

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 resulting 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

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

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

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 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 promising development of code-based SNARKS, which achieve linear time prover complexity. While the first schemes that fall into this category [6, 7, 8] seemed to be only of theoretical interest, recent works [19, 30] 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. Breakdown [19] observes that [7] implicitly describe a polynomial commitment scheme with a linear time commitment phase (relying on linear-time encodable codes), which the authors make explicit and combine with the PIOP for R1CS from Spartan [27] to obtain a SNARK that has a linear-time prover and a square root verifier and proof size. 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 [30], which improves over this polynomial commitment scheme by reducing the verification time and proof size from $O(\sqrt{N})$ to $O(\log^2(N))$. 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 succinct proof systems in the literature.

High-level overview of Orion. The key innovation of [6] and its use as a polynomial commitment in [19] is the use of linear codes. On a high level, let $W \in \mathbb{F}^{k \times k}$ be a square matrix with $k^2 = N$, and D be the matrix obtained by encoding every row using a linear code. Computing $D_{\vec{\gamma}} = \sum_{i \in [k]} \vec{\gamma}_i D_i$ for random vector $\vec{\gamma} \in \mathbb{F}^k$ will then again be a codeword, but crucially, if any row of D was far away from a codeword, with overwhelming probability so is the resulting random linear combination $D_{\vec{\gamma}}$. This enables the following procedure to check that every row is close to a codeword: first, check that $D_{\vec{\gamma}_j} = \sum_{i \in [k]} \vec{\gamma}_i D_{ij}$ for $j \in J$ a random selection of columns and second, check that $D_{\vec{\gamma}}$ is in fact a codeword. When the procedure is then repeated with a non-random linear combination \vec{x}_0 , the resulting $D_{\vec{x}_0}$ may be decoded to yield $W_{\vec{x}_0} = \sum_{i \in [k]} \vec{x}_{0i} W_i$ in time only square root in the size of W . Taking an inner product with another vector \vec{x}_1 yields $\vec{x}_0^T W \vec{x}_1$. This is sufficient to verify a polynomial evaluation: for example, if W consists of coefficients of a degree d polynomial ϕ , then $\phi(x) = \vec{x}_0^T W \vec{x}_1$, where

$$\vec{x}_0 = \left(x^0 \quad x^1 \quad \dots \quad x^{\sqrt{d+1}-1} \right)^T, \quad \vec{x}_1 = \left(x^0 \quad x^{\sqrt{d+1}} \quad \dots \quad x^{\sqrt{d+1}(\sqrt{d+1}-1)} \right)^T.$$

A multilinear polynomial can likewise be evaluated by choosing appropriate \vec{x}_0 and \vec{x}_1 . Evaluating a polynomial using this method requires $O(\sqrt{d})$ work from the verifier. A proof size of $O(\sqrt{d})$ is achieved by having the prover commit to the matrix entries of D and only revealing the entries queried by the verifier. Crucially, the prover time is only $O(d)$, provided that the encoding algorithm of the linear code can be computed in time $O(k)$.

The Orion proof system [30] improves on this idea 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 [33] is used as the outer zk-SNARK.

To commit, the prover again parses the polynomial coefficients as matrix W . Every row is encoded using the linear code E_C , after which a random row \vec{r}_i is sampled by the prover and both added and appended to every codeword to achieve zero-knowledge. This yields matrix D . The resulting matrix is then again encoded using E_C , this time column-wise, giving matrix E . The prover finally creates a Merkle tree commitment to E to obtain the commitment \mathcal{C} 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 $\vec{\gamma} \in_R \mathbb{F}^k$ to the prover, which the prover uses to compute $D_{\vec{\gamma}} = \sum_{i \in [n]} \vec{\gamma}_i D_i$, which may be decoded to obtain $\vec{y}_{\vec{\gamma}} = \sum_{i \in [k]} \vec{\gamma}_i W_i$, a linear combination of the rows of the original witness. In Brakedown, this row vector 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. Orion uses the same idea, but performs these checks in the outer zk-SNARK. In particular, the prover computes

$$\vec{c}_{\vec{\gamma}} = \sum_{i \in [k]} \vec{\gamma}_i D_i, \quad \vec{y}_{\vec{\gamma}} = \sum_{i \in [k]} \vec{\gamma}_i W_i, \quad \vec{r}_{\vec{\gamma}} = \sum_{i \in [k]} \vec{\gamma}_i \vec{r}_i$$

and sends a commitment to $\vec{c}_{\vec{\gamma}}$ to the verifier. At this point the verifier samples a random column set J , after which the prover additionally commits to $\vec{y}_{\vec{\gamma}}$ and $\vec{r}_{\vec{\gamma}}$, as well as $D_{\cdot,j}$ for all columns $j \in J$. For this it uses the outer proof system, 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 CP-SNARK proves the relation and the consistency of the witness in the commitments. The verifier finally provides a row index set I , after which the prover generates the CP-SNARK proof. The outer SNARK circuit now performs the proximity check, that $\sum_{i \in [k]} \vec{\gamma}_i D_{ij} = (E_C(\vec{y}_{\vec{\gamma}} + \vec{r}_{\vec{\gamma}}) || \vec{r}_{\vec{\gamma}})_j$ at columns $j \in J$. This is identical to the check done in Brakedown, modulo the added zero-knowledge. The CP-SNARK also outputs $\vec{c}_{\vec{\gamma}_j}$ so that the verifier can check consistency with the earlier commitment. Because the verifier only checks this commitment at columns in J , a constant-sized set, this takes $O(1)$ work from the verifier. Finally, the proof outputs the column encoding using $E_C(D_{\cdot,j})_i$ at indices $i \in I$, with the goal that these can be checked with the commitment to E without having to open the columns entirely. Since the columns of E are codewords with sufficient distance, this makes sure that the $D_{\cdot,j}$ used in the CP-SNARK commitment matches the polynomial commitment \mathcal{C} to E . Since we have a constant number of openings, the proof size is asymptotically dominated by the proof size of the outer SNARK.

¹In [30], 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.

Issues with the soundness of Orion. Since its publication in CRYPTO 2022 [30], 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 [31].

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, which enabled the prover 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 is not 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 [32] 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 [32].

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 D 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.

The most straightforward solution to this problem would be to add another round of commitments to the protocol. Unfortunately this would either require evaluating $O(k)$ hash functions inside the outer SNARK circuit, or adding a second round of commitments to the CP-SNARK. Both will harm concrete efficiency—in the former case by making the outer SNARK circuit much larger, while in the latter case, even assuming efficient batching, by significantly increasing the proof size.

Resurrecting Orion. To remedy the unsoundness and simultaneously retain succinctness and concrete efficiency, we make a key observation: the column set J provided by the verifier has a dual purpose, which may instead be divided between two challenge sets. This way, we can use one column set J that is unknown to the prover when committing to the witness of the CP-SNARK, while using another set \hat{J} such that the prover can commit to $D_{\cdot j}$ instead of D in its entirety. By separating the challenge space in this way, we avoid adding any overhead to the polynomial commitment: the prover does not need to do more

²A third revision 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.

work, the proof size does not increase, and the verifier’s only additional work is the need to sample this new challenge set.

Applied directly however, this separation of the challenge space breaks zero-knowledge, 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, that has smaller proof sizes and a more efficient verifier.

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.

1.1 Summarizing our Contributions

We briefly summarize the contributions of this work:

- We take another look at the Orion proof system [32] 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 system Orion 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 [32] does not satisfy its 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. 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 we believe to be novel and 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 work over prime field \mathbb{F} with $|\mathbb{F}| = 2^{\Theta(\lambda)}$. 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 $w(\vec{a})$ to denote the Hamming weight

of vector \vec{a} . For vectors \vec{a} and \vec{b} of equal length n , let $\Delta(\vec{a}, \vec{b}) = \{i \in [n] \mid \vec{a}_i \neq \vec{b}_i\}$ and $d(\vec{a}, \vec{b}) = |\Delta(\vec{a}, \vec{b})|$. Similarly, for matrices A and B , we denote by $\Delta(A, B)$ the indices of the columns where A and B differ and $d(A, B) = |\Delta(A, B)|$. We write $s \in_R S$ for s sampled uniformly at random from set S .

2.1 Randomized Linear Codes

See Appendix A for preliminaries on linear codes. Following [8], we make use of randomized linear codes, which are a specific type of linear code where the message \vec{m} is concatenated with randomness \vec{r} . In particular,

Definition 2.1. Let $E_C : \mathbb{F}^{k_m+k_r} \rightarrow \mathbb{F}^n$ be a linear code. For any message $\vec{m} \in \mathbb{F}^{k_m}$, and randomness $\vec{r} \in_R \mathbb{F}^{k_r}$ let $\tilde{E}_C(\vec{m}; \vec{r}) = E_C(\vec{m} \parallel \vec{r})$. We call \tilde{E}_C a q -query uniform randomized linear code if for any q adaptively chosen locations of $\tilde{E}_C(\vec{m}; \vec{r})$ it holds that they are distributed uniformly at random.

Furthermore, we simply write $\tilde{E}_C(\vec{m})$ if $\vec{r} \in_R \mathbb{F}^{k_r}$.

An example of a uniform randomized linear code is the Reed-Solomon code $RS_{[n, k_m+q, d]}$ with $n = k_m + q + d - 1$, where n is the codeword size, $k_m + q$ the length of the message plus randomness and d the minimum distance of the code. It is q -query uniform due to 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 $s \leftarrow \text{Gen}(1^\lambda)$. Given the security parameter λ , it outputs an index $s \in I$, or equivalently a function $H_s : \{0, 1\}^{\ell(\lambda)} \rightarrow \{0, 1\}^{\ell'(\lambda)}$ 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 $s \leftarrow \text{Gen}(1^\lambda)$, 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 function³ 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, \dots, 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 $H(u \parallel v)$ 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, \dots, 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 $m_i \parallel r_i$ where r_i is chosen uniformly at random from $\{0, 1\}^\lambda$, the Merkle tree hash is also a statistically hiding commitment.

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

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

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 RGen is a tuple of algorithms $\Pi = (\text{Setup}, \text{Prove}, \text{Verify})$:

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

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

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

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

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

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

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

Commit-and-Prove SNARKs. In a commit-and-prove SNARK (CP-SNARK) one wants to prove knowledge of (u, w) such that $(u, w) \in R$ holds with respect to a witness $w = (w, \omega)$ 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 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 $(\text{Setup}, \text{Commit}, \text{Open})$, which are defined as follows:

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

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

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

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

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

Definition 2.4 (CP-SNARK [10]). Let RGen be a relation generator for relations R over $\mathcal{D}_u \times \mathcal{D}_w \times \mathcal{D}_\omega$ such that \mathcal{D}_w splits over ℓ arbitrary domains $(\mathcal{D}_1 \times \dots \times \mathcal{D}_\ell)$ for some arity parameter $\ell \geq 1$. Let $\text{Com} = (\text{Setup}, \text{Commit}, \text{Open})$ 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 Com and RGen is a zk-SNARK for a family of relations RGen^{Com} such that:

- every $R^{\text{Com}} \in \text{RGen}^{\text{Com}}$ is represented by a pair (ck, R) where $\text{ck} \in \text{Com.Setup}(1^\lambda)$ and $R \in \text{RGen}$;
- R^{Com} is over pairs (u, w) where the statement is $u := (u, (c_j)_{j \in [\ell]}) \in \mathcal{D}_x \times \mathcal{C}^\ell$, the witness is $w := ((w_j)_{j \in [\ell]}, (o_j)_{j \in [\ell]}, \omega) \in \mathcal{D}_1 \times \dots \times \mathcal{D}_\ell \times \mathcal{O}^\ell \times \mathcal{D}_\omega$, and the relation R^{Com} holds iff

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

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

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

- $\text{crs} := (\text{pk}, \text{vk}) \leftarrow \text{Setup}(\text{ck}, R, \text{aux}_R)$ generates a common reference string crs consisting of a proving key pk and a verification key vk .
- $\pi \leftarrow \text{Prove}(\text{pk}, R, \text{aux}_R, \mathbf{u}, (c_j)_{j \in [\ell]}, (w_j)_{j \in [\ell]}, (o_j)_{j \in [\ell]}, \omega)$ outputs the proof.
- $b \leftarrow \text{Verify}(\text{vk}, R, \text{aux}_R, \mathbf{u}, (c_j)_{j \in [\ell]}, \pi)$ 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 $\text{PC} = (\text{Gen}, \text{Commit}, \text{Open}, \text{Eval})$:

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

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

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

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

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:

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

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

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

where the experiments Real and Ideal are defined as follows:

$\text{Real}_{\mathcal{A}, \prec}(\text{pp})$:

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

$\text{Ideal}_{\mathcal{A}, \prec, \mathcal{S}^{\mathcal{A}}}(\text{pp})$:

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

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

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 D this is not an issue, as these are already bound by the Merkle commitment \mathcal{C} to E . However, the vectors $\vec{y}_\gamma, \vec{y}_1, \vec{r}_\gamma$ 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}_γ 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{C} 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 honestly generated commitment \mathcal{C} to a polynomial ϕ . 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 instead of the honest evaluation corresponding to $\phi(x)$. At step 5, \mathcal{P} needs to provide the CP-SNARK witness. Here the prover can forge \vec{y}_1' and \vec{r}_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 and \vec{r}_1 are validated. The vectors are also not committed to until the commit phase of the CP-SNARK in step 5. The goal of the prover is then to supply \vec{y}_1' and \vec{r}_1' in the CP-SNARK commitment, such that together with the derived $\vec{c}_1' = \text{Ec}(\vec{y}_1') + \vec{r}_1' \parallel \vec{r}_1'$ (Figure 2, step 1), the following conditions hold:

- (1) $\forall j \in J : \mathcal{C}_{\vec{c}_1}$ opens to \vec{c}_{1j}' . (Equality at J checked at Figure 2, step 1.)
- (2) $\forall j \in J : \vec{c}_{1j}' = \sum_{i \in [k]} \vec{x}_{0,i} D_{ij}$. (Figure 2, step 2.)
- (3) $y = \sum_{j \in [k]} \vec{x}_{1j} \vec{y}_{1j}'$. (Figure 2, step 4.)

Since \vec{c}_1 contained in the commitment $\mathcal{C}_{\vec{c}_1}$ has been honestly computed from D by the prover, conditions (2) and (1) can in fact be reduced to the single condition that $\forall j \in J : \vec{c}_{1j}' = \vec{c}_{1j}$, where \vec{c}_{1j}' is the vector derived from \vec{y}_1' and \vec{r}_1' in the outer SNARK. The protocol can be

$\text{Commit}(\text{pp}, \phi; r)$: 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 $D \in \mathbb{F}^{k \times 2n}$ by encoding every row of W using the encoding function E_C and hiding with \vec{r}_i as follows: $\forall i \in [k] : D_i = E_C(W_i) + \vec{r}_i \parallel \vec{r}_i$.
4. Compute $E \in \mathbb{F}^{n \times 2n}$ by encoding every column of D using E_C .
5. Commit to columns: $\forall j \in [2n] : C_j \leftarrow \text{Commit}_M(E_{\cdot j})$.
6. Output final commitment $\mathcal{C} \leftarrow \text{Commit}_M(C_1, \dots, C_{2n})$.

$\text{Eval}(\text{pp}, \mathcal{C}, X = \vec{x}_0 \otimes \vec{x}_1, y = \sum_{j=1}^k \vec{y}_j \vec{x}_{1j}, \mu, \phi)$: 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

$$\begin{aligned} \vec{c}_1 &\leftarrow \sum_{i=1}^k \vec{x}_{0i} D_i, & \vec{y}_1 &\leftarrow \sum_{i=1}^k \vec{x}_{0i} W_i, & \vec{r}_1 &\leftarrow \sum_{i=1}^k \vec{x}_{0i} \vec{r}_i, & \mathcal{C}_{\vec{c}_1} &\leftarrow \text{Commit}_M(\vec{c}_1), \\ \vec{c}_\gamma &\leftarrow \sum_{i=1}^k \vec{\gamma}_i D_i, & \vec{y}_\gamma &\leftarrow \sum_{i=1}^k \vec{\gamma}_i W_i, & \vec{r}_\gamma &\leftarrow \sum_{i=1}^k \vec{\gamma}_i \vec{r}_i, & \mathcal{C}_{\vec{c}_\gamma} &\leftarrow \text{Commit}_M(\vec{c}_\gamma). \end{aligned}$$

3. \mathcal{P} sends $\mathcal{C}_{\vec{c}_1}, \mathcal{C}_{\vec{c}_\gamma}$ and y to the verifier.
4. \mathcal{V} samples and sends column indices $J \subset [2n]$ uniformly at random, under the condition that

$$|J| = 2t, \quad |J \cap [n]| = t, \quad |J \cap [n+1 \dots 2n]| = t, \quad \neg \exists j : j, j+n \in J.$$

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

$$\vec{y}_1, \vec{y}_\gamma, \forall_{(i,j) \in [k] \times J} : D_{ij}, \vec{r}_1, \vec{r}_\gamma.$$

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 along with

$$\begin{aligned} \forall j \in J : & \text{Open}_M(\mathcal{C}_{\vec{c}_1}, j), \text{Open}_M(\mathcal{C}_{\vec{c}_\gamma}, j), \\ \forall j \in J : & \text{Open}_M(\mathcal{C}, j), \quad \forall (i, j) \in I \times J : \text{Open}_M(\mathcal{C}_j, i). \end{aligned}$$

8. \mathcal{V} checks the proof π using the CP-SNARK verification procedure.
9. \mathcal{V} checks the Merkle tree proofs of E_{ij} in \mathcal{C} for all $(i, j) \in I \times J$.
10. \mathcal{V} checks the Merkle tree proofs of \vec{c}_{1j} and $\vec{c}_{\gamma j}$ in $\mathcal{C}_{\vec{c}_1}$ and $\mathcal{C}_{\vec{c}_\gamma}$.

The verifier accepts if all checks are successful.

Figure 1: The full description of the Orion polynomial commitment protocol. Note that μ , the number of variables of ϕ , is only used implicitly by the verifier, who does not know ϕ . t is a constant derived from λ . In Commit , randomness r is used to sample rows \vec{r}_i and to randomize the commitments.

Committed witness: $\vec{y}_1, \vec{y}_\gamma, \vec{r}_1, \vec{r}_\gamma, \forall j \in J : D_{.j}$.

Public input:

- $\vec{\gamma}, x_0, x_1, y, I, J$
- $\forall j \in J : \vec{c}_{1j}, \vec{c}_{\gamma j}$
- $\forall (i, j) \in I \times J : E_{ij}$

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

1. Build encodings of row vector combinations:
 - $\vec{c}_\gamma \leftarrow \text{EC}(\vec{y}_\gamma) + \vec{r}_\gamma \parallel \vec{r}_\gamma$.
 - $\vec{c}_1 \leftarrow \text{EC}(\vec{y}_1) + \vec{r}_1 \parallel \vec{r}_1$.
 - Check for $j \in J$ that $\vec{c}_{\gamma j}$ and \vec{c}_1 match the input.
2. Check that these are valid linear combinations:
 - Check $\forall j \in J : \vec{c}_{\gamma j} \stackrel{?}{=} \sum_{i \in [k]} \vec{\gamma}_i D_{ij}$.
 - Check $\forall j \in J : \vec{c}_{1j} \stackrel{?}{=} \sum_{i \in [k]} \vec{x}_{0i} D_{ij}$.
3. Encode columns and compare:
 - $\forall j \in J : E_{.j} = \text{EC}(D_{.j})$.
 - Check for $(i, j) \in I \times J$ that E_{ij} matches the input.
4. Check claimed evaluation:
 - Check $y \stackrel{?}{=} \sum_{j \in [k]} \vec{x}_{1j} \vec{y}_{1j}$.

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.

broken as follows: we first select any vector \vec{y}_1' that satisfies condition (3).⁴ Then we need that

$$\begin{aligned} \forall j \in J \cap [n] : & \quad \vec{r}_{1j}' = \text{Ec}(\vec{y}_1)_j - \text{Ec}(\vec{y}_1')_j + \vec{r}_{1j}, \\ \forall j \in J \cap [n+1 \dots 2n] : & \quad \vec{r}_{1j}' = \vec{r}_{1j}. \end{aligned}$$

For zero-knowledge, the challenge space is restricted so that these two sets are 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 $D_i = \text{Ec}(W_i)$ and J is just sampled uniformly random from $[n]$, the protocol can still be broken. Take the generator matrix G of Ec . The condition $\vec{c}_{1j}' = \vec{c}_{1j}$ for $j \in J$ can be expressed by defining the matrix $B \in \mathbb{F}^{t \times n}$, where element B_{ij} is 1 if $J_i = j$ and 0 otherwise, for some canonical ordering J_1, J_2, \dots, J_t of J . B selects the entries of the codeword whose indices appear in J , yielding the equivalent condition $B\vec{c}_1' = B\vec{c}_1$. Condition (3) can be appended as a column to the resulting matrix to obtain the linear system

$$\begin{bmatrix} \vec{c}_{1(J_1)} \\ \vec{c}_{1(J_2)} \\ \dots \\ \vec{c}_{1(J_t)} \\ y \end{bmatrix} = \begin{bmatrix} BG^T \\ \vec{x}_1 \end{bmatrix} \begin{bmatrix} \vec{y}_{1_1}' \\ \vec{y}_{1_2}' \\ \dots \\ \vec{y}_{1_k}' \end{bmatrix}. \quad (1)$$

There are two reasons why there may not exist a solution to this system. First, it is possible that there is no other codeword equal to \vec{c}_1 at J . Second, it may be that all candidate codewords have messages that are orthogonal to \vec{x}_1 .

Let us consider the first possibility. Intuitively, there must be multiple codewords \vec{c}_1' that are equal to \vec{c}_1 at J , provided the message space is bigger than $|J|$.⁵ We define 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

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

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 if it evaluated the polynomial directly. As such, in this situation one should always opt to commit to the polynomial using just 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.⁶ As k increases, so does $\dim(A) \geq k - t$, and therefore it becomes more likely that there exists such a message. Although Orion supports arbitrary openings of the form $\vec{x}_0 W \vec{x}_1$, we consider $X = \vec{x}_0 \otimes \vec{x}_1$ corresponding to (potentially multilinear) polynomial evaluations. We have the following lemma:

Lemma 2. Let ϕ be a univariate or multilinear polynomial, and its evaluation point chosen uniformly at random. Then Eq. (1) has a solution except with negligible probability.

⁴ $\vec{x}_{11} = 1$ for the constant term of the polynomial, which means that such a \vec{y}_1' can always be found.

⁵This is analogous to how t evaluations are insufficient to uniquely define a degree $k \geq t$ polynomial.

⁶We know $\vec{x}_1 \neq \vec{0}$, so $\dim(\ker(\vec{x}_1)) \neq k$.

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 W . Take any $\vec{m}' \in A$, and suppose that $\sum_{j \in [k]} \vec{x}_{1j} \vec{m}' \neq 0$. Then

$$\vec{y}_1 + \left(y - \sum_{j \in [k]} \vec{x}_{1j} \vec{y}_{1j} \right) \left(\sum_{j \in [k]} \vec{x}_{1j} \vec{m}'_j \right)^{-1} \vec{m}'$$

is a solution to the system.

This procedure only fails if $\sum_{j \in [k]} \vec{x}_{1j} \vec{m}' = 0$. For a univariate polynomial, \vec{x}_1 has the form

$$\vec{x}_1 = (\alpha^0 \quad \alpha^1 \quad \dots \quad \alpha^{k-1})^T$$

for some $\alpha \in \mathbb{F}$ distributed uniformly at random. By the Schwartz-Zippel lemma therefore, this procedure fails with probability at most $k|\mathbb{F}|^{-1}$. For a multilinear polynomial,

$$\vec{x}_1 = (1 \quad \alpha_1) \otimes (1 \quad x_1) \otimes \dots \otimes (1 \quad x_{\log_2(k)})$$

and so by the Schwartz-Zippel lemma this procedure fails with probability at most $\log_2(k)|\mathbb{F}|^{-1}$. In both cases, this probability is negligible. \square

Clearly this attack succeeds with overwhelming probability. We note that an evaluation point chosen uniformly at random is the usual setting where polynomial commitment schemes are used in practice, and it is the setting that is used for Orion. Of course, even a small non-negligible cheating probability for the prover would already be problematic.

Furthermore, there are two ways in which even an extremely unlucky prover can simply retry: first, in the proof we only consider a single $\vec{m}' \in A$, while in reality the prover can pick another vector in A and try again. Although this works even in the interactive setting, it requires $k > t + 1$, since if A is one-dimensional the roots will be identical for all $\vec{m}' \in A$. Second, in the non-interactive setting the prover can simply resample the evaluation point. The prover will then always succeed in expected polynomial time.

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 and in terms of memory usage. Fortunately, we do not need to construct G explicitly for the attack either.

The encoding algorithm $E_C^{(d)}(\vec{x})$ 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 $\vec{c}_1 = E_C^{(d-1)}(\vec{m}_1)$ through a recursive call. Lastly $\vec{c}_2 = \vec{c}_1 A^{(d) \prime}$ 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 $A^{(d) \prime}$ 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 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.⁷

4 Restoring Soundness of Orion

The order in which challenges and commitments must be sent is naturally constrained by the protocol. First, to retain the linear prover we do not want to commit to D entirely using the CP-SNARK and so the column set J needs to be known at this time. Second, to check that D is consistent with the commitment to E , we should sample I only after committing to these columns. Third, the vector \vec{y}_γ needs to be encoded to \vec{c}_γ and compared to $\mathcal{C}_{\vec{c}_\gamma}$, but here the column vector J must be sampled afterward to ensure \vec{y}_γ 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}_γ before J is sampled, and opening this commitment in the SNARK. This can be generically achieved by creating a Merkle commitment. Unlike \vec{c}_γ however, we need to use \vec{y}_γ inside the CP-SNARK circuit, and so we need to open the commitment there. This is undesirable, since hash-computations are expensive to perform inside the 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}_γ would be committed to first, and when J is sampled, then the prover commits to D_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 (succinct) commitments to polynomials of half the size instead of one. Since 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 by moving to another commitment scheme and making use of batching. One option is to use FRI [4], which can be efficiently batched with minimal overhead in terms of proof and verification times.⁸ Even then, due to the need for two separate query phases, the proof size is still significantly increased.

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. A key observation is that the index set used to select columns of D for the CP-SNARK witness does not need to be the same as the set used

⁷An implementation of this attack can be found at <https://github.com/ThomasdenH/orion-attack>.

⁸Note that using a discrete log or pairing-based commitment scheme is undesirable, as it would render Orion no longer plausibly post-quantum secure.

to check consistency with $C_{\vec{c}_1}$ later on. This means that it is not necessary to split the CP-SNARK commitment into two phases at all. We can instead have \mathcal{V} sample a new set \hat{J} to choose the columns of D to include in the CP-SNARK witness and postpone the sampling of the set J to when it is needed to compute π , and only after the prover has committed to the CP-SNARK witness. This way, we can have the column set \hat{J} available when we commit to selected columns of D , while not allowing the prover to pick \vec{y}_γ and \vec{r}_γ such that it opens correctly precisely at J , since it would be sampled later.

This has several advantages: first, we do not increase the witness size, only the statement size of the CP-SNARK. We have one additional challenge, which is not part of the proof and thus doesn't increase its size. We don't create any new Merkle commitments or even open existing commitments at additional points at all. In fact, the only cost associated with our approach is that the verifier samples this additional fixed-size column set \hat{J} .

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 . This means that we have to switch the zero-knowledge scheme, which will actually improve verifier time and proof size, while keeping the prover time comparable.

4.3 Using Zero-knowledge Linear Codes

To remedy this issue, we will make use of 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 $|J \cup \hat{J}| - 1$. Our construction is an instance of a q -query uniform randomized linear code, which in turn is a specific case of a zero-knowledge linear code. This general transformation of any code to a perfectly zero-knowledge code is to the best of our knowledge novel and 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 $E_{C,ZK}$ to be linearly encodable in k , not in q .

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

$$\forall i \in [2n] : E_{C,ZK}^q(\vec{y}; \vec{r})_i = (E_C(\vec{y}) \parallel E_C(\vec{y}))_i + \sum_{j=1}^q \vec{r}_j i^j,$$

is the randomized linear code $[2n, k + q, 2d]$. Furthermore, if E_C is encodable in $O(n)$ operations, then $E_{C,ZK}^q$ is encodable in $O(qn)$ operations.

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

To see that the minimum distance is $2d$, restricting the message to $\vec{r} = \vec{0}$, we have

$$\min\{w(E_{C,ZK}(\vec{y}; \vec{0})) \mid \vec{y} \in \mathbb{F}^k, \vec{y} \neq \vec{0}\} = 2d.$$

Now, assume towards a contradiction that there exists a message (\vec{y}, \vec{r}) with $\vec{r} \neq \vec{0}$ such that $E_{C,ZK}(\vec{y}, \vec{r})$ has weight less than $2d$. Notice that if for some i , $f(i) = \sum_{j=1}^q \vec{r}_j i^j$ disagrees with $f(i+n)$, either $E_{C,ZK}(\vec{y}; \vec{r})_i \neq 0$ or $E_{C,ZK}(\vec{y}; \vec{r})_{i+n} \neq 0$. This means that $f(i) \neq f(i+n)$ at less than $2d$ indices $i \in [n]$ and therefore that $f(i) = f(i+n)$ at more than $n - 2d = q$ such

indices. Expanding this condition and defining $\vec{r}_0 = 0$ for simplicity, we get

$$\begin{aligned} \sum_{j=0}^q \vec{r}_j i^j &= \sum_{j=0}^q \vec{r}_j (i+n)^j = \sum_{j=0}^q \vec{r}_j \sum_{k=0}^j \binom{j}{k} i^k n^{j-k} \\ &= \sum_{k=0}^q \sum_{j=k}^q \vec{r}_j \binom{j}{k} i^k n^{j-k} = \sum_{j=0}^q \sum_{k=j}^q \vec{r}_k \binom{k}{j} i^j n^{k-j}, \end{aligned}$$

where we made a change of variables in the last step. It follows that

$$\sum_{j=0}^q \left(\vec{r}_j - \sum_{k=j}^q \vec{r}_k \binom{k}{j} n^{k-j} \right) i^j = \sum_{j=0}^q \sum_{k=j+1}^q \left[\vec{r}_k \binom{k}{j} n^{k-j} \right] i^j = 0$$

at more than q points $i \in [n]$. Since this is a polynomial in i of degree q , this means that all coefficients must be 0, so

$$\forall j \in \{0, \dots, q\} : \sum_{k=j+1}^q \binom{k}{j} n^{k-j} \vec{r}_k = 0.$$

Shifting $j+1 \rightarrow j$ and omitting $j = q+1$, we can write this condition as $A\vec{r} = \vec{0}$, where A is defined as

$$A_{jk} = \begin{cases} \binom{k}{j-1} n^{1+k-j} & \text{if } k \geq j \\ 0 & \text{otherwise} \end{cases}.$$

From the definition, we see that A is a square, upper-triangular matrix. Since we work over a large prime field, the matrix is non-zero at the diagonal and invertible. We conclude that $\vec{r} = \vec{0}$, a contradiction.

To see that it is q -query uniform, note that the polynomial's zero constant term defines its evaluation at point $i = 0$, outside the evaluation domain. 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 $E_{C,ZK}(\vec{y}; \vec{r})_{i_j} = e_j$ for all $j \in [q]$. This is because $q+1$ 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 $E_{C,ZK}(\vec{y}; \vec{r})$ takes $O(n)$ operations for encoding E_C , and then $2qn = O(qn)$ operations for computing the polynomial. \square

Using $E_{C,ZK}$ 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 $E_C(\vec{y}) + \vec{r} \parallel \vec{r}$. This means that we get the same security level for smaller proof sizes.

Interestingly, only the prover ever needs to run $E_{C,ZK}$. 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 and 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 E_C be a linear code $[n, k, d]$ with $\left(1 - \frac{d}{3n}\right)^t \leq \text{negl}(\lambda)$ and let $E_{C,ZK}^{2t}$ be the $2t$ -query uniform randomized linear code $[\tilde{n}, \tilde{k}, \tilde{d}]$ obtained from 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.

The Spielman code used in Orion has minimum relative distance $\delta = 0.055$, and so we set $t = 4795$ to satisfy the condition $\left(1 - \frac{d}{3n}\right)^t < 2^{-\lambda}$ for $\lambda = 128$.⁹

In addition to our new polynomial commitment scheme, we also make an improvement to the simulator. The one currently used in [32] 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{C} 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 concrete verifier efficiency as well as proof size, and only incurs a small penalty 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

⁹Although Orion chooses $t = 1568$, it needs to satisfy this same condition, which would correspond to $\lambda = 42$. Using the same value $t = 4795$ would have been appropriate instead. Remember that our construction only reveals half the amount of Merkle tree openings compared to Orion.

$\text{Commit}(\text{pp}, \phi; r)$: 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 $D \in \mathbb{F}^{k \times \tilde{n}}$ by encoding every row of W using the encoding function $E_{C, \text{ZK}}^{2t}(W_i; \vec{r}_i)$ for $\vec{r}_i \in_R \mathbb{F}^{2t}$.
3. Compute $E \in \mathbb{F}^{n \times \tilde{n}}$ by encoding every column of D using E_C .
4. Commit to columns: $\forall j \in [\tilde{n}] : C_j \leftarrow \text{Commit}_M(E_{\cdot j})$.
5. Output final commitment $\mathcal{C} \leftarrow \text{Commit}_M(C_1, \dots, C_{\tilde{n}})$.

$\text{Eval}(\text{pp}, \mathcal{C}, X = \vec{x}_0 \otimes \vec{x}_1, y = \sum_{j=1}^k \vec{y}_j \vec{x}_{1j}, \mu, \phi)$: 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

$$\begin{aligned} \vec{c}_1 &\leftarrow \sum_{i=1}^k \vec{x}_{0i} D_i, & \vec{y}_1 &\leftarrow \sum_{i=1}^k \vec{x}_{0i} W_i, & \vec{r}_1 &\leftarrow \sum_{i=1}^k \vec{x}_{0i} \vec{r}_i, & C_{\vec{c}_1} &\leftarrow \text{Commit}_M(\vec{c}_1), \\ \vec{c}_\gamma &\leftarrow \sum_{i=1}^k \vec{\gamma}_i D_i, & \vec{y}_\gamma &\leftarrow \sum_{i=1}^k \vec{\gamma}_i W_i, & \vec{r}_\gamma &\leftarrow \sum_{i=1}^k \vec{\gamma}_i \vec{r}_i, & C_{\vec{c}_\gamma} &\leftarrow \text{Commit}_M(\vec{c}_\gamma). \end{aligned}$$

3. \mathcal{P} sends $C_{\vec{c}_1}, C_{\vec{c}_\gamma}$ 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

$$\vec{y}_1, \vec{y}_\gamma, \vec{r}_1, \vec{r}_\gamma, \forall_{j \in \hat{J}} : D_{\cdot j}.$$

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

$$\begin{aligned} \forall j \in J : & \text{Open}_M(C_{\vec{c}_1}, j), \text{Open}_M(C_{\vec{c}_\gamma}, j), \\ \forall j \in \hat{J} : & \text{Open}_M(\mathcal{C}, j), \forall (i, j) \in I \times \hat{J} : \text{Open}_M(\mathcal{C}_j, i). \end{aligned}$$

8. \mathcal{V} checks the proof π using the CP-SNARK verification procedure.
9. \mathcal{V} checks the Merkle tree proofs of E_{ij} in \mathcal{C} for all $(i, j) \in I \times \hat{J}$.
10. \mathcal{V} checks the Merkle tree proofs of \vec{c}_{1j} and $\vec{c}_{\gamma j}$ in $C_{\vec{c}_1}$ and $C_{\vec{c}_\gamma}$.

The verifier accepts if all checks are successful.

Figure 3: The protocol for our new polynomial commitment scheme Scorpius. μ is again the number of variables of ϕ and t is derived from the security parameter. See Theorem 4 for its dependence on the other parameters. Input r is used to derive all randomness in Commit .

Committed witness: $\vec{y}_1, \vec{y}_\gamma, \vec{r}_1, \vec{r}_\gamma, \forall j \in \hat{J} : D.j$.

Public input:

- $\vec{\eta}, \vec{x}_0, \vec{x}_1, y, \hat{J}, I, J$.
- $\forall j \in J : \vec{c}_{1j}, \vec{c}_{\gamma j}$.
- $\forall (i, j) \in I \times J : E_{ij}$.

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

1. Compute $\vec{\gamma} = \bigotimes_{i=0}^{\log(k)} (1 - \vec{\eta}_i, \vec{\eta}_i)$.
2. Build encodings of row vector combinations:
 - $\vec{c}_\gamma \leftarrow E_{C,ZK}(\vec{y}_\gamma; \vec{r}_\gamma)$
 - $\vec{c}_1 \leftarrow E_{C,ZK}(\vec{y}_1; \vec{r}_1)$
 - Check for $j \in J$ that $\vec{c}_{\gamma j}$ and \vec{c}_{1j} match the input.
3. Check that these are valid linear combinations:
 - Check $\forall j \in \hat{J} : \vec{c}_{\gamma j} \stackrel{?}{=} \sum_{i \in [k]} \vec{\gamma}_i D_{ij}$.
 - Check $\forall j \in \hat{J} : \vec{c}_{1j} \stackrel{?}{=} \sum_{i \in [k]} \vec{x}_{0i} D_{ij}$.
4. Encode columns:
 - $\forall j \in \hat{J} : E_{.j} = E_C(D_{.j})$.
 - Check for $(i, j) \in I \times \hat{J}$ that E_{ij} matches the input.
5. Check claimed evaluation:
 - Check $y \stackrel{?}{=} \sum_{j \in [k]} \vec{x}_{1j} \vec{y}_{1j}$.

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.

it already needs to perform $5k$ encodings of 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 is almost halved, 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 $O(\sqrt{N})$, which is expected to significantly improve verification time as well.

4.7 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:

$$\Pr \left[\begin{array}{l} \text{pp} \leftarrow \text{Gen}(1^\lambda, \mu), (\text{pp}, \mathcal{C}, \vec{x}_1, \dots, \vec{x}_m, y_1, \dots, y_{m-1}) \leftarrow \mathcal{A}(\text{pp},) \\ 1 = \langle \mathcal{A}, \mathcal{V}_{\text{Eval}} \rangle(\text{pp}, \mathcal{C}, \vec{x}_1, \dots, \vec{x}_m, y_1, \dots, y_m, \mu, \phi), \\ (\phi, \text{oh}) \leftarrow \mathcal{E}^{\mathcal{A}}(\text{pp}, (\mathcal{C}, \vec{x}_1, \dots, \vec{x}_m, y_1, \dots, y_m)) : \\ \phi(\vec{x}_1) \neq y_1 \vee \dots \vee \phi(\vec{x}_m) \neq y_m \vee \text{Open}(\text{pp}, \mathcal{C}, \phi, \text{oh}) \neq 1 \end{array} \right] \approx_\lambda 0.$$

We provide a proof in Section 5.

$\text{Eval}_m(\text{pp}, \mathcal{C}, (X_1 = x_{1,0}^{\vec{x}} \otimes x_{1,1}^{\vec{x}}, \dots, X_m = x_{m,0}^{\vec{x}} \otimes x_{m,1}^{\vec{x}}), (y_1, \dots, y_m), \mu, \phi)$:
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 $X' = \sum_{i=1}^m \vec{\alpha}_i X_i$.
- \mathcal{P} and \mathcal{V} compute $y' = \sum_{i=1}^m \vec{\alpha}_i y_i$.
- \mathcal{P} and \mathcal{V} run the protocol $\text{Eval}(\text{pp}, \mathcal{C}, X', y', \mu, \phi)$.

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

4.8 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 a 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}_γ 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

Proximity Testing. Orion uses the result [2, Lemma A.1] that if a single row of matrix D is far away from a codeword, then a uniformly random linear combination will also be far away with high probability. Or conversely, that if such a 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. For a logarithmic verifier, this linear combination can instead be built as a tensor product from $\log(k)$ uniformly random field elements. We use the following lemma due to Diamond and Posen:

Lemma 6. [14, Theorem 3.1] Fix an arbitrary $[n, k, d]$ -code E_C , and a proximity parameter $e \in \left\{0, \dots, \left\lfloor \frac{d-1}{3} \right\rfloor\right\}$. If $D \in \mathbb{F}^{k \times n}$ satisfies

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

then $d(D, E_C^k) \leq e$.

Merkle Tree Extractor. Like [14], we need two extraction strategies. First, we read ROM-queries to open the Merkle tree commitment \mathcal{C} to obtain both the encoded matrix E and the commitment randomness. This is needed as input by the `Open` algorithm to verify that \mathcal{C} binds ϕ . In particular, we define the extractor \mathcal{E}_{MT} that runs the following procedure recursively, with starting input $(\mathcal{C}, 2 \log(k), 0)$:

Given string h , depth d , index j , and initially empty set R , do the following:

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

Indices j missing from E_j are denoted by M and set to 0. Finally, interpret the resulting vector column-first as a $k \times k$ matrix E . Return matrix E and commitment randomness R .

Extraction from \vec{y}_γ . Unfortunately, even though the extracted E will correspond to a unique ϕ , by assumption there does not need to exist an efficient decoding algorithm for the linear code, and so the extractor is not able to obtain this ϕ by decoding. This means that the Merkle tree extractor alone is not enough to obtain D and ultimately ϕ . We will therefore use rewinding in order to obtain k vectors \vec{y}_γ and \vec{r}_γ 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 `Open(pp, C, φ, oh)` as follows: compute D honestly from ϕ and the randomization vectors $(\vec{r}_i)_{i=1}^k$ (contained in `oh`) and encode vertically to obtain E . Also using `oh`, verify the Merkle tree opening \tilde{E} . For every column j , let M_j be the set of

indices without a valid opening. Output 1 if, for more than $\tilde{n} - \frac{\tilde{d}}{3}$ columns j , it holds that $|\Delta(\tilde{E}_{\cdot,j}, E_{\cdot,j}) \cup M_j| < \frac{\tilde{d}}{2}$, and 0 otherwise.

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

On the other hand, since there are more than $\tilde{n} - \frac{\tilde{d}}{3}$ columns for which $|\Delta(\tilde{E}_{\cdot,j}, E_{\cdot,j}) \cup M_j| < \frac{\tilde{d}}{2}$, there are also more than $\tilde{n} - \frac{2\tilde{d}}{3}$ columns j for which

$$\begin{aligned} d &> |\Delta(\tilde{E}_{\cdot,j}, E_{\cdot,j}) \cup M_j| + |\Delta(\tilde{E}'_{\cdot,j}, E'_{\cdot,j}) \cup M'_j| \\ &\geq |\Delta(\tilde{E}_{\cdot,j}, E_{\cdot,j}) \cup \Delta(\tilde{E}_{\cdot,j}, \tilde{E}'_{\cdot,j}) \cup \Delta(\tilde{E}'_{\cdot,j}, E'_{\cdot,j}) \cup M_j \cup M'_j| \\ &\geq |\Delta(E_{\cdot,j}, E'_{\cdot,j}) \cup M_j \cup M'_j| \\ &\geq d(E_{\cdot,j}, E'_{\cdot,j}). \end{aligned}$$

In the second step we used that $\Delta(\tilde{E}, \tilde{E}') \subseteq M_j \cup M'_j$, i.e. the Merkle tree openings can only differ in their missing entries. By the pigeonhole principle there exist at least one column j for which $d > d(E_{\cdot,j}, E'_{\cdot,j}) \geq d$, 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 obtains both E and W .
- (2) We show that the extractor succeeds with overwhelming probability.
- (3) We show that \mathcal{C} opens to a unique E and corresponding decoding D .
- (4) We show that D decodes to W .
- (5) We show that this implies $\text{Open}(\text{pp}, \mathcal{C}, \phi, \text{oh}) = 1$.
- (6) We show that $\vec{x}_0 W \vec{x}_1 = \phi(\vec{x}) = y$.

(1): We construct \mathcal{E} as follows: First, we call Merkle tree extractor \mathcal{E}_{MT} as a subroutine to obtain E and commitment randomness R . 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 $\vec{y}_{\vec{\eta}}$ from the CP-SNARK witness, with $\vec{\gamma} = \bigotimes_{i=1}^{\log(k)} (1 - \vec{\eta}_i, \vec{\eta}_i)$. Since $\vec{y}_{\vec{\eta}}$ is claimed to equal $\sum_{i \in [k]} \vec{\gamma}_i W_i$, 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 $\left(\bigotimes_{j=1}^k (1 - \vec{\eta}_{i_j}, \vec{\eta}_{i_j}) \right)$ are linearly dependent is negligible.

Here \vec{r}_i denotes the verifier challenge from the i th 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 $(k-1)\sqrt{\frac{\log(k)}{|\mathbb{F}|}} \leq \text{negl}(\lambda)$. This means that the extractor succeeds in obtaining a matrix W' .

(3): By the minimum distance of the code E_C , for all $j \in [\tilde{n}]$, column $E_{\cdot j}$ has a Hamming distance of less than $\frac{d}{2}$ to at most one codeword in E_C . There could still be many columns with no such codeword, however. We can define D by choosing for each column j , the value $D_{\cdot j}$ for which $d(E_C(D_{\cdot j}), E_{\cdot j}) < \frac{d}{2}$ if it exists, and setting $D_{\cdot j} = \vec{0}$ otherwise. This D is uniquely defined for E , which was extracted from the Merkle tree. Note again that, even though D is defined, the extractor cannot obtain it directly by decoding.

The adversary claims to commit to these columns in the CP-SNARK. We will denote their values by $D'_{\cdot j}$ and their (column-wise) encoding as computed by the CP-SNARK by E_C as $E'_{\cdot j}$. In the CP-SNARK circuit, $E'_{\cdot j}$ is then compared with the $E_{\cdot j}$ in the Merkle tree commitment at row indices $i \in I$ with $|I| = t$, a set that is sampled (step 6) only after these columns $E'_{\cdot j}$ are committed to (step 5). Let M_j be the entries i in the column j for which \mathcal{E}_{MT} could not extract E_{ij} , i.e. for which there does not exist a valid opening. By the binding property of the CP-SNARK, if $|\Delta(E_{\cdot j}, E'_{\cdot j}) \cup M_j| \geq \frac{d}{2}$ for any $j \in \hat{J}$, the adversary succeeds with probability at most $\left(1 - \frac{d}{2n}\right)^t \leq \text{negl}(\lambda)$. We thus know that if the adversary succeeds with non-negligible probability, then for all $j \in \hat{J}$ in the CP-SNARK, $D'_{\cdot j} = D_{\cdot j}$.

(4): The commitment $\mathcal{C}_{\vec{c}_\gamma}$ is compared with the codeword \vec{c}_γ , computed in the outer SNARK, at indices $j \in J$, again with $|J| = t$. Let \vec{c}'_γ be the opening of $\mathcal{C}_{\vec{c}_\gamma}$, setting unopenable entries to 0. Suppose then that $d(\vec{c}'_\gamma, \vec{c}_\gamma) \geq \frac{\tilde{d}}{2}$. The vector \vec{c}_γ in the CP-SNARK is a codeword computed from \vec{y}_γ and \vec{r}_γ , which are committed to in step 5, and \vec{c}'_γ is bound by $\mathcal{C}_{\vec{c}_\gamma}$, which is sent in step 3. Meanwhile, J is only sampled in step 6. Therefore, the adversary succeeds with probability at most $\left(1 - \frac{\tilde{d}}{2n}\right)^t \leq \text{negl}(\lambda)$. It must therefore hold that \vec{c}'_γ contained in $\mathcal{C}_{\vec{c}_\gamma}$ has $d(\vec{c}'_\gamma, \vec{c}_\gamma) < \frac{\tilde{d}}{2}$ and thus corresponds to and binds the same message. Similarly, $\mathcal{C}_{\vec{c}_1}$ binds the same message as \vec{c}_1 .

The vector \vec{c}_γ is claimed to be a linear combination of the rows of D . Suppose that there are at least $\frac{\tilde{d}}{3}$ columns j for which there does not exist $D_{\cdot j}$ such that both $|\Delta(E_{\cdot j}, E_C(D_{\cdot j})) \cup M_j| < \frac{d}{2}$ and $\vec{c}_{\gamma j} = \sum_{i \in [k]} \tilde{\gamma}_i D_{ij}$. Note that there can be either one or no such $D_{\cdot j}$. We test both conditions at $j \in \hat{J}$ with $|\hat{J}| = t$. Since \mathcal{C} binds E , $\mathcal{C}_{\vec{c}_\gamma}$ binds \vec{c}_γ at step 3 and \hat{J} is sampled afterward at step 4, \mathcal{A} succeeds with probability at most $\left(1 - \frac{\tilde{d}}{3n}\right)^t \leq \text{negl}(\lambda)$. We conclude that for less than $\frac{\tilde{d}}{3}$ columns j either $|\Delta(E_{\cdot j}, E_C(D_{\cdot j})) \cup M_j| \geq \frac{d}{2}$ or $\vec{c}_{\gamma j} \neq \sum_{i \in [k]} \tilde{\gamma}_i D_{\cdot j}$. The same reasoning applies for \vec{c}_1 .

Since \tilde{d} is an integer, $\lfloor \frac{\tilde{d}-1}{3} \rfloor$ is the largest integer smaller than $\frac{\tilde{d}}{3}$. Because $d(\vec{c}_\gamma, \sum_{i \in [k]} \tilde{\gamma}_i D_{\cdot j}) \leq \lfloor (\tilde{d}-1)/3 \rfloor$, Lemma 6 applies. From it, we conclude that D has codewords for all rows, except at at most $\lfloor \frac{\tilde{d}-1}{3} \rfloor$ columns. This in particular means that every row of D has a unique closest codeword. We interpret the corresponding messages as the witness rows W_i and the randomization vectors \vec{r}_i .

(5): Since for less than $\frac{\tilde{d}}{3}$ columns j , $D_{ij} \neq \text{E}_{\mathcal{C}, \text{ZK}}(W_i \parallel \vec{r}_i)_j$ for all $i \in [k]$, also

$$\begin{aligned} \frac{\tilde{d}}{3} &> d \left(\sum_{i \in [k]} \tilde{\gamma}_i \text{E}_{\mathcal{C}, \text{ZK}}(W_i \parallel \vec{r}_i)_j, \sum_{i \in [k]} \tilde{\gamma}_i D_i \right) \\ &= d \left(\text{E}_{\mathcal{C}, \text{ZK}} \left(\sum_{i \in [k]} \tilde{\gamma}_i W_i; \sum_{i \in [k]} \tilde{\gamma}_i \vec{r}_i \right), \sum_{i \in [k]} \tilde{\gamma}_i D_i \right). \end{aligned}$$

We also have $d(\vec{c}_\gamma, \sum_{i \in [k]} \tilde{\gamma}_i D) < \frac{\tilde{d}}{3}$, and therefore the triangle inequality implies they are actually the same codeword:

$$\vec{c}_\gamma = \text{E}_{\mathcal{C}, \text{ZK}} \left(\sum_{i \in [k]} \tilde{\gamma}_i W_i; \sum_{i \in [k]} \tilde{\gamma}_i \vec{r}_i \right),$$

and so they correspond to the same message

$$\vec{y}_\gamma \parallel \vec{r}_\gamma = \sum_{i \in [k]} \tilde{\gamma}_i (W_i \parallel \vec{r}_i). \quad (2)$$

The extractor described in (1) obtained W' and vectors \vec{r}'_i by solving the linear system defined by these values \vec{y}_γ , \vec{r}_γ , and linearly independent $\tilde{\gamma}$. Since \vec{y}_γ is well-formed according to Eq. (2) except with overwhelming probability, the reconstructed matrix W' must equal W , which was uniquely defined for the matrix E extracted from the Merkle commitment \mathcal{C} . Also, for all i , $\vec{r}'_i = \vec{r}_i$.

By (4), there are less than $\frac{\tilde{d}}{3}$ columns j such that $|\Delta(E_{\cdot j}, \text{E}_{\mathcal{C}}(D_{\cdot j})) \cup M_j| \geq \frac{d}{2}$ or $\vec{c}_{\gamma_j} \neq \sum_{i \in [k]} \tilde{\gamma}_i D_{\cdot j}$. Let D' be the matrix obtained by honestly encoding every row $W_i \parallel \vec{r}_i$. Then, since the failing column set is bound by \mathcal{C} and thus stays constant across rewinds, $D' = D$ except at those columns, and so $|\Delta(E_{\cdot j}, E'_{\cdot j}) \cup M_j| < \frac{d}{2}$ except at less than $\frac{\tilde{d}}{3}$ columns, where E' is the honest encoding of D' . We will set $\text{oh} = (E, R, (\vec{r}_i)_i)$, i.e. we include the Merkle tree opening as well as the randomization used for $\text{E}_{\mathcal{C}, \text{ZK}}$. Together, this means $\text{Open}(\text{pp}, \mathcal{C}, \phi, \text{oh}) = 1$.

(6): Finally, by the same logic as (5), \vec{c}_1 decodes to $\vec{y}_1 = \sum_{i=1}^k \vec{x}_{0i} W_i$ and $\vec{r}_1 = \sum_{i=1}^k \vec{x}_{0i} \vec{r}_i$ and so \vec{y}_1 is correctly evaluated. 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}_γ 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 $\text{Real}_{\mathcal{A}, \phi}$ and $\text{Ideal}_{\mathcal{A}, \phi, \mathcal{S}^{\mathcal{A}}}$. The elements sent by the prover are:

- Merkle commitments \mathcal{C} , $\mathcal{C}_{\vec{c}_\gamma}$, $\mathcal{C}_{\vec{c}_1}$ 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 E_{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 D . Since each row of D is a codeword of $E_{C,ZK}$ and we open $|\hat{J}| \leq 2t$ entries, these openings are distributed uniformly at random in both cases.

□

Function $S_0^A(1^\lambda, pp)$: To simulate a commitment, S_0 simply runs $\text{Commit}(pp, \phi)$ for $\phi \equiv 0$.

Function $S_1^A(pp, C, \vec{x}, y, \mu)$: S_1^A obtains $\hat{J} \cup J$ from \mathcal{A} 's random tape. The simulator then engages in the following protocol with the verifier \mathcal{A} :

- \mathcal{A} sends a uniformly random vector $\vec{\eta} \in \mathbb{F}^{\log(k)}$.
- S_1 computes $\vec{\gamma} = \otimes_{i=1}^{\log(k)} (1 - \vec{\eta}_i, \vec{\eta}_i)$.
- S_1 sets $\vec{y}_\gamma = \vec{0}$ and chooses \vec{r}_γ uniformly at random under the restriction that

$$\forall j \in \hat{J} \cup J: \quad E_{C,ZK}(\vec{y}_\gamma; \vec{r}_\gamma)_j = \sum_{i \in [k]} \vec{\gamma}_i D_{ij}.$$

- S_1 chooses any \vec{y}_1 such that $y = \sum_{j \in [k]} \vec{x}_{1j} \vec{y}_{1j}$. It then selects \vec{r}_1 uniformly at random under the restriction that

$$\forall j \in \hat{J} \cup J: \quad E_{C,ZK}(\vec{y}_1; \vec{r}_1)_j = \sum_{i \in [k]} \vec{x}_{0i} D_{ij} - (E_C(\vec{y}_1) \parallel \vec{0})_j$$

- 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.

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 $\sum_{i=1}^m \alpha_i \phi(\vec{x}_i) = \sum_{i=1}^m \alpha_i y_i = \vec{y}$.

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

$$\begin{aligned} \Pr \left[\sum_{i=1}^m \alpha_i \phi(\vec{x}_i) = \sum_{i=1}^m \alpha_i y_i \right] &= \Pr \left[\alpha_{i'} (\phi(\vec{x}_{i'} - y_{i'})) = - \sum_{\substack{i=1 \\ i \neq i'}}^m \alpha_i (\phi(\vec{x}_i) - y_i) \right] \\ &= \Pr \left[\alpha_{i'} = -(\phi(\vec{x}_{i'} - y_{i'}))^{-1} \sum_{\substack{i=1 \\ i \neq i'}}^m \alpha_i (\phi(\vec{x}_i) - y_i) \right] = \frac{1}{|\mathbb{F}|}, \end{aligned}$$

and so \mathcal{A} succeeds with negligible probability. \square

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}_γ 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). We also want to thank the anonymous reviewers for their feedback.

References

- [1] Ames, S., Hazay, C., Ishai, Y., Venkatasubramanian, M.: Ligerio: 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., Venkatasubramanian, M.: Liger: 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., Kloof, 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. Leibniz International Proceedings in Informatics (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/10.1561/33000000030>, <https://dx.doi.org/10.1561/33000000030>
- [29] Van Lint, J.H.: *Introduction to coding theory*, vol. 86. Springer Science & Business Media (2012)
- [30] 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
- [31] 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>

- [32] 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>
- [33] 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>

A Preliminaries on Linear Codes

We provide some preliminaries on Linear Codes.

Definition A.1. [29, Definition 3.2.1] A *linear code* C is a linear subspace of \mathbb{F}^n . If C has dimension k then C is called an $[n, k]$ code.

The minimum distance of a code is the minimum distance between any two codewords.

Definition A.2. [29, Definition 3.1.2] The *minimum distance* of a nontrivial code C is

$$\min\{d(\vec{x}, \vec{y}) \mid \vec{x}, \vec{y} \in C, \vec{x} \neq \vec{y}\}.$$

An $[n, k]$ code with minimum distance d is also called an $[n, k, d]$ code. We denote by E_C the encoding function for C that maps a message $\vec{m} \in \mathbb{F}^k$ to a codeword $\vec{c} \in \mathbb{F}^n$. Finally, we make use of generator matrices:

Definition A.3. [29, Definition 3.2.2] A *generator matrix* G for a linear code C is a k by n matrix for which the rows are a basis of C .

Any codeword \vec{c} of $[n, k, d]$ code C can be generated from its encoding function E_C as $\vec{c} = E_C(\vec{m}) = \vec{m}G$ for some message $m \in \mathbb{F}^k$.