separate strategy to obtain the decodings. This is done through rewinding, in order to obtain k vectors $\vec{y_\gamma}$ for linearly independent $\vec{\gamma} = \bigotimes_{i=1}^{\log(k)} (1 - \vec{\eta}_i, \vec{\eta}_i)$. To make this more difficult, while the verifier is public-coin, its challenges may not be uniformly distributed when restricted to accepting proofs. To this end, we will use [14, Lemma 4.13], which states that these challenges will be linearly independent except with negligible probability, if this fact is taken into account.

We can now give the proof for Theorem 4.

Proof. (Theorem 4)

Completeness. The protocol in Figure 3 is perfectly complete by construction. The CP-SNARK is complete by assumption so we conclude that the polynomial commitment scheme as a whole is complete.

Binding. Define the algorithm $\mathtt{Open}(\mathtt{pp}, \mathcal{R}, \phi, \mathtt{oh})$ as follows: construct matrix $\tilde{D} \in \mathbb{F}^{n \times \tilde{n}}$ by filling in all entries that have valid Merkle tree openings contained in $\mathtt{oh}$, setting the others to 0. Compute C honestly from ϕ and the randomization vectors $(\vec{r}_i)_{i=1}^k$ (contained in $\mathtt{oh}$) and encode vertically to obtain D. Output 1 if, for more than $\tilde{n} - \frac{\tilde{d}}{3}$ columns j, it holds that $|\Delta(\tilde{D}_{\cdot j}, D_{\cdot j}) \cup M_j| < \frac{d}{2}$, where M_j denotes the indices in this column for which $\mathtt{oh}$ did not contain a Merkle tree opening. Output 0 otherwise.

Suppose there exists a PPT adversary $\mathcal{A}$ that can find ϕ, r, ϕ', r' such that $\phi \neq \phi'$ and both $\mathtt{Open}(\mathtt{pp}, \mathcal{R}, \phi, r) = 1$ and $\mathtt{Open}(\mathtt{pp}, \mathcal{R}, \phi', r')$. On the one hand, for the honestly computed encodings C and C', there must be a row i such that $d(C_i, C'_i) \geq \tilde{d}$. Thus, there must exist at least $\tilde{d}$ indices where the columns of C and C' differ and thus as many columns for which $d(D_{\cdot j}, D'_{\cdot j}) > d$.

Consider then the Merkle tree openings $\tilde{D}$ and $\tilde{D}'$. Since a PPT adversary cannot open the Merkle tree anywhere at two distinct points, it must hold that any index in $\Delta(D_{\cdot j}, D'_{\cdot j})$ must be in either $\Delta(D_{\cdot j}, \tilde{D}_{\cdot j}) \cup M_j$ or $\Delta(D'_{\cdot j}, \tilde{D}'_{\cdot j}) \cup M'_j$, i.e. it must differ from the opening or be absent in (at least) one of the two claimed openings. Since we know that for less than $\frac{\tilde{d}}{3}$ columns j it holds that $|\Delta(D_{\cdot j}, D'_{\cdot j}) \cup M_j| \geq \frac{d}{2}$, there exist at least one column j for which $|\Delta(D_{\cdot j}, D'_{\cdot j}) \cup M_j| < \frac{d}{2}$ and $|\Delta(\tilde{D}_{\cdot j}, \tilde{D}'_{\cdot j}) \cup \tilde{M}_j| < \frac{d}{2}$. Now the set $\Delta(D_{\cdot j}, D'_{\cdot j})$ is a subset of $\Delta(D_{\cdot j}, D'_{\cdot j}) \cup M_j \cup \Delta(\tilde{D}_{\cdot j}, \tilde{D}'_{\cdot j}) \cup \tilde{M}_j$, and so it has size at most d, a contradiction.

Knowledge Soundness. For knowledge soundness, we construct an extractor $\mathcal{E}$ which opens the polynomial commitment for any adversary $\mathcal{A}$ that succeeds with non-negligible probability. Although we use the same high-level structure as [14], the details will be quite different due to the use of the outer SNARK and the code-switching technique. In particular, the proof will be structured as follows:

(1) We first define the extractor $\mathcal{E}$, which provides both D and W.

(2) We show that the extractor succeeds with overwhelming probability.

(3) We show that $\mathcal{R}$ opens to a unique D and corresponding decoding C.

(4) We show that C decodes to W.

(5) We show that this implies $\mathtt{Open}(\mathtt{pp}, \mathcal{R}, \phi, \mathtt{oh}) = 1$.

(6) We show that $\vec{x_0} W \vec{x_1} = \phi(\vec{x}) = y$.

(1): We construct $\mathcal{E}$ as follows: First, $\mathcal{E}$ reads the random oracle queries and attempts to reconstruct the Merkle tree witness D. To this end, it runs the following procedure recursively, with starting input $(\mathcal{R}, 2\log(k), 0)$:

Given string h, depth d and index j, do the following:

- If $d = 0$ and $h = H(x)$ where $x \in \mathbb{F}$ is a preimage of h, write $D_j = x$.
- If $d > 0$ and $h = H(h_0 \mid\mid h_1)$, recurse with inputs $(h_0, i-1, 2j)$ and $(h_1, d-1, 2j+1)$.

Indices j missing from D_j are denoted by M and set to 0. Finally, interpret the resulting vector column-first as a $k \times k$ matrix D.

Second, $\mathcal{E}$ needs to extract the polynomial coefficients without making use of an efficient decoder. To this end, we run the adversary repeatedly for distinct uniform random vectors $\vec{\eta} \in \mathbb{F}^{\log(k)}$. In each run where we obtain a verifying proof, we extract $\vec{y_\gamma} = \sum_{i=1}^{k} \vec{\gamma}_i W_i$ from the CP-SNARK witness, with $\vec{\gamma} = \bigotimes_{i=1}^{\log(k)} (1 - \vec{\eta}_i, \vec{\eta}_i)$. After k runs, if the vectors $\vec{\gamma}$ are linearly independent, we solve the linear system to obtain W and the randomization vectors $\vec{r}_i$ for $i \in [k]$.

(2): The only way in which the extractor can fail is if the vectors $\vec{\gamma}$ are not linearly independent and so the resulting linear system cannot be solved to recover W. Fortunately, the following lemma is applicable:

Lemma 7. [14, Lemma 4.13] The probability that the rows $\left(\bigotimes_{j=1}^{k} (1 - \vec{\eta}_{ij}, \vec{\eta}_{ij})\right)$ are linearly dependent is negligible.

Here $\vec{\eta}_i$ denotes the verifier challenge from the ith iteration in the second extractor step. The lemma proof requires that the probability of the adversary convincing the verifier is at least $\sqrt{\frac{\log(k)}{|\mathbb{F}|}}$, which is the case for an adversary that succeeds with non-negligible probability. The extractor then fails with probability at most $(k-1)\sqrt{\frac{\log(k)}{|\mathbb{F}|}} \leq \mathrm{negl}(\lambda)$.

(3): From the CP-SNARK, we can also extract the columns $C_{\cdot j}$ for $j \in \hat{J}$. Denote their encoding by $\mathrm{E_C}$ as D'. In the CP-SNARK circuit, $D'_{\cdot j}$ is checked at row indices $i \in I$ with $|I|$, a set that is sampled only after these columns are committed to. Denote the entries in M corresponding to column j by M_j. By the binding property of the CP-SNARK, if $|\Delta(D_{\cdot j}, D'_{\cdot j}) \cup M_j| > \frac{\tilde{d}}{2}$ for any $j \in \hat{J}$, the adversary succeeds with probability at most $\left(1 - \frac{\tilde{d}}{2\tilde{n}}\right)^t \leq \mathrm{negl}(\lambda)$.

We thus know that if the adversary succeeds with non-negligible probability, then $C_{\cdot j}$ with $j \in \hat{J}$ in the CP-SNARK equals the unique decoding of $D_{\cdot j}$. Suppose now that there are at least $\frac{\tilde{d}}{3}$ column indices for which the adversary cannot succeed with non-negligible probability. The success probability is then at most $\left(1 - \frac{\tilde{d}}{3\tilde{n}}\right)^t \leq \mathrm{negl}(\lambda)$. This means that D has a unique decoding to C, except at fewer than $\frac{\tilde{d}}{3}$ columns and except with negligible probability.

(4): The commitment $\mathcal{R}_{\vec{c_\gamma}}$ is compared with the codeword $\vec{c_\gamma}$ in the outer SNARK, at indices $j \in J$ that are sampled only after the CP-SNARK commitment fixes its value. Let M_j be the unopenable entries in $\mathcal{R}_{\vec{c_\gamma}}$ and $\vec{c_\gamma}'$ the opening of $\mathcal{R}_{\vec{c_\gamma}}$, setting unopenable entries to 0. Suppose then that $|\Delta(\vec{c_\gamma}', \vec{c_\gamma}) \cup M_j| > \frac{\tilde{d}}{2}$. The adversary succeeds with probability at most $\left(1 - \frac{\tilde{d}}{2\tilde{n}}\right)^t \leq \mathrm{negl}(\lambda)$. It must therefore hold that $\mathcal{R}_{\vec{c_\gamma}}$ already binds $\vec{c_\gamma}$, and similarly $\mathcal{R}_{\vec{c_1}}$ binds $\vec{c_1}$.

The vector $\vec{c_\gamma}$ is claimed to be a linear combination of rows of the unique C opening D. Suppose that this $\vec{c_\gamma}$ is at least $\frac{\tilde{d}}{3}$-far from the correct linear combination. We test this linear combination in the CP-SNARK at columns $\hat{J}$ ($\mathcal{R}_{\vec{c_\gamma}}$ binds $\vec{c_\gamma}$ before $\hat{J}$ is sampled), and so again $\mathcal{A}$ succeeds with probability at most $\left(1 - \frac{\tilde{d}}{3\tilde{n}}\right)^t \leq \mathrm{negl}(\lambda)$. By the exact same logic, $\vec{c_1}$ must be $\frac{\tilde{d}}{3}$-close to the claimed linear combination.

Since $\tilde{d}$ is an integer, $\left\lfloor \frac{\tilde{d}-1}{3} \right\rfloor$ is the first integer smaller than $\frac{\tilde{d}}{3}$. Because $\vec{c_\gamma}$ is a codeword that is $\left\lfloor \frac{\tilde{d}-1}{3} \right\rfloor$-close to the claimed linear combination, Lemma 6 applies. From it, we conclude that C has

codewords for all rows, except at at most $\left\lfloor \frac{\tilde{d}-1}{3} \right\rfloor$ columns. This in particular means that every row of C decodes to a unique message, which we interpret as W and the randomization vectors.

(5): Since $\vec{c_\gamma}$ is $\frac{\tilde{d}}{3}$-close to the claimed linear combination of rows of C, it must hold that $\vec{y_\gamma} = \sum_{i=1}^{k} \vec{\gamma}_i W_i$ and $\vec{r_\gamma} = \sum_{i=1}^{k} \vec{\gamma}_i \vec{r}_i$ whenever the adversary convinces the verifier. Since the extractor succeeded, this means that $\mathcal{E}$ recovers the unique W and randomization vectors that form the message corresponding to C. We interpret W as the coefficients of polynomial ϕ. Furthermore, we have seen in (3) that the Merkle tree is well-formed at sufficiently many places that $\mathcal{E}$ obtains opening hints $\mathtt{oh}$ such that $\mathtt{Open}(\mathtt{pp}, \mathcal{R}, \phi, \mathtt{oh}) = 1$.

(6): Finally, by the same logic as (4), $\vec{c_1}$ must be $\frac{\tilde{d}}{3}$-close to the correct linear combination of rows of C, which means that it decodes to $\vec{y_1}$ and $\vec{r_1}$ and so $\vec{y_1} = \sum_{i=1}^{k} \vec{x}_{0i} W_i$ as claimed. The last check of the CP-SNARK finally ensures that $y = \sum_{j \in [k]} \vec{x}_{1j} \vec{y}_{1j}$.

Zero-Knowledge: The simulator can be found in Figure 6. First note that $\mathcal{S}$ is complete by construction: $\hat{J} \cup J$ determines at most $2t$ coefficients of $\vec{r_\gamma}$ and $\vec{r_1}$, and so it is always possible to find such polynomials. Second, it makes non-black box use of our zero-knowledge code to pick the randomness for each codeword, but makes no assumptions on (the efficient decodability of) the code underlying the transform. To see that no PPT adversary $\mathcal{A}$ can distinguish between two executions, we show that the communication is distributed identically in both $\mathrm{Real}_{\mathcal{A},\phi}$ and $\mathrm{Ideal}_{\mathcal{A},\phi,\mathcal{S}^{\mathcal{A}}}$. The elements sent by the prover are:

- Merkle commitments $\mathcal{R}$, $\mathcal{R}_{\vec{c_\gamma}}$, $\mathcal{R}_{\vec{c_1}}$ and its opening proofs. The unopened values remain statistically hidden.
- Evaluation y, which is identical in both games.
- The CP-SNARK proof π and corresponding commitment, which we compute honestly and which are in both cases indistinguishable from the output of the CP-SNARK simulator by zero-knowledge of the CP-SNARK.
- The openings D_{ij} for $I \times \hat{J}$, and $\vec{c}_{1j}$, $\vec{c}_{\gamma j}$ for $j \in \hat{J}$. Note that these are all linear combinations of columns $j \in \hat{J}$ of C. Since each row of C is a codeword of $\mathrm{E}_{\mathrm{C,ZK}}$ and we open $\left|\hat{J}\right| \leq 2t$ entries, these openings are distributed uniformly at random in both cases.

□

Proof of Lemma 5. We now show that the construction in Figure 3 is a polynomial commitment scheme even for openings at multiple adversarially chosen points.

Proof. (Lemma 5):

Completeness, binding, zero-knowledge. These follow directly from Thm. 4.

Knowledge-Soundness. By Thm. 4, any PPT adversary has negligible probability of succeeding without the extractor obtaining $\phi(\cdot)$ that opens the polynomial commitment such that $\sum_{i=0}^{m} \alpha_i \phi(\vec{x}_i) = \sum_{i=0}^{m} \alpha_i y_i = \tilde{y}$.

Due to the binding property of our underlying polynomial commitment, ϕ is fixed before $\vec{\alpha}$ is picked uniformly at random by the verifier. Suppose that there exists one i' for which $\phi(\vec{x_{i'}}) \neq y_{i'}$. Then

$$\Pr\left[\sum_{i=0}^{m-1} \alpha_i \phi(\vec{x_i}) = \sum_{i=0}^{m-1} \alpha_i y_i\right] = \Pr\left[\alpha_{i'}(\phi(\vec{x_{i'}} - y_{i'})) = -\sum_{\substack{i=0 \\ i \neq i'}}^{m-1} \alpha_i (\phi(\vec{x_i}) - y_i)\right]$$

$$= \Pr\left[\alpha_{i'} = -(\phi(\vec{x_{i'}} - y_{i'}))^{-1} \sum_{\substack{i=0 \\ i \neq i'}}^{m-1} \alpha_i (\phi(\vec{x_i}) - y_i)\right] = \frac{1}{|\mathbb{F}|},$$

and so the prover has negligible success probability. □

Function $\mathcal{S}_0^{\mathcal{A}}(1^\lambda, \mathtt{pp})$**:** To simulate a commitment, $\mathcal{S}_0$ simply runs $\mathtt{Commit}(\mathtt{pp}, \phi)$ for $\phi \equiv 0$.

Function $\mathcal{S}_1^{\mathcal{A}}(\mathtt{pp}, \mathcal{R}, \vec{x}, y, \mu)$: The simulator engages in a protocol with the verifier $\mathcal{V}$.

- $\mathcal{V}$ sends a uniformly random vector $\vec{\eta} \in \mathbb{F}^{\log(k)}$.
- $\mathcal{S}_1$ computes $\vec{\gamma} = \bigotimes_{i=1}^{\log(k)}(1 - \vec{\eta}_i, \vec{\eta}_i)$.
- $\mathcal{S}_1$ sets $\vec{y_\gamma} = \vec{0}$ and chooses $\vec{r_\gamma}$ uniformly at random under the restriction that

$$\forall j \in \hat{J} \cup J: \quad \mathrm{E}_{\mathrm{C,ZK}}(\vec{y_\gamma}; \vec{r_\gamma})_j = \sum_{i \in [k]} \vec{\gamma}_i C_{ij}.$$

- $\mathcal{S}_1$ chooses any $\vec{y_1}$ such that $y = \sum_{j \in [k]} \vec{x_1}_j \vec{y_1}_j$. It then selects $\vec{r_1}$ uniformly at random under the restriction that

$$\forall j \in \hat{J} \cup J: \quad \mathrm{E}_{\mathrm{C,ZK}}(\vec{y_1}; \vec{r_1})_j = \sum_{i \in [k]} \vec{x_0}_i C_{ij} - (\mathrm{E}_{\mathrm{C}}(\vec{y_1}) \mid\mid \vec{0})_j$$

- $\mathcal{S}_1$ continues the rest of the protocol honestly, starting with step 3.

Figure 6: The zero-knowledge simulator for the protocol in Figure 3.

5.1 Non-interactivity through Fiat-Shamir

Since the protocol in Figure 3 is public coin, we can apply the Fiat-Shamir transform [16]. In the proof of Theorem 4 we show that rewinding k times to the start of `Eval` will fully extract ϕ from its commitment.

This means that the non-interactive version of the polynomial commitment scheme is also knowledge sound. We will use different random oracles for the Merkle hash tree, the verifier challenges and (potentially) for the outer SNARK. In practice this can be implemented using domain separation. Let $\mathcal{A}$ be an adversary that is able to finish the protocol with $\mathcal{V}$ except with non-negligible probability, performing at most Q random oracle queries. We let $\mathcal{A}$ complete the protocol and in addition run the CP-SNARK extractor. By definition the CP-SNARK extractor succeeds with overwhelming probability. We obtain a transcript that verifies except with negligible probability as well as the CP-SNARK witness. We then reset the random oracle for the verifier challenges and start another, independent run of $\mathcal{A}$. After k runs, we have k verifying transcripts and $\vec{y_\gamma}$ for linearly independent $\vec{\gamma}$, and we can reconstruct W as described in the proof for Theorem 4. The adversary has performed at most kQ ROM queries in the process, in addition to the overhead of running the CP-SNARK extractor k times.

As a result we obtain a non-interactive polynomial commitment scheme. Orion uses the polynomial IOP from [27, 19]. Both works apply the Fiat-Shamir transformation but do not provide a full analysis. We leave such a detailed analysis of using Fiat-Shamir to obtain a zk-SNARK and the associated security loss to future work.

Acknowledgements

We want to thank the authors of Orion for their comments on this work. In particular for mentioning the switch from Virgo to a FRI based outer SNARK that supports batching with minimum overhead as a way to reduce the overhead for the fix that requires multiple rounds of commitments (as discussed in Section 4.1).

References

[1] Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: Lightweight sublinear arguments without a trusted setup. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017: 24th Conference on Computer and Communications Security. pp. 2087–2104. ACM Press, Dallas, TX, USA (Oct 31 – Nov 2, 2017). https://doi.org/10.1145/3133956.3134104

[2] Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: lightweight sublinear arguments without a trusted setup. Designs, Codes and Cryptography **91**(11), 3379–3424 (2023). https://doi.org/10.1007/s10623-023-01222-8

[3] Baum, C., Braun, L., Delpech de Saint Guilhem, C., Klooß, M., Orsini, E., Roy, L., Scholl, P.: Publicly verifiable zero-knowledge and post-quantum signatures from VOLE-in-the-head. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023, Part V. Lecture Notes in Computer Science, vol. 14085, pp. 581–615. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 20–24, 2023). https://doi.org/10.1007/978-3-031-38554-4_19

[4] Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Fast reed-solomon interactive oracle proofs of proximity. In: Chatzigiannakis, I., Kaklamanis, C., Marx, D., Sannella, D. (eds.) ICALP 2018: 45th International Colloquium on Automata, Languages and Programming. LIPIcs, vol. 107, pp. 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Prague, Czech Republic (Jul 9–13, 2018). https://doi.org/10.4230/LIPIcs.ICALP.2018.14

[5] Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: Goldwasser, S. (ed.) ITCS 2012: 3rd Innovations in Theoretical Computer Science. pp. 326–349. Association for Computing Machinery, Cambridge, MA, USA (Jan 8–10, 2012). https://doi.org/10.1145/2090236.2090263

[6] Bootle, J., Cerulli, A., Ghadafi, E., Groth, J., Hajiabadi, M., Jakobsen, S.K.: Linear-time zero-knowledge proofs for arithmetic circuit satisfiability. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017, Part III. Lecture Notes in Computer Science, vol. 10626, pp. 336–365. Springer, Cham, Switzerland, Hong Kong, China (Dec 3–7, 2017). https://doi.org/10.1007/978-3-319-70700-6_12

[7] Bootle, J., Chiesa, A., Groth, J.: Linear-time arguments with sublinear verification from tensor codes. In: Pass, R., Pietrzak, K. (eds.) TCC 2020: 18th Theory of Cryptography Conference, Part II. Lecture Notes in Computer Science, vol. 12551, pp. 19–46. Springer, Cham, Switzerland, Durham, NC, USA (Nov 16–19, 2020). https://doi.org/10.1007/978-3-030-64378-2_2

[8] Bootle, J., Chiesa, A., Liu, S.: Zero-knowledge IOPs with linear-time prover and polylogarithmic-time verifier. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology – EUROCRYPT 2022, Part II. Lecture Notes in Computer Science, vol. 13276, pp. 275–304. Springer, Cham, Switzerland, Trondheim, Norway (May 30 – Jun 3, 2022). https://doi.org/10.1007/978-3-031-07085-3_10

[9] Bünz, B., Fisch, B., Szepieniec, A.: Transparent SNARKs from DARK compilers. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology – EUROCRYPT 2020, Part I. Lecture Notes in Computer Science, vol. 12105, pp. 677–706. Springer, Cham, Switzerland, Zagreb, Croatia (May 10–14, 2020). https://doi.org/10.1007/978-3-030-45721-1_24

[10] Campanelli, M., Fiore, D., Querol, A.: LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019: 26th Conference on Computer and Communications Security. pp. 2075–2092. ACM Press, London, UK (Nov 11–15, 2019). https://doi.org/10.1145/3319535.3339820

[11] Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., Zaverucha, G.: Post-quantum zero-knowledge and signatures from symmetric-key primitives. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017: 24th Conference on Computer and Communications Security. pp. 1825–1842. ACM Press, Dallas, TX, USA (Oct 31 – Nov 2, 2017). https://doi.org/10.1145/3133956.3133997

[12] Chen, B., Bünz, B., Boneh, D., Zhang, Z.: HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023, Part II. Lecture Notes in Computer Science, vol. 14005, pp. 499–530. Springer, Cham, Switzerland, Lyon, France (Apr 23–27, 2023). https://doi.org/10.1007/978-3-031-30617-4_17

[13] Dao, Q., Grubbs, P.: Spartan and bulletproofs are simulation-extractable (for free!). In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023, Part II. Lecture Notes in Computer Science, vol. 14005, pp. 531–562. Springer, Cham, Switzerland, Lyon, France (Apr 23–27, 2023). https://doi.org/10.1007/978-3-031-30617-4_18

[14] Diamond, B.E., Posen, J.: Proximity testing with logarithmic randomness. IACR Communications in Cryptology (CiC) **1**(1), 2 (2024). https://doi.org/10.62056/aksdkp10

[15] Faonio, A., Fiore, D., Kohlweiss, M., Russo, L., Zajac, M.: From polynomial IOP and commitments to non-malleable zkSNARKs. In: Rothblum, G.N., Wee, H. (eds.) TCC 2023: 21st Theory of Cryptography Conference, Part III. Lecture Notes in Computer Science, vol. 14371, pp. 455–485. Springer, Cham, Switzerland, Taipei, Taiwan (Nov 29 – Dec 2, 2023). https://doi.org/10.1007/978-3-031-48621-0_16

[16] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology – CRYPTO'86. Lecture Notes in Computer Science, vol. 263, pp. 186–194. Springer, Berlin, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 1987). https://doi.org/10.1007/3-540-47721-7_12

[17] Ganesh, C., Khoshakhlagh, H., Kohlweiss, M., Nitulescu, A., Zajac, M.: What makes fiat-shamir zkSNARKs (updatable SRS) simulation extractable? In: Galdi, C., Jarecki, S. (eds.) SCN 22: 13th International Conference on Security in Communication Networks. Lecture Notes in Computer Science, vol. 13409, pp. 735–760. Springer, Cham, Switzerland, Amalfi, Italy (Sep 12–14, 2022). https://doi.org/10.1007/978-3-031-14791-3_32

[18] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: 17th Annual ACM Symposium on Theory of Computing. pp. 291–304. ACM Press, Providence, RI, USA (May 6–8, 1985). https://doi.org/10.1145/22145.22178

[19] Golovnev, A., Lee, J., Setty, S.T.V., Thaler, J., Wahby, R.S.: Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023, Part II. Lecture Notes in Computer Science, vol. 14082, pp. 193–226. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 20–24, 2023). https://doi.org/10.1007/978-3-031-38545-2_7

[20] Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge proofs from secure multiparty computation. SIAM J. Comput. **39**(3), 1121–1152 (2009). https://doi.org/10.1137/080725398, https://doi.org/10.1137/080725398

[21] Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) Advances in Cryptology – ASIACRYPT 2010. Lecture Notes in Computer Science, vol. 6477, pp. 177–194. Springer, Berlin, Heidelberg, Germany, Singapore (Dec 5–9, 2010). https://doi.org/10.1007/978-3-642-17373-8_11

[22] Kohlweiss, M., Pancholi, M., Takahashi, A.: How to compile polynomial IOP into simulation-extractable SNARKs: A modular approach. In: Rothblum, G.N., Wee, H. (eds.) TCC 2023: 21st Theory of Cryptography Conference, Part III. Lecture Notes in Computer Science, vol. 14371, pp. 486–512. Springer, Cham, Switzerland, Taipei, Taiwan (Nov 29 – Dec 2, 2023). https://doi.org/10.1007/978-3-031-48621-0_17

[23] Libert, B.: Simulation-extractable KZG polynomial commitments and applications to hyperplonk. In: Tang, Q., Teague, V. (eds.) Public-Key Cryptography - PKC 2024 - 27th IACR International Conference on Practice and Theory of Public-Key Cryptography, Sydney, NSW, Australia, April 15-17, 2024, Proceedings, Part II. Lecture Notes in Computer Science, vol. 14602, pp. 68–98. Springer (2024). https://doi.org/10.1007/978-3-031-57722-2_3, https://doi.org/10.1007/978-3-031-57722-2_3

[24] Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) Advances in Cryptology – CRYPTO'89. Lecture Notes in Computer Science, vol. 435, pp. 218–238. Springer, New York, USA, Santa Barbara, CA, USA (Aug 20–24, 1990). https://doi.org/10.1007/0-387-34805-0_21

[25] Nitulescu, A.: zk-snarks: A gentle introduction (2020), https://api.semanticscholar.org/CorpusID:211530704

[26] Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) Advances in Cryptology – CRYPTO'89. Lecture Notes in Computer Science, vol. 435, pp. 239–252. Springer, New York, USA, Santa Barbara, CA, USA (Aug 20–24, 1990). https://doi.org/10.1007/0-387-34805-0_22

[27] Setty, S.: Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020, Part III. Lecture Notes in Computer Science, vol. 12172, pp. 704–737. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 17–21, 2020). https://doi.org/10.1007/978-3-030-56877-1_25

[28] Thaler, J.: Proofs, arguments, and zero-knowledge. Foundations and Trends® in Privacy and Security **4**(2–4), 117–660 (2022). https://doi.org/10.1561/3300000030, http://dx.doi.org/10.1561/3300000030

[29] Xie, T., Zhang, Y., Song, D.: Orion: Zero knowledge proof with linear prover time. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2022, Part IV. Lecture Notes in Computer Science, vol. 13510, pp. 299–328. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 15–18, 2022). https://doi.org/10.1007/978-3-031-15985-5_11

[30] Xie, T., Zhang, Y., Song, D.: "orion: Zero knowledge proof with linear prover time". Cryptology ePrint Archive, Paper 2022/1010 (10 2022), https://eprint.iacr.org/archive/2022/1010/20231027:195131

[31] Xie, T., Zhang, Y., Song, D.: "orion: Zero knowledge proof with linear prover time". Cryptology ePrint Archive, Paper 2022/1010 (11 2022), https://eprint.iacr.org/archive/2022/1010/20231102:185740

[32] Zhang, J., Xie, T., Zhang, Y., Song, D.: Transparent polynomial delegation and its applications to zero knowledge proof. In: 2020 IEEE Symposium on Security and Privacy. pp. 859–876. IEEE Computer Society Press, San Francisco, CA, USA (May 18–21, 2020). https://doi.org/10.1109/SP40000.2020.00052